From a UMM Business Process Model to a Business Environment Specific ebXML Process

Birgit Hofreiter
Department of Distributed and Multimedia Systems
University of Vienna
birgit.hofreiter@univie.ac.at

Christian Huemer
Department of Distributed and Multimedia Systems
University of Vienna
christian.huemer@univie.ac.at

ABSTRACT

UN/CEFACT's Modeling Methodology (UMM) is used to design inter-organizational business process models in e-business. Developing a new model for each small variation in different business environments would lead to a multitude of similar models. We extend UMM by a constraint mechanism for adding business environment specific constraints to models. Thus, multiple business environments may share the same model. The implementation of business process models using ebXML or Web Services is supported by choreography languages like BPSS or BPEL. These choreography languages are usually XML-based in order to be processed by a business partner's application. Since this business application runs in the particular business environment of the business partner, BPSS and BPEL processes are always specific to the business environment. Consequently, this paper elaborates on the mapping of a standard UMM process model defined for multiple business environments to business environment specific BPSS processes.

Key words: Inter-organizational Business Processes, Business Process Modeling, e-Business Standards, ebXML

1. Motivation

For a long time activities in the area of inter-organizational systems followed the pure data centric EDI (electronic data interchange) approach [Chau 2001]. Most attention was spent on business document types, messaging formats, and application interface descriptions. For any system—be it an IT system implemented in the scope of a single organization or be it an inter-organizational B2B system—the dynamic aspects are as important as the structural definitions. The development of any good system of a certain size requires a well defined analysis and design process. For the specific focus of developing B2B systems, the United Nation's Centre for Trade Facilitation and Electronic Business (UN/CEFACT) has developed the UN/CEFACT modeling methodology (UMM) [UN/CEFACT 2003a, UN/CEFACT 2003b] for analyzing both dynamic and structural B2B aspects. In this paper we built up on UMM in order to define inter-organizational standard business process models.

Following the idea of a model-driven architecture (MDA), the platform-independent UMM models must be mapped to platform-specific artefacts. In this paper we concentrate on the mapping of the dynamic aspects of a UMM model to platform-specific artefacts. It is envisioned that software tools will be able to process the defined choreography and execute the business process accordingly. Thus, the choreography must be defined in a machine-readable format. Recently, there are standardization efforts toward choreography languages. The most prominent examples include the Business Process Execution Language (BPEL) [Andrews et al. 2003] and the ebXML Business Process Specification Schema (BPSS) [UN/CEFACT 2003c]. Since all of them are XML-based, software tools are able to process them and subsequently track the B2B exchanges. The challenge is to automatically derive business process choreographies in these choreography languages from the platform independent UMM models. This paper contributes to this task by specifying mappings from the UMM to ebXML BPSS.

Another central point is the scope of UMM models. UMM demands unambiguous specifications of inter-organizational processes called business collaboration models. It is rather easy to develop a model for a very specific business environment. This is how the steps are currently defined in the UMM [UN/CEFACT 2003b]. However, the scope of UMM should be developing unambiguous business process models for global e-business. In practice, one and the same business process varies a little bit with respect to the business environment. Developing a new model for each variation will result in a multitude of models. Thus, a generic model together with constraints for different business environments is a much more effective approach to ensure unambiguousness. We extend UMM in order to

capture business environment-specific constraints. Thereby, we conceptually make use of the context driver concept as identified by ebXML core components [UN/CEFACT 2003d].

We had to choose a certain syntax for our context driver-based language. The preferred language for specifying constraints in UML is the Object Constraint Language (OCL) [OMG 2003]. Since UMM is based on UML it seemed to be straight forward to specify constraints in OCL. It was our goal to give guidance on using OCL in UMM. However, UMM puts UML into a very small corset that requires special kinds of constrains. Since OCL does not focus on constraints on activity graphs and does not mention access to tagged values, we had to specify some necessary extensions to OCL. Therefore, the developed templates for business context constraints, which are defined in an extended Bachus-Naur-Form (BNF), are not 100% OCL compliant, but follow the spirit of OCL.

In contrary to UMM, choreography languages like BPEL or BPSS define a business process always in its very specific business environment. This is due to the fact, that the intended consumer of a choreography language is a software tool or a module of the business application governing and/or executing the B2B process. Since a business application is always executed in the specific business environment in which the business partner operates, the choreography is specified exactly for this business environment. Therefore, a mapping from UMM to BPSS must always result in a context-specific choreography.

It follows that our proposed extension to UMM results in a gap between UMM models valid for multiple business environment and their business environment specific equivalents in a business choreography language. Consequently, it is required to map a single multi-environment UMM model to multiple business choreographies—one for each business environment. For example, a purchase order management process for books might result in a slightly different BPSS choreography than one for tourism products, although both may be based on the same UMM purchase order management model. In this paper we present a mapping that evaluates the business context in the UMM business constraints before creating the BPSS choreography file.

2. Related Work

The idea of defining business processes crossing organizational boundaries goes back to ISO's Open-edi reference model [ISO 1997]. A first implementation of the choreography aspects of this model was a Petri-Net approach contributed by Lee [Lee 2000]. Also other authors used Petri-Nets to define the workflow between organizations [Lenz and Oberweis 2003, van der Aalst 1999]. In addition to the workflow approaches, collaborative processes were also considered from a business transaction perspective. Web Services are considered as an enabler to implement long-running business transactions in a distributed environment of different business partners [Little 2003, Papazoglou 2003]. Furthermore, Web Services have a potential to serve as an enabler for e-business applications [Chen et al. 2003a, Chen and Meixell 2003b]. A lot of different standard languages describing an orchestration or a choreography of a process have been developed, e.g. the Business Process Modeling Language (BPML) [Arkin 2002], the Business Process Execution Language (BPEL) [Andrews et al. 2003], and the Web Services Choreography Interface (WSCI) [W3C 2002]. An orchestration describes the order in which a composite Web Service invokes other Web Services to realize its function. A choreography describes to the outside world in which order communication with Web Services is expected to fulfill an overall function (c.f. [Peltz 2003]). The specification of collaborative business processes focuses on a complementary choreography between business partners. BPEL is not only the most commonly supported orchestration language, but also supports an abstract version for choreographies. Hence, BPEL seems to be the right choice [Leymann et al. 2002]. However, BPEL always describes a choreography from the viewpoint of a single partner. This means it is not possible to describe a single choreography for the overall collaboration. To find a business partner in a registry one cannot refer to a common process, but must perform a complicated match-making of the private process interfaces [W3C 2002]. As a consequence, the first draft of the Web Services Choreography Description Language (WS-CDL) [Kavantzas et al. 2004] has been developed to specify a Web Services choreography from a neutral perspective. Within the ebXML framework which is another candidate for a platform solution, the business process specification schema (BPSS) always describes the choreography of a business collaboration from an overall perspective.

Since choreography languages and Web Services are expressed in XML, there have been attempts to model them in a graphical syntax. For this purpose BPMI is developing the Business Process Modeling Notation (BPMN) [White 2004]. This notation presents the amalgamation of best practices in the business process modeling community. Other approaches used UML to visualize Web Services and their choreography [Mantell 2003, Provost 2003, Thoene et al. 2002]. More advanced approaches provide a development process for inter-organizational business processes. These are either driven by existing private workflows [Jung et al. 2004] or they are driven by the inter-organizational requirements instead of the private ones [Kramler et al. 2005]. The latter approach is also used in the development of RosettaNet Partner Interface Processes (PIPs) [RosettaNet 2002] and UN/CEFACT's modeling methodology (UMM) [UN/CEFACT 2003a, UN/CEFACT 2003b].

3. A Guide to UMM and BPSS

3.1. UMM and BPSS Overview

UMM concentrates on the business semantics of a B2B partnership. The goal of UMM is capturing the commitments made by business partners. These commitments are manifested in the resulting choreography of the business process. UMM is a methodology for defining business aspects of a B2B collaboration that is based on UML. It should be noted that most of the terminology used in this paper refers to the definitions in the UMM methodology. The methodology is based on the UMM meta model [UN/CEFACT 2003a] that defines a coherent set of stereotypes, constraints, and tag definitions, i.e. a UML profile for the purpose of modeling inter-organizational business processes called business collaborations.

The UMM procedure consists of 4 views. Each view delivers a set of patterns and is based on its own well-formed meta-model that defines the syntax and semantics for each view. The ***business domain view (BDV)*** is used to gather existing knowledge. It identifies the business processes in the domain of the business problems that are important to stakeholders. It is important at this stage that business processes are not constructed, but discovered. The discovered processes may indicate a chance for a collaboration between business partners. The identification of possible business collaborations is part of the ***business requirements view (BRV)***. The goal of the business requirements view is harmonizing the requirements of different stakeholders. This results in a requirements specification for a business collaboration shared amongst the stakeholders.

The ***business transaction view (BTV)*** represents the view of the business process analyst who transforms the requirements into an analysis model. The analysis model defines both the choreography of information exchanges and a data model for each information exchange. This data model is defined by a class diagram. In order to describe the choreography of the inter-organizational process UMM uses to special kinds of activity graphs. The activity graph of a *business transaction* models an exchange from one business partner to the other with an optional response. Afterwards the business applications must always be in synch. The activity graph of a *business collaboration protocol* defines a choreography amongst the business transactions. Since these two types of activity graphs present the source of our mapping process we further detail them in the next two sub-sections. The optional ***business service view (BSV)*** transforms the artifacts of the previous view to show message exchanges between network components.

The ebXML standard to describe business processes is the ***business process specification schema (BPSS)***. ebXML does not require any specific modeling language or modeling methodology, but it recommends UMM in order to develop business process definitions. BPSS provides an XML schema to specify a collaboration between business partners. It provides configuration parameters for the partners' runtime systems in order to execute this collaboration between a set of e-business software components.

The work on BPSS was based on the UMM meta model. It concentrated on those modeling elements necessary for a business process execution and excluded the rest. Those UMM artifacts relevant to BPSS are the activity graphs of *business transactions* and of *business collaboration protocols*. The mapping of these artifacts to BPSS is described in the following sub-sections. We do not care about any other UMM artifacts. The reader interested in UMM is referred to [UN/CEFACT 2003b].

Foreseeing our requirements of Section 4—which is a business collaboration executed slightly differently in two different business environments—we use the case study of a simple purchase order management. Later it will be used to order books as well as tourism products like hotels, flights, etc. Of course considering all real world requirements would lead to a rather complex business collaboration model which goes far beyond the scope of this paper. Hence, we choose a limited functionality that still allows understanding the basic concepts of UMM and BPSS.

3.2. Business Transaction

In UMM *business transaction* is the basic building block to define a choreography between business partners appointed to authorized roles. If a business partner recognizes an event that changes the state of a business object, it initiates a *business transaction* to synchronize with the collaborating business partner. It follows that a *business transaction* is an atomic unit that leads to a synchronized state in both information systems. We distinguish one way and two-way business transactions: In the former case, the initiating business partner reports an already effective and irreversible state change that the reacting business partner has to accept. Examples are the notification of shipment or the update of a product in a catalog. It is a one way business transaction, because business information (not including business signals for acknowledgments) flows only from the initiating to the reacting business partner. In the other case, the initiating partner sets the business object(s) into an interim state and the final state is decided by the reacting business partner. Examples include request for registration, search for products, etc. It is a two way transaction, because business information flows from the initiator to the responder to set the interim state and backwards to set the final and irreversible state change. In a business context irreversible means that returning to an

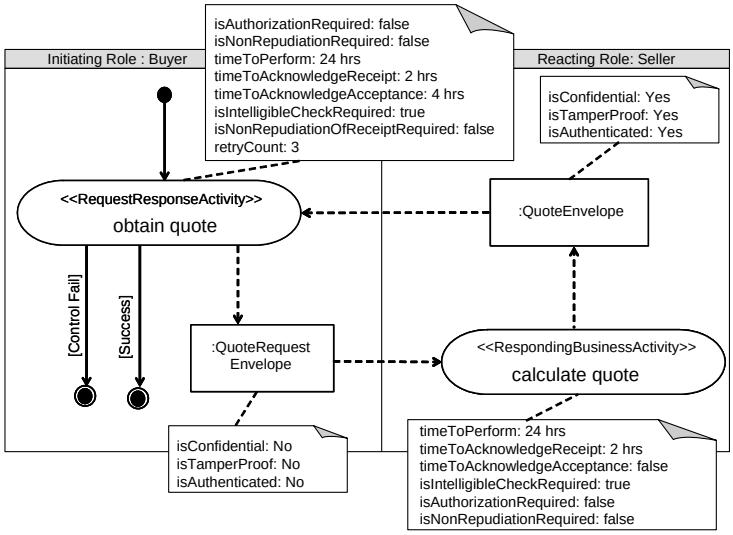

Figure 1. Business Transaction: Request for Quote

original state requires another business transaction. E.g., once a purchase order is agreed upon in a business transaction a rollback is not allowed anymore, but requires the execution of a cancel order business transaction.

In UMM a *business transaction* is defined by an activity graph that follows a certain pattern. Figure 1 depicts the *request for quote* business transaction which follows the typical pattern of a two way transaction. We assume the *request for quote* business transaction is used in the order management of books as well as in that of tourism product. The basic flow described below is the same in both cases. However, they differ in the details of timing and security. The variations on timing and security are further discussed in section 4.1.

The activity graph uses always two swimlanes, one for the initiating partner and one for the reacting partner. Each business partner performs exactly one activity that is assigned to the respective swimlane. The transaction starts with the requesting business activity *obtain quote* which outputs a first envelope (*quote request envelope*), but does not necessarily end afterwards. The envelope is input to the responding business activity *calculate quote* which is triggered by its receipt. In case of a two way transaction the requesting activity *obtain quote* must still be alive to receive an envelope (*quote envelope*) produced by the responding business activity. The requirement of synchronized states—which is realized by both business information exchanges and business signals for acks—allows the initiator to determine the final state of the business object(s). Therefore, the completion of the requesting business activity results in a deterministic end state.

UMM adopts the six types of business transactions identified by RosettaNet [RosettaNet 2002]: *information distribution*, *notification*, *query/response*, *request/confirm*, *request/response*, and *commercial transaction*. The stereotype of the requesting activity is named after the underlying transaction type. The different types of transactions also differ in their defaults for the tagged values characterizing the requesting and responding activity. The self-explaining tags are *time to perform*, *time to acknowledge receipt*, *time to acknowledge acceptance*, *is authorization required*, *is intelligible check required* and *is non-repudiation required*. In addition, *is non-repudiation of receipt required* and *retry Count* characterize requesting activities. It follows that acks are not modeled by an object flow in UMM. Instead, time values in the tagged values define the required flow of business signals for acks. An ack of receipt is sent after schema and sequence validation and an ack of acceptance after validating additional business rules.

The business information covered in an envelope is modeled by a class diagram. We do not go into the details of these class diagrams in this paper. However, security parameters apply to the envelope. These self-explaining security parameters are *is confidential*, *is tamper proof*, and *is authenticated*.

Like UMM, BPSS supports the concept of a business transaction. Before going into the details of BPSS, it should be mentioned that all BPSS elements include an attribute for a unique id (*nameID*) and another one for a human readable name (*name*) for the business element defined. The unique id might be used for reference purposes from other elements. To enable the reader to easily trace these references, we manipulate the *nameID* attribute in all examples to show an id that is “reader-friendly”.

In addition to the two standard attributes, a business transaction includes a context-dependent attribute that defines the type of the business transaction (*pattern*). Valid instances correspond to the six types used in UMM.

Furthermore, the attribute is *guaranteed delivery required* signals that partners must employ a delivery channel that provides a delivery guarantee. Since UMM always assumes the existence of such a channel, a transformation will result in a positive value.

The two child elements of business transaction are *requesting business activity* and *responding business activity*. The attributes correspond more or less to the context-sensitive tagged values of the corresponding UMM activity. However, an ebXML service interface does not care about the execution time of an intra-organizational activity. Thus, the tagged value *time to perform* of both requesting and responding business activity cannot be mapped to BPSS. Furthermore, *time to acknowledge receipt* and *time to acknowledge acceptance* are only specified if the corresponding ack is required, and are omitted otherwise.

The flow of business information is defined in BPSS by defining the envelope (*document envelope*) as child element of that activity that outputs this envelope. Owing to the strict pattern of business transactions, it is clear that the other activity receives the envelope as input. It follows that the *requesting business activity* always includes a child document envelope—the *responding business activity* only in case of a two way transaction. A *document envelope* element includes attributes for all context-sensitive security parameters identified in UMM (with a slightly different naming in case of *is tamper detectable*). BPSS further distinguishes whether these security requirements apply only to the transport (*transient*) and/or the storage after the envelope was delivered (*persistent*). Thus, a UMM value of *true* for one of these security flags equals to *transient* in BPSS. Additionally, the attributes *business document* and *business document IDREF* reference a *business document* element that is covered by the envelope. A mapping of the business transaction *request for quote* depicted in Figure 1 results in the following BPSS code:

```
[1]  <BusinessTransaction
[2]    nameID="BT1"   name="request for quote"
[3]    pattern="RequestResponse" isGuaranteedDeliveryRequired="true">
[4]    <RequestingBusinessActivity
[5]      nameID="RBA1" name="obtain quote"
[6]      isAuthorizationRequired="false" timeToAcknowledgeReceipt="PT2H"
[7]      timeToAcknowledgeAcceptance="PT4H" isNonRepudiationRequired="false"
[8]      isNonRepudiationReceiptRequired="false" isIntelligibleCheckRequired="true" retryCount="3" >
[9]      <DocumentEnvelope
[10]       nameID="DE1" name="QuoteRequestEnvelope"
[11]       businessDocument="QuoteRequest" businessDocumentIDREF="D1"
[12]       isAuthenticated="none" isConfidential="none"
[13]       isTamperDetectable="none"/>
[14]   </RequestingBusinessActivity>
[15]   <RespondingBusinessActivity
[16]     nameID="RBA2" name="calculate quote"
[17]     isAuthorizationRequired="false" timeToAcknowledgeReceipt="PT2H"
[18]     isNonRepudiationRequired="false" isIntelligibleCheckRequired="true">
[19]     <DocumentEnvelope
[20]       nameID="DE2" name="QuoteEnvelope"
[21]       businessDocument="Quote" businessDocumentIDREF="D2"
[22]       isAuthenticated="transient" isConfidential="transient"
[23]       isTamperDetectable="transient"/>
[24]     </RespondingBusinessActivity>
[25]</BusinessTransaction>
```

3.3. Business Collaboration Protocol / Binary Collaboration

In UMM an inter-organizational business process is called *business collaboration*. Its choreography is defined by an activity graph called *business collaboration protocol*. Figure 2 shows an extract of a *business collaboration protoco*l. In practice, the *business collaboration protocol* describes collaborations between two partners only - although it is not limited to. In UMM each activity of a *business collaboration protocol* is stereotyped as *business transaction activity*. A *business transaction activity* is refined by a *business transaction* activity graph. A recursive nesting of *business collaboration protocols* is not possible. A *business transaction activity* has a maximum *time to perform* property. If it is not finished by this time, the initiating partner must send a failure notice. The *is concurrent* property signals the eventuality of a concurrent execution.

The transition from one *business transaction activity* to another requires (1) the completion of the first business transaction and (2) the availability of one or more business object in certain state(s). Furthermore, these business states are reflected in the pre- and post-conditions of a business transaction activity.

Our purchase order management example in Figure 2 involves three business transaction activities: *request for quote*, *reserve product*, and *order product*. All of them may be concurrent, but differ in the maximum execution time (24hrs, 12hrs, and 48 hrs). The *request for quote* activity is refined by the business transaction depicted in Figure 1. Similarly, each of the other two activities is refined by an activity graph of a business transaction. A

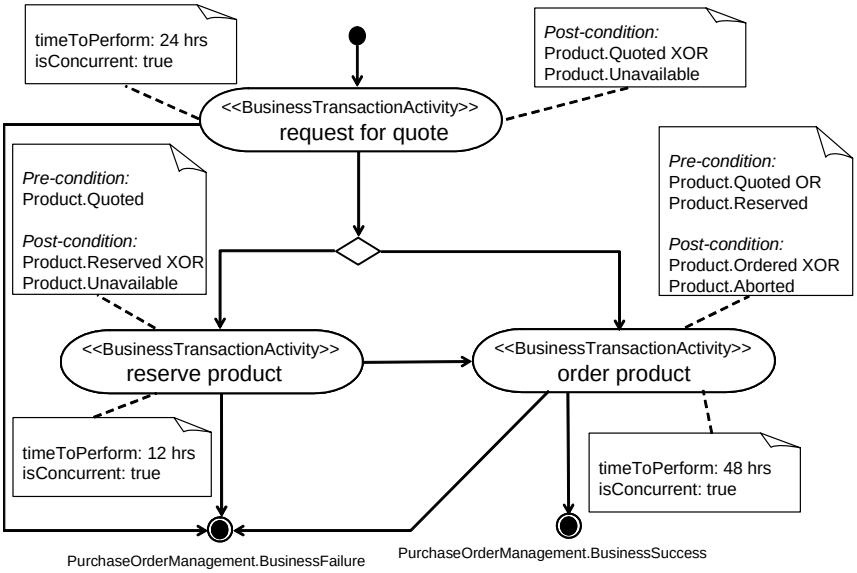

Figure 2. Business Collaboration Protocol: Purchase Order Management

request for quote does not require any pre-conditions to be met. It sets a *product* either into the state *quoted* or into state *unavailable*. If the product is in state *quoted*, the pre-condition of *reserve product* and of *order product* is fulfilled. Thus, either one of these business transaction activities may be performed next. The *reserve product* activity sets the *product* either in state *reserved* or to *unavailable*. Since a product may be reserved prior to ordering, the pre-condition is satisfied if *product* is in state *quoted* or *reserved*. The execution of *order product* leads either to a successful end by setting *product* into state *ordered* or it fails by setting *product* into state *aborted*.

BPSS 1.10 specifies only collaborations between two partners. These binary collaborations might be defined recursively. Since UMM business collaboration protocols do not yet use this concept, it is not considered in our transformation. A *binary collaboration* element includes the following sequence of child elements: *role*, *start*, *business transaction activity*, *success*, *failure*, *transition*, *fork*, *join*, and *decision*. The two role elements specify the two collaborating business partners. A state is either a *business transaction activity* element, or one of the pseudo states *fork*, *join*, and *decision*.

A *business transaction activity* element includes attributes (*business transaction*, *business transaction IDREF*) to reference the underlying business transaction. The *from role* attribute references the initiating role and the *to role* the reacting one. The tagged value equivalents *is concurrent* and *time to perform*, as well as pre-condition and post-condition are subject to business context variations. Furthermore, the attributes *begins when* and *ends when* contain some text that is recorded in UMM use case descriptions on which we do not concentrate here. Decision elements include a sub-element condition expression that may state an OCL constraint to check if a business object is in a given state or not.

The transition element references its source (*from business state IDREF*) and its target *(to business state IDREF*). Special types of transitions are the one from the initial state (*start*) and those to the end states (*success* and *failure*), since the respective BPSS elements combine pseudo state and transition. Their attributes are used similar to that of the transition element. The mapping of the business collaboration protocol of our purchase order management depicted in Figure 2 leads to the following BPSS code:

```
[26]<BinaryCollaboration ... >
[27]   <Role name="buyer" nameID="R1"/>
[28]   <Role name="seller" nameID="R2"/>
[29]   <Start nameID="St1"
[30]     toBusinessState="request for quote" toBusinessStateIDREF="BTA1"/>
[31]   <BusinessTransactionActivity
[32]     name="request for quote" nameID="BTA1"
[33]     … ### all attributes are listed for order product line 41 – 49 ###
[34]     postCondition="Product.oclInState(Quoted) XOR Product.oclInState(Unavailable)"/>
[35]   <BusinessTransactionActivity
[36]     name="reserve product" nameID="BTA2"
[37]     …
[38]     preCondition=" Product.oclInState(Quoted)"
[39]     postCondition="Product.oclInState(Reserved) XOR Product.oclInState(Unavailable)" />
```

```
[40]   <BusinessTransactionActivity
[41]     name="order product" nameID="BTA3"
[42]     businessTransaction="order product" businessTransactionIDREF="BT3"
[43]     fromRole="buyer" fromRoleIDREF="R1"
[44]     toRole="seller" toRoleIDREF="R2"
[45]     isConcurrent="true" timeToPerform="PT48H"
[46]     beginsWhen="some text"
[47]     endsWhen="some text"
[48]     preCondition=" Product.oclInState(Quoted) OR Product.oclInState(Reserved)"
[49]     postCondition="Product.oclInState(Ordered) XOR Product.oclInState(Aborted)"/>
[50]<Success nameID="Su1" conditionGuard="Product.oclInState(Ordered)"
[51]     fromBusinessState="order product" fromBusinessStateIDREF="BTA3"/>
[52]   <Failure nameID="F1" conditionGuard="Product.oclInState(Aborted)"
[53]     fromBusinessState="order product" fromBusinessStateIDREF="BTA3"/>
[54]   <Failure nameID="F2" conditionGuard="Product.oclInState(Unavailable)"
[55]     fromBusinessState="reserve product"fromBusinessStateIDREF="BTA2"/>
[56]<Failure nameID="F3" conditionGuard=" Product.oclInState(Unavailable)"
[57]     fromBusinessState="request for quote "fromBusinessStateIDREF="BTA1"/>
[58]   <Transition
[59]     nameID="T-BTA1-D1"
[60]     fromBusinessState="request for quote" fromBusinessStateIDREF="BTA1"
[61]     toBusinessState="Decision1" toBusinessStateIDREF="D1">
[62]     conditionGuard="BusinessSuccess"
[63]     <ConditionExpression
[64]       expressionLanguage="OCL" expression="Product.oclInState(Quoted)"/>
[65]   </Transition>
[66]   <Transition nameID="T-D1-BTA2" ... > ...</Transition>
[67]   <Transition nameID="T-D1-BTA3"
[68]     fromBusinessState="Decision 1" fromBusinessStateIDREF="D1"
[69]     toBusinessState="order product" toBusinessStateIDREF="BTA3">
[70]     conditionGuard="BusinessSuccess"
[71]     <ConditionExpression
[72]       expressionLanguage="OCL" expression="Product.oclInState(Quoted)"/>
[73]   </Transition>
[74]   <Transition nameID="T-BTA2-BTA3" ... > ...   </Transition>
[75]   <Decision name="Decision1" nameID="D1" />
[76]</BinaryCollaboration>
```

4. Constraints on UMM Models

In this section we demonstrate that a UMM business collaboration model may be shared by multiple business environments, if they share a significant overlap in their business requirements. However, the UMM model must not be ambiguous. Hence, a generic model for multiple business environments must unambiguously specify constraints for each business environment. One option for specifying constraints is natural language which results in ambiguity. Another option is formal languages like OCL.

However, it turned out that OCL does not hundred percent fit our requirements. OCL typically specifies invariant conditions that must hold for the system being modeled. In our work we do not want to specify invariants on the stereotypes of the meta-model - where OCL would be appropriate - , but constrain the assignment of tagged values - where OCL is not appropriate. For this reason we decided to develop a constraint language in the spirit of OCL. We use the basic OCL syntax - extended by an elsif-clause - and some properties of OCL. Attributes describing the business environment are defined as omnipresent. Hence, we develop a set of templates that are defined by an extended version of the Bachus-Naur-Form (BNF) for specifying business environment specific constraints.

In this section we show how to apply these templates to our case study of purchase order management for books and for tourism products, which differ somewhat but share a significant overlap. Furthermore, we demonstrate how to map a multi-context UMM model to BPSS. The BPSS choreography must be specified exactly for the business environment of the participating applications. Thus, a mapping from the multi-context purchase order management model leads to different BPSS files for the book and the tourism case. This means that our business context-specific constraints must be evaluated before generating a corresponding BPSS code.

4.1. Constraints on Business Transactions

The basic flow of a business transaction is independent of the business context. This means that the business transaction *request for quote* – depicted in Figure 1 and Figure 3 – is always executed by a *buyer* and a *seller* and that a *quote request envelope* is sent from *obtain quote* to *calculate quote* and a *quote envelope* in return. However,

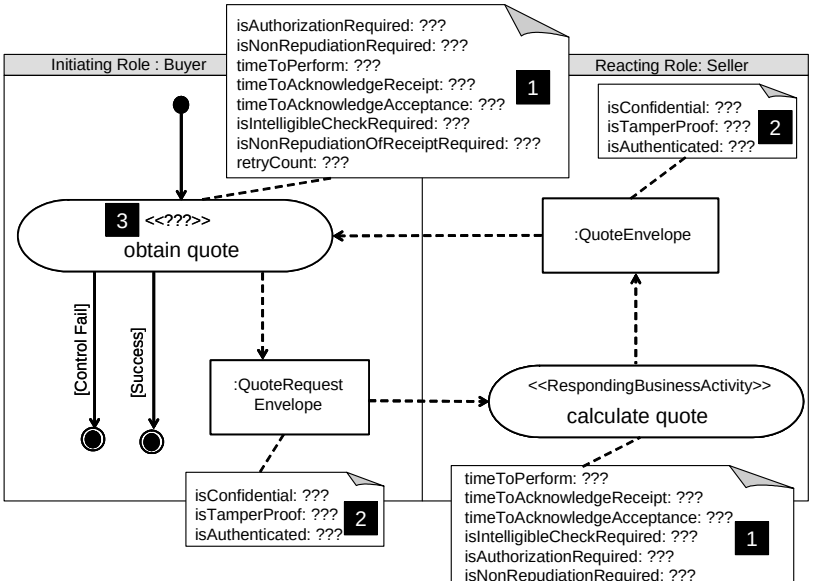

Figure 3. Business Transaction: Request for quote

Figure 3 already highlights those parts of a business transaction that are subject to business environment specific variations:

1 Type of business transaction

2 Tagged values for requesting and responding business activities

3 Tagged values for information envelopes

In order to define business environment specific constraints we first have to define in which business environment our business transaction may be used. Accordingly, a tagged value of the business transaction must capture these business environments. The best way to describe a business environment is by the concept of business context as introduced by ebXML core components [UN/CEFACT 2003d]. In this specification business context is defined as a mechanism of qualifying and refining core components according to their use under particular business circumstances. We enlarge the scope of this definition to apply the mechanism not only to core components but to any UMM artifact.

The business context in which the business collaboration takes place is specified by a set of categories and their associated values. In ebXML eight categories have been identified: *business process*, *product classification*, *industry classification*, *geopolitical* and *official constraints*, *business process role*, *supporting role*, and *system capabilities*. We split the category *business process* into the two categories *business collaboration* and *business transaction*, because both exist in a UMM model and must be distinguished. The context categories are not limited to the ones identified, but we do not recommend the use of other categories.

The definition of a business environment is nothing else than a constraint on the activity graph of a business collaboration protocol. Thus, we have to develop a constraint template for this purpose. The syntax is similar to that of OCL invariants for classifiers. In case of defining the business environment of the business collaboration protocol, we specify the corresponding business transaction after the OCL keyword context and followed by the keyword inv for invariants. Instead of defining constraints on attributes of a classifier, we assign constraints to the tagged values describing the business environment. The business environment is defined as key-value-pairs for the context categories connected by Boolean operators. Before going into the details of the different variations for business transactions, we introduce the syntax to define the business context. Note, for a better understanding the context drivers refer to readable text instead of appropriate code lists (e.g. UN/SPSC). For reasons of simplicity our example uses only the categories *product classification* and *industry classification* in addition to the categories *business collaboration* and *business transaction.*

The template for defining invariants of a business transaction is defined in BNF further below. Although our order management collaboration seems to be rather general, we restrict it to two business environments for demonstration purposes. The example constraint in the note at top of Figure 4 restricts our business transaction to the book and the tourism case.

BusinessTransactionInvariant ::=
context <*BusinessTransaction*> **inv:** <BusinessContextStatement>

BusinessContextStatement :: =
[<BusinessContext> [<BooleanOperator> <BusinessContextStatement>]? |
[(<BusinessContextStatement> <BooleanOperator> <BusinessContextStatement>)]
BusinessContext ::= <BusinessContextDriver> <relationalOperator> "<literal>"

BusinessContextDriver ::= **BusinessCollaboration** | **BusinessTransaction** |
ProductClassification | **IndustryClassification** | **Geopolitical** |
Official Constraints | **BusinessProcessRole** | **SupportingRole** |
SystemCapabilities | <OtherBusinessContextDriver>

OtherBusinessContextDriver ::= <literal>
BooleanOperator ::= **AND** | **OR** | **XOR**
relationalOperator ::= = | > | < | >= | <= | <>

The type of business transaction (1) is the first business environment specific variation we are going to look at. In the book case, *request for quote* is a query against an existing catalog. The responder already has the information available before receiving the request. By definition this is a *query/response* transaction. In the tourism case the responder does not have the information available beforehand. Thus it is a *request/response* transaction. We assume that not only the requesting business activity is stereotyped according to the business transaction type, but the business transaction carries a corresponding tagged value as well. By the following constraint assigned to a business transaction either the tagged value for business transaction type is the same for all business environments or an *if*-statement is used for business context specific variations. The variations are specified by checking the business environment in the *if*- or *elsif*-clause and setting the type of the business transaction in the *then*-clause. A default value for the business transaction type must be specified in the *else*-clause. It is the one that will be used in the stereotype of the requesting business activity.

context <*BusinessTransaction*> **inv:**
BusinessTransactionType = "<BusinessTransactionType>" /
[if <BusinessContextStatement>
then BusinessTransactionType = "<BusinessTransactionType>"
[**elsif** <BusinessContextStatement>
then BusinessTransactionType = "<BusinessTransactionType>"
]*
[**else BusinessTransactionType =** "<BusinessTransactionType>"
endif]

BusinessTransactionType ::= **InformationDistribution** | **Notification** |
QueryResponse | **RequestConfirm** |
RequestResponse | **CommercialTransaction**

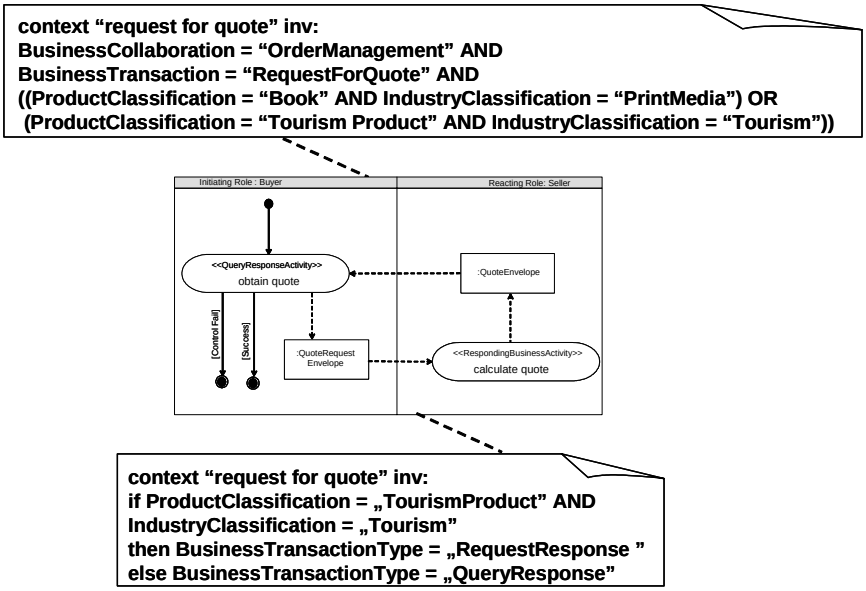

Figure 4. Valid Business Environments and Constraint on the Business Transaction Type

The resulting constraint of our example case study is depicted in the note at bottom of Figure 4. A mapping of the business transaction and its type to BPSS leads to different values in the *pattern* attribute of the element *business transaction*. The code below shows the BPSS code for both examples. The line numbers correspond to those used to introduce business transactions in section 3.2.

```
<!-- Book Example -->
[1]   <BusinessTransaction
[2]      nameID="BT1" name="request for quote"
[3]      pattern="QueryResponse" isGuaranteedDeliveryRequired="true">
[25]  ... </BusinessTransaction>

<!-- Tourism Example -->
[1]  <BusinessTransaction
[2]    nameID="BT1" name="request for quote"
[3]    pattern="RequestResponse" isGuaranteedDeliveryRequired="true">
[25]  ... </BusinessTransaction>
```

The second variation (2) based on different business environments concerns the characteristics of the requesting and the responding business activity. Therefore, we define invariants for both types of activities. If there are no context-sensitive variations the invariant defines one or more of the tagged values *time to perform*, *time to acknowledge receipt*, time *to acknowledge acceptance*, *is authorization required*, *is non repudiation required*, *is intelligible check required*, *is non repudiation of receipt required* and *retry count*. Of course, the last two tagged values are reserved for requesting business activities only. A logical *and* separates the assignments. Otherwise the variations are specified by checking the business environment in an if-statement. The basic structure of the if-statement is very similar to the one for assigning the business transaction type. In this case - and in all other constraints from this point forward - the else-clause is optional.

```
RequestingBusinessActivityTaggedValuesConstraint ::=
context <RequestingBusinessActivity> inv:
   <MultipleRequestingBusinessActivityTaggedValueStatement> |
   [if <BusinessContextStatement>
   then <MultipleRequestingBusinessActivityTaggedValueStatement>
   [elsif <BusinessContextStatement>
     then <MultipleRequestingBusinessActivityTaggedValueStatement>
    ]*
    [else <MultipleRequestingBusinessActivityTaggedValueStatement>]?
 endif]
```

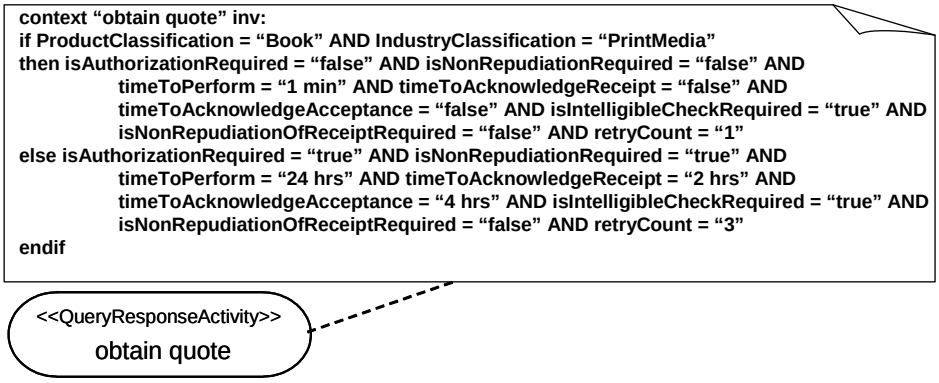

Figure 5 Constraint on a Requesting Business Activity: Obtain Quote

MultipleRequestingBusinessActivityTaggedValueStatement ::=

<RequestingBusinessActivityTaggedValueStatement>

[**AND** <MultipleRequestingBusinessActivityTaggedValueStatement>]?

RequestingBusinessActivityTaggedValueStatement ::=

<RequestingBusinessActivityTaggedValue> = “<literal>”

RequestingBusinessActivityTaggedValue ::= **isAuthorizationRequired** |

isNonRepudiationRequired | **timeToPerform** | **timeToAcknowledgeReceipt** |

timeToAcknowledgeAcceptance | **isIntelligibleCheckRequired** |

isNonRepudiationOfReceiptRequired | **retryCount**

In Figure 5 we present the constraints on the requesting activity *obtain quote*. As mentioned before, we assume that in the book case the *obtain quote* results in a simple query on an existing catalog that everyone is allowed to run. In the tourism case only paying customers may obtain a quote, which is calculated on an individual basis. It follows that the security requirement are higher and the reaction times are longer in the tourism case. Consequently, the tagged values differ considerably.

The mapping of the constraint of the requesting business activity *obtain quote* results in the two BPSS fragments below. Since no acknowledgements are required in the book case, the attributes *time to acknowledge receipt* and *time to acknowledge acceptance* are simply omitted in the first fragment.

```
<!-- Book Example -->
[4]  <RequestingBusinessActivity
[5]    nameID="RBA1" name="obtain quote"
[6]    isAuthorizationRequired="false"
[7]    isNonRepudiationRequired="false"
[8]    isNonRepudiationReceiptRequired="false" isIntelligibleCheckRequired="true" retryCount="1" >
[9]    …
[14]</RequestingBusinessActivity>
<!-- Tourism Example -->
[4]  <RequestingBusinessActivity
[5]      nameID="RBA1" name="obtain quote"
[6]      isAuthorizationRequired="true" timeToAcknowledgeReceipt="PT2H"
[7]      timeToAcknowledgeAcceptance="PT4H" isNonRepudiationRequired="true"
[8]      isNonRepudiationReceiptRequired="false" isIntelligibleCheckRequired="true" retryCount="3" >
[9]      …
[14]</RequestingBusinessActivity>
```

The last business environment specific constraint concerns the security parameters assigned to the information envelopes (3). The structure of the constraint is very similar to the one for the requesting business activity illustrated above. Thus, we show just a rudimentary fragment of its syntax below. Here, the constraint is used to set the tagged values of *is confidential*, *is tamper proof*, and *is authenticated*.

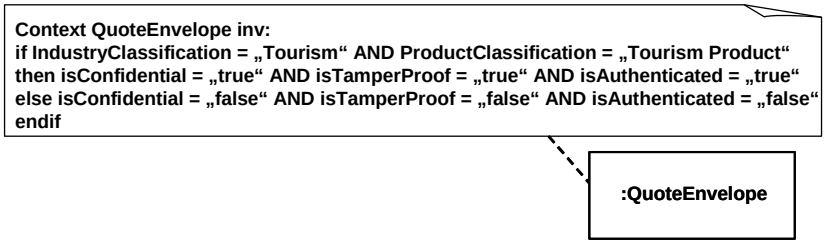

Figure 6 Constraint on an Information Envelope: Quote Envelope

EnvelopeInvariant ::=
context <*Envelope*> **inv:**
<MultipleEnvelopeTaggedValueStatement> |
[if <BusinessContextStatement>
then <MultipleEnvelopeTaggedValueStatement>
// rest of if-statement is truncated
endif]

MultipleEnvelopeTaggedValueStatement ::=
<EnvelopeTaggedValueStatement>
[**AND** <MultipleEnvelopeTaggedValueStatement>]?

EnvelopeTaggedValueStatement ::=
<EnvelopeTaggedValue>="<literal>"

Envelope TaggedValue ::= isConfidential | isTamperProof | isAuthenticated

In our example we assume that the *quote envelope* usually is neither confidential, tamper proof, nor authenticated. This default applies to our book example. However, in the tourism environment it is exactly the other way round. The resulting constraint is depicted in Figure 6 and the two BPSS fragments are listed below. Note that a UMM *true* corresponds to *transient* in BPSS.

```
<!-- Book Example -->
[19]<DocumentEnvelope
[20]   nameID="DE2" name="QuoteEnvelope"
[21]   businessDocument="Quote" businessDocumentIDREF="BD2"
[22]   isAuthenticated="none" isConfidential="none"
[23]   isTamperDetectable="none"/>
<!-- Tourism Example -->
[19]<DocumentEnvelope
[20]   nameID="DE2" name="QuoteEnvelope"
[21]   businessDocument="Quote" businessDocumentIDREF="BD2"
[22]   isAuthenticated="transient" isConfidential="transient"
[23]   isTamperDetectable="transient"/>
```

4.2. Constraints on Business Collaboration Protocols

In this subsection we concentrate on business environment specific constraints on a business collaboration protocol that are subsequently mapped to BPSS binary collaboration. Again the basic flow of a business collaboration like purchase order management – depicted in Figure 2 and Figure 7 - is independent of the business environment. However, some transitions may be defined as invalid for certain business environments. Consequently, we se business context-sensitive variations for the following parts of a business collaboration protocol:

4 Tagged values, pre- and post-conditions of business transaction activities

5 Transitions

The context-sensitive constraints on the tagged values *time to perform* and *is concurrent* of business transaction activities (4) are very similar to the constraints 2 and 3. So we do not detail them any further. However, their pre- and post condition may vary as well. In our example a product may be ordered after it is quoted. A reservation

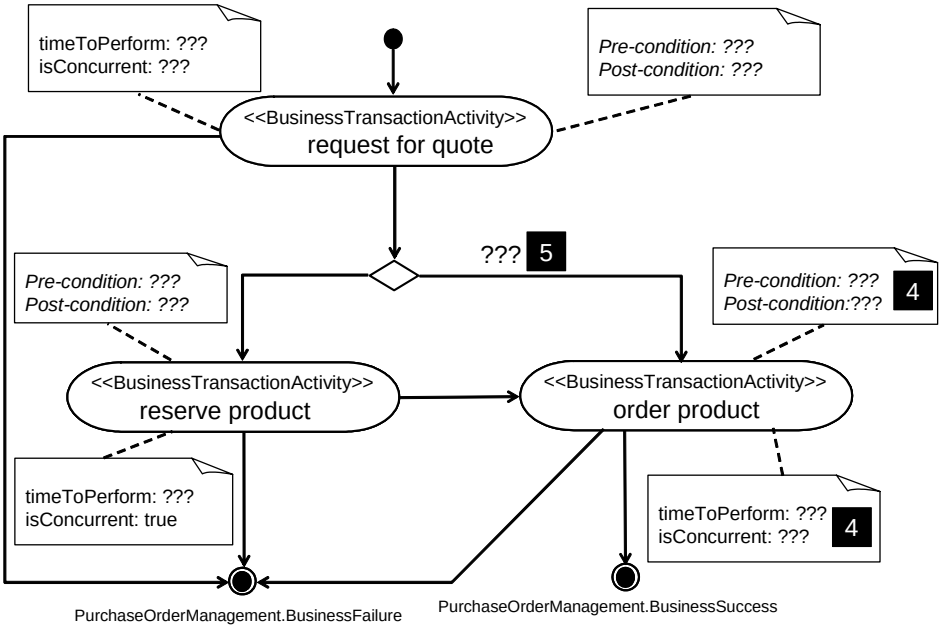

Figure 7. Business Collaboration Protocol: Purchase Order Management

of the product is optional. It follows that a valid pre-condition for *order product* is that the business object *product* is in state *quoted* or is in state *reserved*. The tourism case is even more restrictive. Products must be reserved prior to ordering. Accordingly the state *quoted* is not sufficient for ordering products in tourism. The valid pre-condition for order product is restricted to the fact that a business object *product* must be in state *reserved*.

Thus an invariant for a business transaction activity includes both a block for constraints on tagged values and a block for pre- and post-conditions. The block for pre- and post-conditions is recognized by the keyword *pre:* or *post:*, respectively. Both blocks may be assigned to a business transaction activity independent of the business environment or otherwise both blocks are set in an if-statement similar to the ones used before.

BusinessTransactionActivityInvariant ::=

context <*BusinessTransactionActivity*> **inv:**
[[<MultipleBusinessTransactionActivityTaggedValueStatement>]?
[**pre:** <MultipleBusinessEntityStateConditions>] ?
[**post:** <MultipleBusinessEntityStateConditions>] ?] |
[**if** <BusinessContextStatement>
then [<MultipleBusinessTransactionActivityTaggedValueStatement>]?
[**pre:** <MultipleBusinessEntityStateConditions>] ?
[**post:** <MultipleBusinessEntityStateConditions>] ?
// rest of if-statement is truncated
endif]

MultipleBusinessEntityStateConditions ::=
[<BusinessEntityStateCondition> [<BooleanOperator> <MultipleBusinessEntityStateConditions>] ?] |
[**(**<BusinessEntityStateCondition> <BooleanOperator> <MultipleBusinessEntityStateConditions>**)**]

BusinessEntityStateCondition ::=
[**NOT**]? <*BusinessEntity*>**.oclInState(**<*BusinessEntityState*>**)**

BusinessTransactionActivity ::= “<literal>”
BusinessEntity ::= “<literal>”
BusinessEntityState ::= “<literal>”

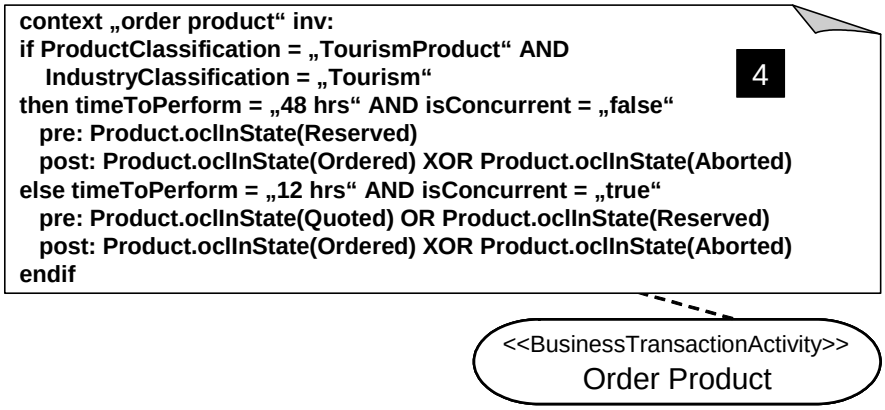

Figure 8. Constraint on a Business Transaction Activity: Order Product

The variation in the tagged values *time to perform* and *is concurrent* as well as in the pre- and post-conditions of *order product* is defined in the constraint of Figure 8. The resulting BPSS code for the business transaction activity *order product* replacing code lines 40 to 49 in the code of section 3.3 is stated below.

```
<!-- Book Example -->
[40]<BusinessTransactionActivity
[41]    name="order product" nameID="BTA3"
[42]    businessTransaction="order product" businessTransactionIDREF="BT3"
[43]    fromRole="buyer" fromRoleIDREF="R1"
[44]    toRole="seller" toRoleIDREF="R2"
[45]    isConcurrent="true" timeToPerform="PT12H"
[46]    beginsWhen="some text"
[47]    endsWhen="some text"
[48]    preCondition=" Product.oclInState(Quoted) OR Product.oclInState(Reserved)"
[49]    postCondition="Product.oclInState(Ordered) XOR Product.oclInState(Aborted)"/>
<!-- Tourism Example -->
[40]<BusinessTransactionActivity
[41]    name="order product" nameID="BTA3"
[42]    businessTransaction="order product" businessTransactionIDREF="BT3"
[43]    fromRole="buyer" fromRoleIDREF="R1"
[44]    toRole="seller" toRoleIDREF="R2"
[45]    isConcurrent="false" timeToPerform="PT48H"
[46]    beginsWhen="some text"
[47]    endsWhen="some text"
[48]    preCondition=" Product.oclInState(Reserved)"
[49]    postCondition="Product.oclInState(Ordered) XOR Product.oclInState(Aborted)"/>
```

Finally, transitions may be business context-sensitive as well (5). We use again the fact that a product must be reserved prior to ordering in the tourism case. Instead of restricting the pre-condition of *order product* another option is to prohibit a transition from *request for quote* to *order product* in the tourism case. In other words the transition from *request for quote* (via the decision state) to *order product* is only valid for the book case. For this purpose we do not develop a new constraint template. Instead we use the guard condition of a transition to limit a transition to certain business environments. It follows, that a transition not valid in all business environments must have a guard condition that follows the syntax of our business context statement.

The constraint cutting the transition from *request for quote* (or better the decision state) to *order product* results in the effect that the transition element in code lines 67 to 73 is part of the book case, whereas the tourism case does not include this transition at all.

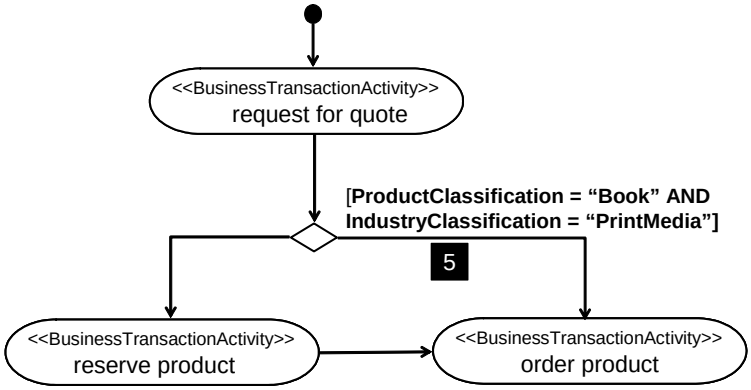

Figure 9. Business Environment dependent Transition

```
<!-- Book Example -->
[67]<Transition nameID="T-D1-BTA3"
[68]     fromBusinessState="Decision 1" fromBusinessStateIDREF="D1"
[69]     toBusinessState="order product" toBusinessStateIDREF="BTA3">
[70]     conditionGuard="BusinessSuccess"
[71]     <ConditionExpression
[72]        expressionLanguage="OCL" expression="Product.oclInState(Quoted)"/>
[73]  </Transition>
<!-- Tourism Example -->
<!-- No transition "T-D1-BTA3 -->
```

5. Summary

In this paper we elaborated on the design of inter-organizational business processes. We did not develop our own modeling methodology. Instead we decided to base our approach on the well accepted UN/CEFACT modeling methodology (UMM) which we are also currently leading under the auspices of the United Nations. Our approach presented in this paper contributes to the development and improvement of UMM.

A UMM model concentrates on the process in the collaborative space between the partners and not on processes that are internal to a business partner. In the original UMM it is important that a business collaboration model is always developed in the context of its business environment. This guarantees that UMM delivers unambiguous definitions on how business partners have to interact in order to reach a common business goal. Due to the focus on the collaborative space, UMM models will not be translated into executable software. But they exactly define the interface for each business partner's side. These interfaces and their choreography may be described in a machine-readable format such as WSDL and BPEL in Web Services or business service interfaces (BSI) and BPSS in ebXML. We elaborated on the mapping of UMM models to BPSS.

By the means of a simple case study of purchasing books and purchasing tourism products, we demonstrated that a business process may be very similar in two different environments and differs only in a very few aspects. Developing UMM models for each business context variation leads to a countless number of similar models. Our proposed approach is to share a business collaboration model between different environments, if they have a significant overlap. The slight differences in the business process definition of different business environments must be defined by appropriate constraints. We have developed a constraint language for business environment specific adaptations of UMM artifacts. Our approach leads to multi-context business collaborations that still unambiguously define how a process is executed in a certain business environment.

The fact that a UMM business collaboration model is defined for multiple business environments affects the mappings to the choreography language. Since choreography languages like BPSS must be specified in the business environment of the business application(s) monitoring and executing the business process, a BPSS process is always specific to its business environment. We have demonstrated that a single UMM business collaboration model for purchase order management results in different BPSS choreographies for each business environment. In future work we are developing business environment specific mappings from UMM to BPEL and WS-CDL.

REFERENCES

Arkin, A., "Business Process Modeling Language," Version 1.0., 2002. http://www.bpmi.org/bpml-spec.esp

Andrews, T., et al., "Business Process Execution Language for Web Services," Version 1.1., 2003.
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbizspec/html/bpel1-1.asp

Chau, P.Y.K., "Inhibitors to EDI Adoption in Small Businesses: An Empirical Investigation," *Journal of Electronic Commerce Research*, Vol. 2, No. 2:78-88, May 2001.

Chen, M., A.N.K. Chen, and B.B.M. Shao, "The Implications and Impacts of Web Services to Electronic Commerce Research and Practices," *Journal of Electronic Commerce Research*, Vol. 4, No. 4:128-139, November 2003.

Chen, M., M.J.Meixell, "Web Services Enabled Procurement in the Extended Enterprise: An Architectural Design and Implementation," *Journal of Electronic Commerce Research*, Vol. 4, No. 4:140-155, November 2003.

ISO, "Open-edi Reference Model," ISO/IEC JTC 1/SC30 ISO Standard 14662, 1997

Kavantzas, N., et al, "Web Services Choreography Description Language," Version 1.0., W3C, 2004. http://www.w3.org/TR/ws-cdl-10

Kramler, G., E. Kapsammer, W. Retzschitzegger, and G. Kappel, "Towards Using UML 2 for Modelling Web Services Collaboration Protocols," Proceedings of the 1st Int. Conf. on Interoperability of Enterprise Software and Applications, Geneva, Switzerland, 2005.

Jung, J.-Y. , W. Hur, S.-H. Kang, and H. Kim, "Business Process Choreography for B2B Collaboration," *IEEE Internet Computing*, Vol. 8., No. 1:37-45, January/February 2004.

Lee, R.M., "Documentary Petri Nets: A Modeling Representation for Electronic Trade Procedures," Business Process Management, Models, Techniques, and Empirical Studies, Springer LNCS, Vol. 1806, pp. 259 – 375, 2000.

Lenz, K., and A. Oberweis, "Interorganizational Business Process Management with XML Nets," Advances in Petri Nets, Springer LNCS, Vol. 2472, 2003.

Leymann, F., D. Roller, and M.-T. Schmidt, "Web Services and Business Process Management," *IBM Systems Journal*, Vol. 41, No. 2, 2002.

Little, M., "Transactions and Web Services," *Communications of the ACM*, Vol. 46, No. 10:49-54, 2003.

Mantell, K., "From UML to BPEL - Model Driven Architecture in a Web Services World," IBM Developer Works, 2003. http://www-128.ibm.com/developerworks/webservices/library/ws-uml2bpel/

OMG, "Object Constraint Language Specification," March 2003. http://www.omg.org/cgi-bin/doc?formal/03-03-13

Papazoglou, M.P., "Web Services and Business Transactions," *WWW, Internet and Web Information Systems*, Kluwer, Vol. 6, No. 1:49-91, 2003.

Peltz, C., "Web Services Orchestration and Choreography," *IEEE Computer*, Vol. 36, No. 10:46-52, October 2003.

Provost, W., "UML for Web Services," XML.com, August 2003,http://www.xml.com/lpt/a/ws/2003/08/05/uml.html

RosettaNet, "RosettaNet Implementation Framework," Core Specification Version 02.00.01., 2002. http://www.rosettanet.org/rnif

Thöne, S., R. Depke, and G. Engels, "Process-Oriented, Flexible Composition of Web Services with UML," Int'l Workshop on Conceptual Modeling Approaches for e-Business: A Web Service Perspective (eCOMO), Tampere, Finland, October 2002.

UN/CEFACT, "UMM Meta Model," Revision 12, January 2003a. http://www.untmg.org/dmdocuments/UMM_MM_V20030117.pdf

UN/CEFACT, "UMM User Guide," Revision 12, September 2003b. http://www.untmg.org/dmdocuments/UMM_UG_V20030922.pdf

UN/CEFACT, "ebXML Business Process Specification Schema," Version 1.10, October 2003c. http://www.untmg.org/dmdocuments/BPSS_v110_2003_10_18.pdf

UN/CEFACT, "Core Components Technical Specification," Version 2.01, November 2003. http://www.untmg.org/dmdocuments/CCTS_v201_2003_11_15.pdf

van der Aalst, W.M.P., "Interorganizational Workflows: An Approach based on Message Sequence Charts and Petri Nets," *Systems Analysis - Modelling - Simulation*, Vol. 34, No. 3:335-367, 1999.

White, S, "Business Process Modeling Notation Working Draft," Version 1.0., May 2004. http://www.bpmn.org/Documents/BPMN%20V1-0%20May%203%202004.pdf

W3C, "Web Service Choreography Interface," Version 1.0., August 2002. http://www.w3.org/TR/wsci/