

AN MDA-BASED METHOD FOR DESIGNING INTEGRATION PROCESS MODELS IN B2B COLLABORATIONS

Ivanna M. Lazarte, Pablo D. Villarreal

CIDISI, National Technological University Santa Fe Faculty, Lavaise 610, S3004EWB, Santa Fe, Argentina

Omar Chiotti

INGAR, National Council of Scientific and Technological Research, Avellaneda 3657, S3002GJC, Santa Fe, Argentina

Lucinéia Heloisa Thom, Cirano Iochpe

Institute of Informatics, Federal University of Rio Grande do Sul, Bento Gonçalves 9500, 91501-970, Porto Alegre, Brazil

Keywords: Business-to-Business, Collaborative Business Process, Integration Business Process, Model-Driven Architecture, Workflow Activity Patterns.

Abstract: The design of integration processes is a key issue for implementing collaborative business processes in Business-to-Business collaborations. A collaborative process is executed through the enactment of the integration process of each organization, which contains the public and private logic required to support the role an organization performs in the collaborative process. Integration process models must be aligned with and derived from their corresponding collaborative process models to guarantee interoperability among organizations. In this work, we propose a method based on a Model-Driven Architecture to enable organizations to support and automate the design of integration process models. This method provides a model transformation process that uses Workflow Activity Patterns to generate the public/private activities required in integration processes to support cross-organizational message exchanges.

1 INTRODUCTION

In current global markets, organizations are implementing Business-to-Business (B2B) collaborations with their business partners to improve their performance and competitiveness. A B2B collaboration requires collaborative business processes to be designed at the business level and implemented at the technological level. A *collaborative business process* (CBP) defines the global view of the interactions among organizations to achieve common business goals (e.g. decrease inventory average levels) (Villarreal et al., 2007; Bauer et al., 2005; Roser and Bauer, 2005).

The design and implementation of B2B collaborations involves the fulfillment of particular requirements such as the ability to cope with changes, decentralized management of CBPs, peer-to-peer interactions, preservation of organizations' autonomy, representation of complex negotiations, the required support for interoperability, and the alignment between

business solution and technological solution so that the technological solution provides full support to CBPs.

In order to implement a CBP in a decentralized way, each organization (or business partner) has to define its integration business process. An *integration process* (also called *private* (OMG, 2010), *executable* (Bauer et al., 2005) or *orchestration process* (Weske, 2007)) specifies the public and private behavior that supports the role an organization performs in a CBP. It contains, from the viewpoint of an organization, the public and private logic required to process or generate the information exchanged with its partners.

CBPs and integration processes should be modeled at a high level of abstraction and in a technology-independent way to facilitate design and communication among stakeholders as well as to enable their implementation in different platforms (Lazarte et al., 2010). Designing integration processes is a highly complex, time consuming and error prone task, which

requires business analysts' modeling expertise and knowledge to identify and add the public and private logic that supports cross-organizational message exchange.

In addition, integration process models of organizations must be aligned and consistent with the public logic defined in its corresponding CBP model, and hence, they must be correctly defined in order to guarantee their interoperability and provide full support to the CBP.

In this work, we propose a method based on the Model-Driven Architecture (OMG, 2003) for generating integration process models from a CBP model. The main goals of this method are twofold: (1) to provide an automated support for the design of integration process models required by the organizations for implementing CBPs, (2) to enable organizations to define integration processes which are interoperable and consistent with the CBPs. By *interoperability* we mean the capability of organizations' integration processes for interacting with each other in a synchronized way for executing a CBP. By *consistency* we mean the coherence between the behavior defined in an integration process and its corresponding CBP.

To automatically generate the public and private logic of an integration process model, this method provides a model-to-model transformation process that makes use of the Workflow Activity Patterns (Thom et al., 2009). These patterns represent recurrent business functions frequently found in business processes (e.g. task execution request, notification activity, approval). The use of these activity patterns enables reusing the process knowledge captured in them and generating the activities and process logic required in integration process models.

The proposed method uses CBP models defined with the UP-ColBPIP language (Villarreal et al., 2007; Villarreal et al., 2010) and generates integration process models defined with the Business Process Model and Notation (BPMN) language (OMG, 2010). CBPs are modeled as interaction protocols with the UP-ColBPIP language. In an interaction protocol, a *speech act* (e.g. agree, refuse, propose) represents sender's intention with respect to the business document being exchanged in a message (Villarreal et al., 2007; Villarreal et al., 2010). In this work, the semantics of speech acts is analyzed and used for determining the activity patterns applied in each defined transformation rule to derive an integration process model from a CBP model.

The remainder of the paper is organized as follows: Section 2 describes features of the CBP models and the language used to model them. Applied activity patterns are discussed in Section 3. Section 4

describes the MDA-based method for designing integration process model, and Section 5 presents a case study of that method. Section 6 discusses related work. Section 7 draws conclusions and future work.

2 COLLABORATIVE BUSINESS PROCESS MODELS

CBPs must be modeled using concepts that are closer to the B2B collaboration domain and in an independent way of the implementation technology. To support this, CBPs are modeled with UP-ColBPIP language (UML Profile for Collaborative Business Processes based on Interaction Protocols) (Villarreal et al., 2007; Villarreal et al., 2010). This language fulfills the requirements for B2B collaborations mentioned in Section 1. By using UP-ColBPIP language, the behavior of CBPs is modeled as interaction protocols. An *interaction protocol* describes a high-level communication pattern through a choreography of business messages between organizations who play different roles. A *business message* represents an interaction between two organizations. The message choreography describes the global view of interactions between the roles played by the organizations.

Interaction protocols define not only the information exchange but also the communication of actions between organizations (Villarreal et al., 2007; Villarreal et al., 2010). Coordination and communication aspects of B2B interactions are represented in interaction protocols through the use of *speech acts* (Table 1) associated to business messages. Speech acts provide semantics to the business messages. This enables the definition of complex negotiations and avoids the ambiguity in the semantics and understanding of the business messages of CBPs.

The UP-ColBPIP language was defined as a UML Profile in order to provide well-known graphical notations for modeling CBP that were easily understood by business analysts and system designers. This language encourages a top-down approach to model CBPs and supports the modeling of five views: *B2B Collaboration View*, *Collaborative Business Process View*, *Interaction Protocol View*, *Business Document View*, and *Business Interface View*. For space reasons, this work only includes the description of the *Interaction Protocol View* that is used to define the behavior of CBPs. More details about this language can be found in (Villarreal et al., 2007; Villarreal et al., 2010).

UP-ColBPIP language extends the semantics of UML2 Interactions to model interaction protocols. Accordingly, they are defined using UML2 Sequence

Table 1: Main speech acts used by UP-ColBPIP language.

<i>Accept-Proposal</i> : represents the action of accepting a previously submitted proposal to perform an action.
<i>Agree</i> : represents the action of agreeing to perform some action, possibly in the future. It expresses an agreement on a previously submitted request to perform some action.
<i>Call-for-Proposal</i> : represents the action of a role requesting another role a proposal to perform some action.
<i>Inform</i> : represents the action of a role informing and communicating another role a given information.
<i>Propose</i> : represents the action of submitting a proposal to perform a certain action. It is used to make a proposal or respond to an existing proposal in a negotiation process.
<i>Refuse</i> : represents the action of refusing to perform a given action, and explaining the reason for the refusal.
<i>Reject-Proposal</i> : represents the action of rejecting a previously submitted proposal to perform some action.
<i>Request</i> : represents the action of a role requesting another role to perform some action. The document associated with the act describes the action to be executed.

Diagrams. The main conceptual elements used to define interaction protocols are described as follows.

Partners (organizations) and the role they fulfill are represented through lifelines. Business messages are the basic building blocks of interaction protocols. A business message defines a one-way asynchronous interaction between two roles, a sender and a receiver. It contains a business document (the exchanged information) and a speech act (the semantics of the message). Table 1 describes the speech acts used, which were obtained from the speech act library of the Agent Communication Language (FIPA, 2002).

A Protocol Reference represents a sub-protocol or nested protocol. Protocols have an implicit termination. A Termination represents an explicit end event of a protocol, which can be labeled as success or failure.

A Control Flow Segment represents complex message sequences. It contains a control flow operator and one or more interaction paths. An interaction path contains an ordered sequence of protocol elements: messages, termination events, protocol references and nested control flow segments. The semantics of the control flow segments depends on the operator being used: And, Xor, Or, Loop, Exception, Cancel, Multiple Instances, and If.

As an example, Figure 1 presents the Collaborative Demand Forecast interaction protocol, which describes a CBP of a Vendor-Managed Inventory collaborative model. This protocol represents a simple negotiation process between a customer and a supplier to determine a demand forecast. The process begins with the customer, who requests a collaborative demand forecast to the supplier.

The generated request message conveyed the data to be considered in the forecast (e.g. products, time-frame). The supplier processes the request and may respond by accepting or rejecting it. In case it is accepted, the supplier undertakes to realize the required forecast; otherwise, the process finalizes with a fail-

ure. If the supplier accepts the request, the customer informs, in parallel, a sales forecast of points of sales (POS) and planned sales policies. With this information, the supplier generates a demand forecast and sends it to the customer. Then, the process finishes.

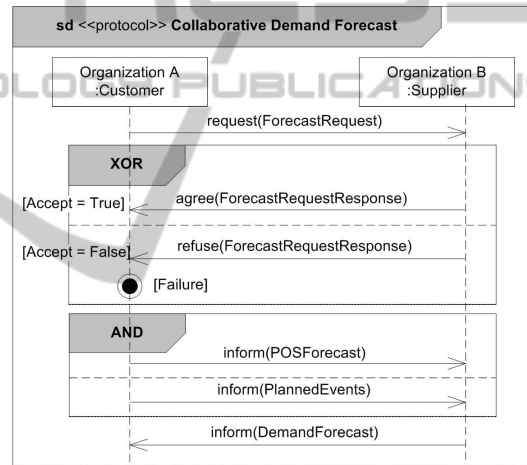


Figure 1: Collaborative Demand Forecast protocol.

3 PROCESS PATTERNS FOR DESIGNING INTEGRATION PROCESSES

An integration process model defines the particular view of an organization participating in a CBP, and contains the public and private logic and activities. Public activities are those that support the exchange of messages with other organizations. Private activities comprise data transformations and/or activities executed either by internal information systems or by human beings.

In order to facilitate the design task of integration processes, process patterns can be used. Process patterns are behavioral and functional patterns intended to increase process model quality (Eriksson

and Penker, 2000). A *pattern* is a generalized solution that can be implemented and applied in a given context in order to resolve a recurrent problem. The use of process patterns facilitates the design of process models, reduces process modeling time and cost, and encourages the reuse of (fragment of) process model (Gschwind et al., 2008).

In this work, we use the process patterns proposed in (Thom et al., 2009) and called Workflow Activity Patterns for supporting cross-organizational business message exchange and ensuring interoperability in the message exchange.

A *Workflow Activity Pattern* (activity pattern for short) refers to the description of a recurrent business function (e.g., notification, task execution request) as it can be frequently found in business processes (Thom et al., 2009). Seven activity patterns were identified based on an extensive literature study about business process types. These activity patterns are: *Approval*, *Question-answer*, *Uni- / Bi-directional Performative*, *Information Request*, *Notification*, and *Decision Making*. This pattern set is close to the vocabulary and abstraction level at which business processes are usually described by domain experts. This fosters pattern reuse when modeling business processes and therefore contributes to more standardized and better comparable business process models (Thom et al., 2009). The frequency of co-occurring activity patterns in real world process models was also studied (Lau et al., 2009).

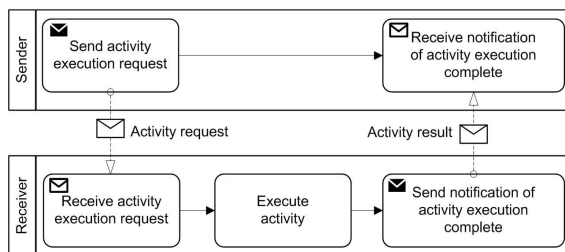


Figure 2: Bi-Directional Performative (Single-Request-Response variant) illustrated with the BPMN language.

A sample of activity pattern used in this work is *Bi-Directional Performative* (see Figure 2): A sender requests the execution of a particular activity from another role (e.g. a human or a software agent) involved in the process. The sender waits until the receiver notifies him that the requested activity has been performed.

The complete set of activity patterns can be found in (Thom et al., 2009).

4 AN MDA-BASED METHOD FOR GENERATING INTEGRATION PROCESS MODELS

To automate the design of integration process models we propose a method that applies the principles of the Model-Driven Architecture (MDA) (OMG, 2003). MDA is an approach to system development, which separates the views of a system on different abstraction levels or layers: *Computation Independent Model* (CIM), *Platform Independent Model* (PIM), and *Platform Specific Model* (PSM). In MDA, the models are primary artifacts in the software development process. This is accomplished through a series of transformations between models specified in the same abstraction level (horizontal transformations) or different abstraction levels (vertical transformations).

The proposed MDA-based method focuses on horizontal transformations between business process models at a PIM level. This method takes as input a CBP model and generates as output an integration process model of one organization, according to the role the organization fulfills in the CBP model. The integration process of the other organizations involved in the CBP are not generated so that each organization can focus on its own integration process model. The input model is defined with the UP-ColBPIP language, and the output model is defined with the BPMN language. These languages allow representing platform-independent process models at a high level of abstraction.

The transformation process consists in analyzing each element of a interaction protocol from the viewpoint of a selected protocol role, and generating public/private logic and activities in the target integration process model by applying transformation rules. The input of a rule is a protocol element. The output of a rule is a predefined BPMN pattern that expresses the semantics of a protocol element in terms of the elements and semantics provided by BPMN. For transforming business messages, BPMN patterns were defined according with the Activity Patterns in order to generate the activities required by an organization to support the exchange of business messages. For the other protocol elements, such as control flow segments, protocol reference and termination, BPMN patterns were also defined to generate the control flow and elements required by an integration process to be consistent with its corresponding CBP.

The transformation process of an interaction protocol into an integration process model takes into account the following actions:

1. The *lifeline* of the selected role of the source interaction protocol is analyzed and a BPMN diagram

is generated, which represents the integration process of the organization that performs such role in the CBP.

2. The BPMN diagram is built through the composition of the predefined BPMN patterns by applying model transformation rules.
3. For each protocol element there is a rule that transforms such element into the corresponding BPMN element/s in a BPMN diagram, according with an output BPMN pattern associated to the rule.

The generated integration process model contains: public activities that enable to an organization to send or receive business messages defined in an interaction protocol, private activities that enable the organization to process or generate the information to be exchanged, and the control flow that enables the organization to carry out the routing of business messages as it was defined in the interaction protocol.

Private activities are defined in a coarse-grained way, and therefore they may be refined by business analysts in order to define how they would be carried out in a particular organization. Public activities and the control flow must remain inalterable in order to preserve the consistency between the interaction protocol and the integration processes, and the interoperability among the integration processes of the organizations involved in the interaction protocol.

To define the transformation rules that generate the activities for the exchange of business messages, the correspondence between speech acts and activity patterns was obtained analyzing the semantics of both and identifying the similarity between them. The semantics of speech acts used by the UP-ColBPIP language is essential to identify activity patterns for generating the public and private activities of an integration process model. Process languages such as BPMN do not provide semantics to cross-organizational business messages. For this another reason the UP-ColBPIP language is used for modeling CBPs.

4.1 Transformation Rules for Business Messages

Each transformation rule has two output BPMN patterns, one to be applied in case of the generation of the integration process of the role that is the sender of the business message, and another one to be applied in case of the generation of the process of the role that is the receiver of the business message. The first pattern contains all the activities corresponding to the sender role of the activity pattern used in the rule. The

second pattern contains all activities corresponding to the receiver role of the activity pattern used.

All activities of the BPMN patterns of the rules are labeled with the *verb-object* style, which is considered less ambiguous and more useful by model users (Mendling et al., 2010).

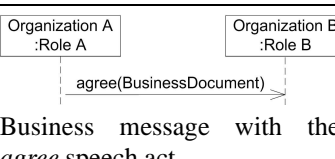
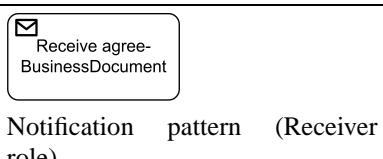
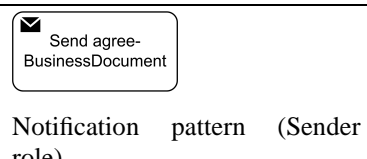
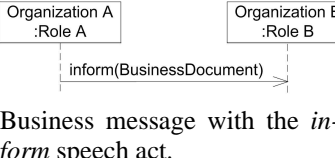
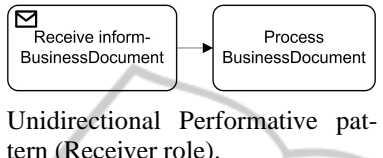
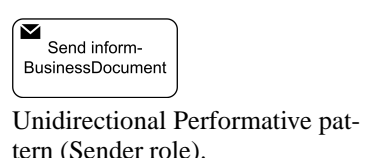
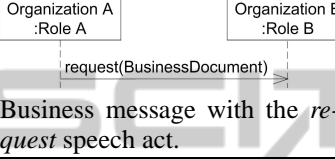
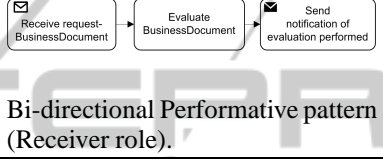
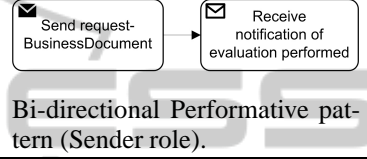
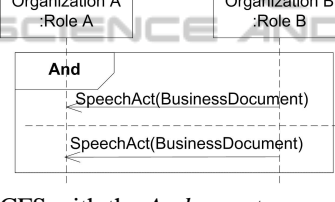
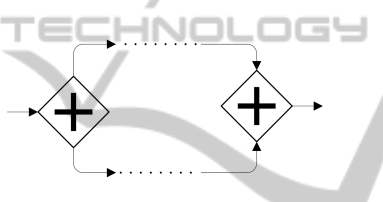
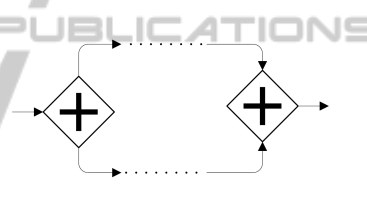
In a BPMN pattern, a public activity that enables an organization to send a business message is defined as a *Send Task*, which is a task type that represents the sending of a message to an external participant (OMG, 2010). A public activity that enables an organization to receive a business message is defined as a *Receive Task*, which is a task type that waits for a message to arrive from an external participant (OMG, 2010). Both *Send Task* and *Receive Task* types have an associated *Message*, which represents the content of a communication between two participants (OMG, 2010). A message in a BPMN pattern represents the business document (BD) exchanged in a business message of a protocol.

Private activities generated by the output of the rule are defined as *Abstract Task* type (OMG, 2010). Then, business analysts, who know the structure and operation of organizations, should indicate the behavior that these activities might represent: *Service*, *User*, *Manual*, *Script*, or *Business Rule*.

Following we describe each transformation rule for the business messages indicating the activity pattern used and providing the corresponding BPMN patterns to be used for the sender of the message and the receiver of the message. Table 2 shows graphically some transformation rules of business messages.

- Rule *Accept-Proposal*: is applied for each business message with the *accept-proposal* speech act. This is based on the Notification pattern, which allows the sender to notify the acceptance of a previously submitted proposal. The reason of the acceptance is communicated in the BD.
 - Sender: for the role sending the message, a *send task* is added and labeled as *Send + accept-proposal + BD name*.
 - Receiver: for the role receiving the message, a *receive task* is added and labeled as *Receive + accept-proposal + BD name*.
- Rule *Agree* (see Table 2.a): it is applied for each business message with the *agree* speech act. This is based on the Notification pattern, which allows the sender to notify the acceptance of a previously submitted request to perform some action. The reason of the acceptance is communicated in the BD.
 - Sender: a *send task* is added and labeled as *Send + agree + BD name*.

Table 2: BPMN patterns of transformation rules for some elements of the UP-ColBPIP language.

	Input patterns (UP-ColBPIP)	Output patterns (BPMN)	
		Receiver role	Sender role
a	 <p>Business message with the <i>agree</i> speech act.</p>	 <p>Notification pattern (Receiver role).</p>	 <p>Notification pattern (Sender role).</p>
b	 <p>Business message with the <i>inform</i> speech act.</p>	 <p>Unidirectional Performative pattern (Receiver role).</p>	 <p>Unidirectional Performative pattern (Sender role).</p>
c	 <p>Business message with the <i>request</i> speech act.</p>	 <p>Bi-directional Performative pattern (Receiver role).</p>	 <p>Bi-directional Performative pattern (Sender role).</p>
d	 <p>CFS with the <i>And</i> operator.</p>		

- Receiver: a *receive task* is added and labeled as *Receive + agree + BD name*.
- Rule *Call-for-Proposal*: it is applied for each business message with the *call-for-proposal* speech act. This is based on the Bi-directional Performative pattern, which allows the sender to request the receiver generates a proposal to perform some action. The sender waits until the receiver notifies him about completion of the requested activity.
 - Sender: two tasks are added; a *send task*, which is labeled as *Send + call-for-proposal + BD name*, and a *receive task* which is labeled as *Receive + notification of evaluation performed*.
 - Receiver: three tasks are added; a *receive task* which is labeled as *Receive + call-for-proposal + BD name*, an *abstract task* which is labeled as *evaluate + BD name*, and a *send task* which is labeled as *Send + notification of evaluation performed*.
- Rule *Inform* (see Table 2.b): it is applied for each business message with the *inform* speech act. This is based on the Unidirectional Performative pattern, which allows the sender to inform about a particular BD so that the receiver may have knowledge of the information contained in such BD.
 - Sender: a *send task* is added and is labeled as *Send + inform + BD name*.
 - Receiver: two tasks are added: a *receive task* which is labeled as *Receive + inform + BD name*, and an *abstract task* which is labeled as *evaluate + BD name*.
- Rule *Propose*: it is applied for each business message with the *propose* speech act. This is based on the Bi-directional Performative pattern, which allows the sender to propose the receiver to perform a certain action. The sender waits until the receiver notifies him about completion of the requested activity.
 - Sender: two tasks are added; a *send task* which is labeled as *Send + propose + BD name*, and a *receive task* which is labeled as *Receive + notification of evaluation performed*.
 - Receiver: three tasks are added; a *receive task* which is labeled as *Receive + propose + BD name*, an *abstract task* which is labeled as *evaluate + BD name*, and a *send task* which is la-

beled as *Send + notification of evaluation performed*.

- Rule *Refuse*: it is applied for each business message with the *refuse* speech act. This is based on the Notification pattern, which allows the sender to notify the refusal of the previously submitted request to perform some action. The reason for the refusal is communicated in the BD.
 - Sender: a *send task* is added and labeled as *Send + refuse + BD name*.
 - Receiver: a *receive task* is added and labeled as *Receive + refuse + BD name*.
- Rule *Reject-Proposal*: it is applied for each business message with the *reject-proposal* speech act. This is based on the Notification pattern, which allows the sender to notify the rejection of the previously submitted proposal. The reason for the rejection is communicated in the BD.
 - Sender: a *send task* is added and labeled as *Send + reject-proposal + BD name*.
 - Receiver: a *receive task* is added and labeled as *Receive + reject-proposal + BD name*.
- Rule *Request* (see Table 2.c): it is applied for each business message with the *request* speech act. This is based on the Bi-directional Performative pattern, which allows the sender to request the receiver to perform some action. The sender waits until the receiver notifies him that the requested activity has been performed.
 - Sender: two tasks are added; a *send task* which is labeled as *Send + request + BD name*, and a *receive task* which is labeled as *Receive + notification of evaluation performed*.
 - Receiver: three tasks are added; a *receive task* which is labeled as *Receive + request + BD name*, an *abstract task* which is labeled as *evaluate + BD name*, and a *send task* which is labeled as *Send + notification of evaluation performed*.

4.2 Transformation Rules for the Remaining Protocol Elements

To carry out the transformation of the remaining protocol elements, such as control flow segments (CFS), protocol reference and termination event, we propose a set of predefined BPMN patterns that represents the semantics of a protocol element in terms of the elements and semantics provided by BPMN language from the viewpoint of a role. Due to UP-ColBPIP is a

block-structured language, BPMN patterns for transforming CFSs are defined in a structured way. Table 2 shows graphically some transformation rules.

- Rule *Start*: a *None Start Event* is added to represent the start of an integration process.
- Rule *And* (see Table 2.d): a CFS *And* is mapped into a *Parallel Gateway* with a *gate* for each interaction path. A *synchronizing Parallel Gateway* is added to synchronize each path. Paths that have a termination event are not synchronized.
- Rule *Xor*: A CFS *Xor* (either data-based or event-based) is mapped into an *Event-Based Gateway* if the role receives messages, or it is mapped into an *Exclusive Gateway* if the role sends messages in the interaction paths. One *gate* per interaction path is added. A *converging Exclusive Gateway* is added to merge each path. Paths that have a termination event are not merged.
- Rule *Or*: a CFS *Or* is mapped into an *Inclusive Gateway* with a *gate* for each interaction path. A *converging Inclusive Gateway* is added to synchronize each path. Paths that have a termination event are not synchronized.
- Rule *Loop*: for each CFS with the *Loop* operator, an *expanded Sub-Process* with a *Loop Marker* is created. The transformation depends on the Loop type. (1) For a "while loop" whose condition is $[(0, n), Var1=True]$, the *LoopCondition* attribute with the *var1=True* value, the *loopMaximum* attribute with the *n* value, and the *testBefore* attribute with the *true* value are settled in the expanded Sub-Process. (2) For a "repeat until loop" whose condition is $[(1, n), Var1=True]$, the *LoopCondition* attribute with the *not var1* value, the *loopMaximum* attribute with the *n* value, and the *testBefore* attribute with the *false* value are settled in the expanded Sub-Process.
- Rule *Cancel*: a CFS *Cancel* is mapped into an *expanded Sub-Process*. This Sub-Process is triggered by a *Timer Intermediate Event* if the interaction path of the CFS handles a time constraint, or by a *Conditional Intermediate Event* for exceptions related to the protocol logic. The *outgoing Sequence Flow* of this Sub-Process is connected to a *Terminate End Event*.
- Rule *Ref*: for each *protocol reference*, a *Call Activity* is created to refer to a process defined in another BPMN diagram. The name of the *Call Activity* is the same as the protocol it refers to.
- Rule *Explicit-End*: for each *termination event* in a protocol, a *Terminate End Event* labeled *Success* or *Failure* is added to the BPMN diagram. If this

event is in an *expanded Sub-Process*, it is modeled by a *Signal End Event*. Then, a *Signal Intermediate Event* is attached to the Sub-Process to catch the signal. The *outgoing Sequence Flow* of this event is connected to a *Terminate End Event*. This ensures the protocol execution ends when the Sub-Process returns the control to the main process.

- Rule *Implicit-End*: A *None End Event* is added to BPMN diagram to represent the implicit termination of a protocol.

4.3 Eclipse-based IDE to Support the MDA-based Method

In order to implement the proposed MDA-based method we use the extended IDE proposed in (Lazarte et al., 2010) to support the model-driven development methodology for B2B collaborations. This IDE is based on the Eclipse open development platform (Eclipse, 2004) in order to take advantage of a well-known development environment and the Eclipse platform extension mechanisms. Figure 3 shows a view of the architecture of the Eclipse-based IDE for supporting the proposed MDA-based method.

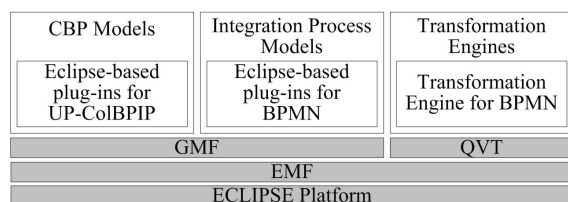


Figure 3: Architecture of the Eclipse-based IDE.

The transformation engine for BPMN interprets and executes the transformation definition, which uses a UP-ColBPIP meta-model as the source meta-model and a BPMN meta-model as the target meta-model. Both meta-models are implemented with EMF (Eclipse, 2004).

Transformation rules required for the transformation definition and described in sections 4 are specified with QVT (OMG, 2011), which is a standard language for model transformations. These rules are implemented with the QVT Operational component (Eclipse, 2004) of Eclipse.

5 A CASE STUDY EXAMPLE

The *Collaborative Demand Forecast* interaction protocol described in section 2 is used for exempli-

fying the model transformation process aforementioned. Due to space limitations, the model transformation process is described only from the viewpoint of the supplier, and therefore all business message are analyzed from this viewpoint.

Figure 4 shows the BPMN diagram of the supplier's integration process model generated by the transformation process. The applied transformation rules are indicated in the diagram.

A *None Start Event* was added to represent the start of the integration process (rule *Start*). The first protocol element is the *request(ForecastRequest)* business message, which is received by the supplier. The *request* message is transformed by applying the rule *Request*. This rule adds the activities that enables the supplier to receive and evaluate the request, and then send a notification of the evaluation performed.

The second protocol element is the CFS with the *Xor* operator with two alternative paths. This element is transformed by applying the rule *Xor*. This rule adds an *Exclusive Gateway* with two *gates*, one for each interaction path. Then, each *path* is analyzed to determine the rule to be used in the transformation.

After the reception of a request, the supplier has to respond to the customer with an agreement or refusal throughout the corresponding business message: *agree(ForecastRequestResponse)* and *refuse(ForecastRequestResponse)*, respectively. The *agree* message is transformed by applying the rule *Agree* to enable the supplier to notify to the customer the reason of such decision. The rule *Refuse* is applied to notify the refusal. In this case, after the *refuse* message is sent, the process finalizes as it is indicated by the explicit *termination* element. This element is transformed by the rule *Explicit-End*, which adds a *Terminate End Event* labeled *Failure*. Because one path has an explicit *termination*, the two gates are not synchronized and the transformation continues along the path which does not have the explicit termination.

The next protocol element is the CFS with the *And* operator, which represents the parallel execution of two paths. This element is transformed by the rule *And*, which is mapped into a *Parallel Gateway* with a gate for each interaction path.

The first path contains the *inform(POSForecast)* business message that is received by the supplier. The *inform* message is transformed by the rule *Inform* that generates the tasks that enable the supplier to receive the business message and then to process the information received within the document from the customer. The second path contains the *inform(PlannedEvents)* business message also received by the supplier. This message is also transformed by applying the rule *Inform*. Both paths are synchronized by a synchronizing

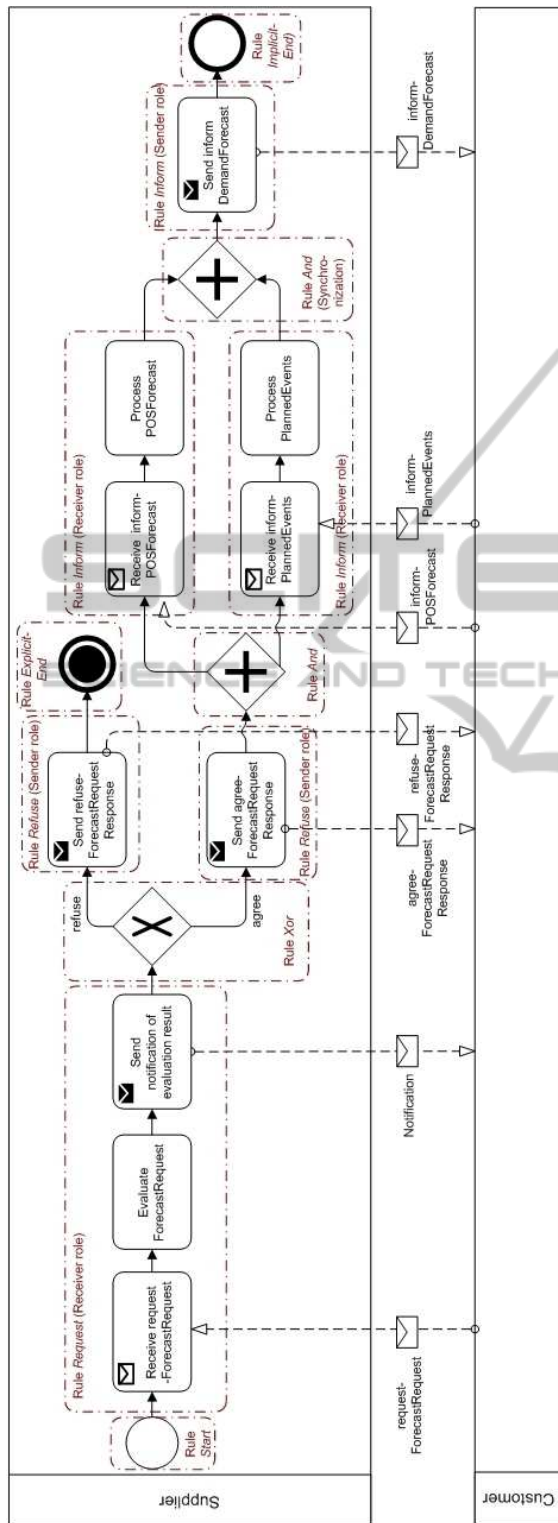


Figure 4: Supplier's integration process model.

Parallel Gateway (see rule And).

Following, the *inform(DemandForecast)* message sent by the supplier is transformed by the rule *Inform*, which generates the task that enables the supplier to send the *DemandForecast* document so that the customer can have knowledge about this document. Finally, the implicit termination of the protocol is modeled with a *None End Event* (rule *Implicit-End*).

6 RELATED WORK

The modeling of CBPs has been a subject of intensive research. However, this did not occur with the modeling of the business processes required to execute CBPs. In (Hofreiter, 2009), an approach for modeling local choreographies is proposed. This approach provides a UML Profile which extends the UN/CEFACT Modeling Methodology (UMM) and enables organizations to model their local choreography in alignment with the global choreography defined in UMM models. However, it is not based on a model-driven approach and does not provide mechanisms to identify private activities that should be added to a local choreography. Instead, in the present work we propose an MDA-based approach that uses speech acts to identify and automatically generate these activities. Also, we use the standard BPMN language instead of a particular one.

In (Zaha et al., 2006), an algorithm for generating service local models from a service choreography model is proposed in order to provide a local view of each organization that participates in the choreography. Such work focuses on process models based on service interactions. Instead, we focus on independent-platform process models.

In (Gschwind et al., 2008), an extension of a business process modeling tool with basic workflow patterns is proposed. This modeling tool provides to the users the support to select patterns that are applicable in some user-determined context. However, it is not applied in the context of B2B collaborations and only focuses on the basic workflow patterns for control flow. Instead, in this work we provide support for modeling business process in a B2B context. In addition, do not only use control flow patterns but also use activity patterns, which are suited for defining business processes from a variety of application domains.

7 CONCLUSIONS AND FUTURE WORK

In this work we have proposed an MDA-based method for the automatic generation of the integration process model of an organization from a CBP model. This method allows organizations to generate integration processes which are interoperable as well as consistent with the global logic agreed on the CBPs. Consistence is guaranteed since the integration process models of the organizations are derived from CBP models by applying a top-down model-to-model transformation process. In addition, interoperability is achieved by applying activity patterns that guarantee the synchronization between the generated public activities of the integration process models.

The method enables to transform CBP models defined with the UP-ColBPIP language. This language uses interaction protocols to represent the behavior of technology-independent CBPs. The use of interaction protocols supports the main features of B2B collaborations: global view of the B2B interactions, organization autonomy, decentralized management, peer-to-peer interactions and representation of complex negotiations. Also, the use of speech acts in interaction protocols provides semantics to business messages defined in CBP models. This semantics enables the common understanding of the meaning of B2B interactions of a CBP model. Furthermore, this semantics enables the identification and application of the activity patterns for generating the public and private activities of integration process models.

The method allows generating integration process models defined with the BPMN language. This provides process models defined at a high level of abstraction and in a platform-independent way. This also enables the communication among stakeholders and their implementation in different platforms. Thus, organizations can understand and focus on the business requirements to fulfill the role they perform in CBPs.

Activity patterns are used in the proposed transformation process to generate the public and private activities required in the integration processes to support cross-organizational message exchanges. The use of activity patterns brings several benefits for designing integration process models: automate and facilitate the design of process models, reduce process modeling time and cost, improve process model quality, and enable the reuse of the process knowledge captured in them to generate the public and private activities. Also, the use of activity patterns ensures the interoperability in the message exchange between integration processes by providing a synchronization among the

receiving activities and the sending activities generated in the processes.

Finally, the proposed MDA-based method shows that a direct mapping can be applied to generate a BPMN diagram of the integration processes of an organization from a CBP model represented as interaction protocol. No intervention is required by a modeler. For each protocol element, a BPMN pattern is provided to generate its behavior from the viewpoint of the role an organization performs in the protocol.

Future work is about the identification of process model fragments that enable the business analysts to refine private activities in order to obtain fine-grained models.

REFERENCES

- Bauer, B., Roser, S., and Müller, J. (2005). Adaptive design of cross-organizational business processes using a model-driven architecture. In *Wirtschaftsinformatik 2005*, pages 103–121. Springer.
- Eclipse (2004). Eclipse Platform. <http://www.eclipse.org>.
- Eriksson, H.-E. and Penker, M. (2000). *Business Modeling With UML: Business Patterns at Work*. John Wiley & Sons, Inc.
- FIPA (2002). Communicative Act Library Specification. <http://www.fipa.org/specs/fipa00037/SC00037J.html>.
- Gschwind, T., Koehler, J., and Wong, J. (2008). Applying patterns during business process modeling. In *Proc. of BPM*, volume 5240 of *LNCS*, pages 4–19. Springer.
- Hofreiter, B. (2009). Extending un/cefact's modeling methodology by a uml profile for local choreographies. *Inf Syst E-Bus Manage*, 7(2):251–271.
- Lau, J., Iochpe, C., Thom, L., and Reichert, M. (2009). Discovery and Analysis of Activity Pattern Cooccurrences in Business Process Models. *Proc. of ICEIS*, pages 83–88.
- Lazarte, I. M., Tello-Leal, E., Roa, J., Chiotti, O., and Villarreal, P. D. (2010). Model-Driven Development Methodology for B2B Collaborations. In *Proc. of EDOCW*, pages 69–78. IEEE Computer Society.
- Mendling, J., Reijers, H., and Recker, J. (2010). Activity labeling in process modeling: empirical insights and recommendations. *Inform. Syst.*, 35(4):467–482.
- OMG (2003). Model Driven Architecture Guide, V.1.0.1. <http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf>.
- OMG (2010). Business Process Model and Notation, V. 2.0. <http://www.omg.org/spec/BPMN/2.0/Beta2>.
- OMG (2011). Query/View/Transformation Specification, V.1.1. <http://www.omg.org/spec/QVT/1.1>.
- Roser, S. and Bauer, B. (2005). A categorization of collaborative business process modeling techniques. In *Proc. of CECW*, pages 43–51. IEEE Computer Society.

- Thom, L., Reichert, M., and Iochpe, C. (2009). Activity patterns in process-aware information systems: basic concepts and empirical evidence. In *IJBPM*, volume 4, pages 93–110. Inderscience.
- Villarreal, P. D., Lazarte, I. M., Roa, J., and Chiotti, O. (2010). A Modeling Approach for Collaborative Business Processes Based on the UP-ColBPIP Language. In *BPM Workshops*, volume 43 of *LNBIP*, pages 318–329. Springer.
- Villarreal, P. D., Salomone, E., and Chiotti, O. (2007). *Enterprise modeling and computing with UML*, chapter Modeling and specifications of collaborative business processes using a MDA approach and a UML profile, pages 13–45. Idea Group Inc.
- Weske, M. (2007). *Business Process Management: Concepts, Languages, Architectures*. Springer-Verlag.
- Zaha, J. M., Dumas, M., ter Hofstede, A., Barros, A., and Decker, G. (2006). Service Interaction Modeling: Bridging Global and Local Views. In *Proc. of EDOC*, pages 45–55. IEEE Computer Society.

