OCL-Constraints for UMM Business Collaborations

Birgit Hofreiter, Christian Huemer, and Werner Winiwarter

Department of Computer Science and Business Informatics
University of Vienna, Liebiggasse 4, 1010 Vienna, Austria
{birgit.hofreiter, christian.huemer, werner.winiwarter}@univie.ac.at

Abstract. Recently, a trend towards business processes in Business-to-Business e-Commerce (B2B) is apparent. One of the most promising approaches is UN/CEFACT's modeling methodology (UMM) based on UML. However, developing a new UMM model for each small variation in a business process would lead in a multitude of "similar" business processes. Thus, a more generic UMM model together with well-defined constraints for different business environments is a better approach to ensure unambiguity. In this paper we develop templates for such constraints based on an extended version of OCL.

1 Motivation

For a long time standardization in Business-to-Business e-Commerce (B2B) followed a pure data centric approach. Recent standardization approaches take business processes into account. The most prominent examples include: Business Process Execution Language (BPEL) [2], Business Process Modeling Language (BPML) [1], and ebXML Business Process Specification Schema (BPSS) [13]. Since all of them are XML-based, software tools are able to process the choreography and execute the busines process. In contrast, UN/CEFACT's modeling methodology (UMM) [16] starts from the business requirements in order to define a choreography that meets the business needs. UMM uses the Unified Modeling Language (UML) for describing the business aspects of the business processes and the information exchanged. The resulting choreography provides semantics to be expressed in the XML languages mentioned above.

Usually, a UML diagram does not provide all relevant aspects of a specification. There exist additional constraints that cannot be expressed in the graphical syntax. The preferred language for specifying these constraints in UML is the Object Constraint Language (OCL) [12]. Since UMM is based on UML it seems to be straight forward to specify constraints in OCL. The current Revision 12 of the UMM User Guide references and even mandates the use of OCL for specifying pre- and post-conditions, rules, guards, etc. However, it does not show in any instance how to use OCL in UMM.

The goal of this paper is to define how to use OCL in UMM. UMM does not make use of all existing UML features. It defines a very strict UML Profile for the specific purpose of modeling B2B business processes, so-called business collaborations.

Inasmuch UMM puts UML into a very small corset, which needs only a limited set of constraint types. Consequently, UMM requires only a small subset of OCL. Therefore, we develop OCL-based templates that reflect all useful constraints for UMM business collaborations. Since OCL originally does not focus on activity graphs and does not mention access to tagged values, we make some necessary extensions to OCL.

The remainder of this paper is structured as follows: Section 2 concentrates on related work in the area of business processes for B2B environments. In Section 3 we introduce the core concepts of UMM. We keep them to a minimum necessary to understand how our OCL-based templates will fit into. Section 4 defines OCL-based templates for the UMM artefacts business collaboration protocol and business transaction. The notation of our templates is an extended Backus Naur. Form A short summary in Section 5 concludes the paper.

2 Related Work

Today different approaches exist for choreographing atomic Web Services to complex business processes. Microsoft based XLANG [11] on the pi-calculus, whereas IBM developed the Web Services Flow Language (WSFL) [8] on the foundation of petri nets. The first organization to combine these two approaches was BPMI with their Business Process Modeling Language (BPML) [1]. Later on BEA, IBM and Microsoft started a unification of XLANG and WSFL that became known as Business Process Execution Language (BPEL) [2,9]. Currently this approach seems to be the winner among the competing standards. Another well know approach is W3C's Web Services Choreography Interface (WSCI) [18] that describes only one partner's participation in a business process. Similarly to Web Services, ebXML provides a stack of protocols to standardize B2B on top of XML. The protocol for describing the choreography of message exchanges between business partners is ebXML Business Process Specification Schema (BPSS) [13].

All protocols mentioned above describe the behavior between Web Services and/or the execution side of a business process. They do not consider the design of a business process by a business process analyst. For this purpose BPMI is developing the Business Process Modeling Notation (BPMN) [17]. This notation presents the amalgamation of best practices in the business process modeling community. Another option for a graphical syntax is UML. RosettaNet uses a UML-based methodology to develop their Partner Interface Processes (PIPs) [10]. UN/CEFACT started the development of its methodology on top of UML. During the ebXML initiative the company EDIFECS - that owned copyright of the methodology used in RosettaNet - transferred these copyrights to UN/CEFACT. Inasmuch the current version 12 of UMM [16] represents also a successor of RosettaNet's methodology.

UN/CEFACT's vision is developing business process models for global e-business. These business process models must not include any ambiguity. 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 unambiguity. This results in a key difference

between UMM and the XML-based approaches. The XML-based approaches describe an executable process. Consequently, this process must be defined in a specific business environment. In UMM a business process model is valid in more business environments. The semantics of an executable process are derived by applying the constraints defined for a specific business environment to the generic model. In the future, transformation rules from UMM to BPEL, BPSS, etc., will enable to derive executable business process from a common generic basis. This transformation goes beyond the scope of this paper. In our paper [6] we demonstrate the transformation from UMM to ebXML BPSS.

One option for specifying constraints is natural language which results in ambiguity. Another option is formal languages which are often hard to understand by businessexperts or system modelers. There exist rule based languages which have been developed for e-business in a Web environment, e.g. Business Rules Markup Language (BRML) [3]. Nevertheless, UMM needs a constraint language that reflects its meta-model. Since UMM is UML-based, the preferred language for specifying constraints is the Object Constraint Language (OCL) [12]. OCL has been developed by IBM as a business modeling language. Later it became part of OMG's set of UML specifications. It is a formal language that is said to be easy to read and write by modelers.

3 UN/CEFACT's Modeling Methodology (UMM)

UMM consists of 4 views, corresponding patterns, as well as a well-formed meta-model which defines the syntax and semantics for each view. Due to space limitations we will not go into the details of each view. The interested reader is referred to the UMM Meta Model [15] and the UMM User Guide [16]. In this Section we briefly describe those concepts of UMM needed to understand the proposed OCL-based templates. Fig. 1. presents an overview of the most basic concepts. The diagram does not present the UMM meta-model nor is it a class diagram. The graph is used to explain the UMM ontology and each box represents a concept in the UMM ontology.

A *business process* is defined as an organized group of related activities that together create customer value [4]. If all the activities are performed by one organization this leads to an intra-organizational business process. In B2B the activities are executed by different organizations which collaborate to create value. UMM concentrates on the unambiguous definition of an inter-organizational business processes and calls it *business collaboration*.

A business collaboration is performed by two (= binary collaboration) or more (multi-party collaboration) business partners. A business collaboration might be complex involving a lot of activities between business partners. However, the most basic business collaboration is a binary collaboration realized by a request from one side and an optional response from the other side. This simple collaboration is a unit of work that allows roll back to a defined state before it was initiated. Therefore, this special type of collaboration is called *business transaction*.

Since UMM is based on UML, it uses the concept of use cases to capture requirements. In case of a complex business collaboration the requirements are described in a so-called *business collaboration protocol use case*. These requirements

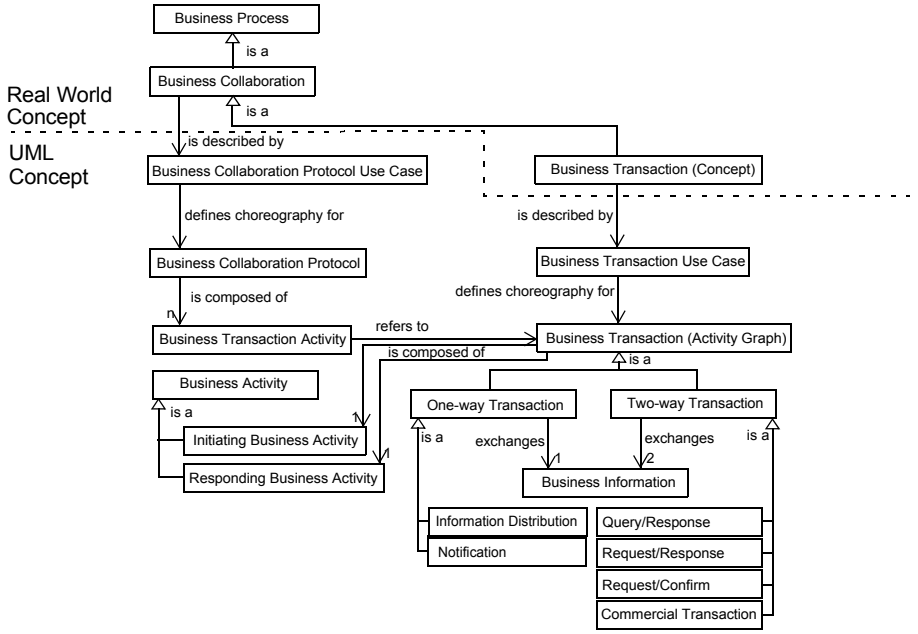

Fig. 1. UMM in a Nutshell

lead to a choreography of activities in order to create the customer value. The activity graph representing this choreography is called *business collaboration protocol* (c.f. Fig. 2). Each activity shown in a business collaboration protocol refers to exactly one business transaction. Therefore, each activity of the business collaboration protocol is called a *business transaction activity*. Each of these activities is characterized by the tagged values *timeToPerform* and *isConcurrent*.

The requirements of a business transaction are described by a *business transaction use case*. Again, the requirements lead to a choreography of the business transaction. The resulting activity graph is what is really called *business transaction* in UMM (c.f. Fig. 3). One might argue, that business transaction activity and business transaction present the same concept. Since different UML elements - an activity and an activity graph - are required in the UML notation, these concepts are distinguished in UMM.

The activity graph of a business transaction is always composed of two *business activities*, an *initiating business activity* performed by the initiator and *reacting business activity* performed by the other business partner. In a *one-way transaction* business information is exchanged only from the *initiating business activity* to the *reacting business activity*. In case of *two-way transaction* the reacting business activity returns business information to the initiating business activity. The UML notation of an object flow is used to show the exchange of business information.

In UMM we distinguish two one-way transactions - *notification* and *information distribution* - and four two-way transactions - *query/reponse*, *request/confirm*, *request/response* and *commercial transaction*. These types of business transactions cover all known legally binding interactions between two decision making applications as defined in Open-edi [7]. Furthermore, the type of business transaction is manifested in

the defaults for the tagged values of the initiating/requesting business activity: *isAuthorizationRequired*, *isNonRepudiationRequired*, *timeToPerform*, *timeToAcknowledgeAcceptance*, *isNonRepudiationOfReceiptRequired*, and *recurrence*.

4 OCL-based Templates for UMM

Having introduced the basic concepts of UMM, it becomes evident that OCL-based templates are useful only for certain artefacts. Use Cases capture the requirements which result in OCL constraints. Constraints do not apply to use cases themselves. Consequently, candidates for OCL-based templates are activity graphs for business collaboration protocols and business transactions as well as class diagrams for business information exchanged. In this paper, we concentrate on the choreography of the activity graphs. Constraints on business information exchanged cannot be explained within the page limit and will be a topic of another paper.

The following two subsections present the OCL-based templates for business collaboration protocols and business transactions. Each template is demonstrated by an example. These examples refer to two very simple case studies. The first one is order management of books and the second one is order management of tourism products. For more details on this case study we refer to our paper introducing business context variations in UMM [5].

4.1 Constraints for Business Collaboration Protocols

The choreography of a business collaboration protocol follows a description provided in the corresponding use case description. Fig. 2. shows the business collaboration protocol of our example. The order management either begins by a search for product or by the query for the reservation list. After a search it is possible to order or reserve a product. Both activities require the customer to be registered. If the result of a search was not satisfying another search is performed or the reserved products are queried. After a reservation was performed the next activity is either a new search or the query for the reserved products. Note that querying products requires customers to be registered, because otherwise they were not able to make a reservation. After querying the reserved products, a product might be ordered. The other choice is to perform a new search. The business collaboration always ends after ordering a product. However, the search for product, the reservation, and the presentation of the reserved products might also be the last activity with the consequence that no book is ordered.

A business collaboration is valid in one or more business environments. Thus, the business environments are specified in a tagged value of the business collaboration. The best way to describe a business environment is by the concept of business context as introduced by ebXML core components [14]. In this specification business context is defined as a mechanism for 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 also 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, official

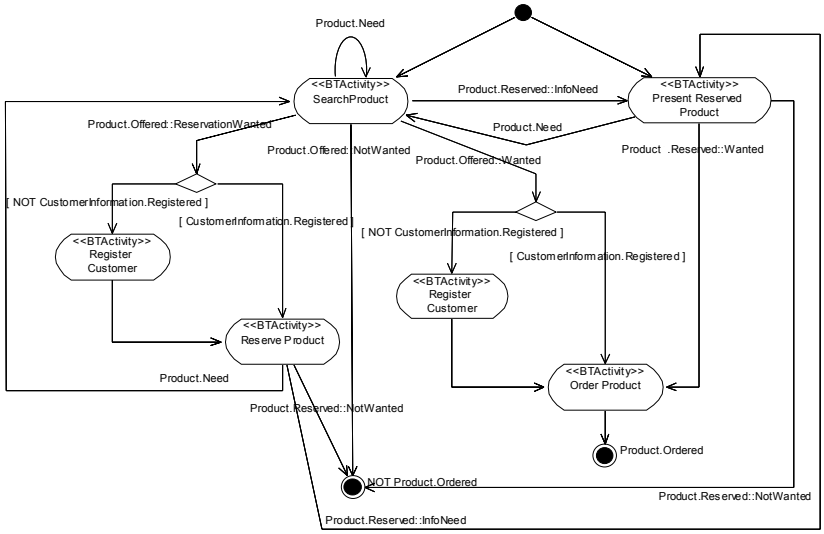

Fig. 2. Business Collaboration Protocol for Order Management

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 then a constraint on the activity graph of a business collaboration protocol. Thus, the definition of an OCL constraint seems to be straight forward. However, OCL was designed to specify invariants of classes and pre- and post-conditions for methods. We need access to the tagged values of an activity graph (and other UML elements). The OCL specification [12] does not consider this type of access. OCL allows invariants for classifiers only. In our approach, we apply invariants to other UML elements as well. The syntax is similar to that of invariants for classifiers. In case of defining the business environment of the business collaboration protocol, we specify the corresponding business collaboration protocol 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 name-value-pairs for the context categories connected by boolean operators.

The template for defining invariants of a business collaboration protocol is defined in BNF further below. The template is followed by an example constraint. Although our order management collaboration seems to be rather general, we restrict it to two business environments for demonstration purposes. The example constraint restricts our business collaboration protocol to the book order management case and the tourism product order management case.

BusinessCollaborationProtocolInvariant ::=
context <*BusinessCollaborationProtocol*> **inv:** <BusinessContextStatement>

BusinessContextStatement :: =
[<BusinessContext> [<BooleanOperator> <BusinessContextStatement>]? |
[(<BusinessContext> <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 ::= **=** | **>** | **<** | **>=** | **<=** | **<>**

Example:

context OrderManagementBusinessCollaborationProtocol inv:
BusinessCollaboration = "OrderManagement"
AND (Product Classification = "Book" OR Product Classification = "Tourism Product")
AND (IndustryClassification = "PrintMedia" OR IndustryClassification = "Tourism")

A business collaboration protocol choreographs business transaction activities. The business environment for each business transaction is identical to the one of the business collaboration protocol. The tagged values of one and the same business transaction activity - which are the concurrency flag and the time to perform - might vary for mutually exclusive subsets of the overall business environment. In the example below *search product* can happen concurrently and must be completed in 24 hours by default. The default applies to the tourism case, whereas searching for books cannot be concurrent and must be completed in 12 hours.

The variations in the tagged values are constraints on the business transaction activity. Thus, we define invariants of business transaction activities. If no variations for the tagged values exist, we simply define the values for *isConcurrent* and *timeToPerform*. Otherwise, we use an if-statement to check the tagged value of the business environment and set the other tagged values if appropriate. The else-clause contains the default values. Unfortunately, OCL does not include an elsif-clause in the if-statement. In reality there exist many different business environments resulting in different combinations of default values. To avoid nested if statements and for reasons of readability we have extended the OCL statement to include an elsif-clause.

BusinessTransactionActivityInvariant ::=
context <*BusinessTransactionActivity*> **inv:**
<MultipleBusinessTransactionActivityTaggedValueStatement> |
[if <BusinessContextStatement>
then <MultipleBusinessTransactionActivityTaggedValueStatement>
[**elsif** <BusinessContextStatement>
then <MultipleBusinessTransactionActivityTaggedValueStatement>
]*
[**else** <MultipleBusinessTransactionActivityTaggedValueStatement>]?
endif]

MultipleBusinessTransactionActivityTaggedValueStatement ::=
<BusinessTransactionActivityTaggedValueStatement>
[**AND** <MultipleBusinessTransactionActivityTaggedValueStatement>]?

BusinessTransactionActivityTaggedValueStatement ::= <BusineesTransactionActivityTaggedValue>**="**<literal>**"**

BusinessTransactionActivityTaggedValue ::= **isConcurrent** | **timeToPerform**

Example:
context SearchProduct inv:
if ProductClassification = “Book” AND IndustryClassification= “PrintMedia“
then timeToPerform = “12 hrs” AND isConcurrent = “false”
else timeToPerform = “24 hrs” AND isConcurrent = “true”

Each business transaction activity requires some preconditions to be met before execution and results in some post-conditions. OCL supports the definition of pre- and post-conditions. According to the UMM User Guide pre- and post-conditions reflect well-defined states in the life-cycle of business entities. For checking the state of an object OCL provides the method *oclInState* which returns a boolean. In our example, *order product* requires that the business entity *product* is either in state *offered* or *reserved* as well as business entity *customer information* is in state *registered*. After executing *order product* a product will be either in state *ordered* or *order failed*. However, the pre- and post-conditions might vary again with respect to the business environment. This fact is accomplished by using an if-clause similar to the one above for tagged value variations. In our example, we suppose that a tourism product might not be ordered without prior reservation. Consequently, the business entity *product* must be in state *reserved* for *order product*. This fact is shown in the if-statement of the example below.

BusinessTransactionActivityPreAndPostConditions ::=
context <*BusinessTransactionActivity*>
[[**pre:** <MultipleBusinessEntityStateConditions>] ?
[**post:** <MultipleBusinessEntityStateConditions>] ?] |
[**if** <BusinessContextStatement>
then
[**pre:** <MultipleBusinessEntityStateConditions>] ?
[**post:** <MultipleBusinessEntityStateConditions>] ?
rest of if-clause is truncated
endif]

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

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

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

Example:
context OrderProduct
if ProductClassification = “TourismProduct” AND IndustryClassification = “Tourism”
then
pre: Product.oclInState(Reserved)
AND CustomerInformation.oclInState(Registered)
post: Product.oclInState(Ordered) XOR Product.oclInState(OrderFailed)
elsif ProductClassification = “Book” AND IndustryClassification = “PrintMedia”
then
pre: (Product.oclInState(Offered) OR Product.oclInState(Reserved))
AND CustomerInformation.oclInState(Registered)
post: Product.oclInState(Ordered) XOR Product.oclInState(OrderFailed)
endif

The last template for the business collaboration protocol specifies constraints on the transitions between business transaction activities. The transition from one business transaction activity to another requires not only the completion of the first activity, but

also the occurrence of an event on the initiator's side of the next activity. For example, the transition from *search product* to *order product* requires the completion of *search product* that, hopefully, results in the state *offered*. However, this does not mean that the buyer must order the product. First, the buyer has to decide that he/she wants the offered product. This decision is modeled as an event that results in the sub-state *wanted* of the parent state *offered*. Furthermore, an optional guard applies to transitions. Valid guards are the context of the business environment and business entity states. In our example the transition from *search product* to *order product* is limited to the book case, because in tourism a reservation is required prior to ordering. Furthermore, the state of *customer information* guards the transition.

BusinessTransactionAcitivityTransition ::=
`context from` <*BusinessTransactionActivity*> **`to`** <*BusininessTransactionActivity*>
`Event:` <MultipleBusinessEntityStateConditions>
`Guard:` <GuardStatement>

GuardStatement ::=
[<Guard> [<BooleanOperator> <GuardStatement>]? | [(<Guard> <BooleanOperator> <GuardStatement>)]

Guard :: = <BusinessContextStatement> | <MultipleBusinessEntityStateConditions>

Example:

context from SearchProduct to OrderProduct
Event: Product.oclInState(Offered::Wanted)
Guard:ProductClassification = "Book" AND IndustryClassification = "PrintMedia"
AND CustomerInformation.InOclState(Registered)

4.2 Constraints for Business Transactions

Each business transaction activity of the business collaboration protocol is refined by a separate activity graph called a business transaction. Fig. 3 depicts the business transaction *search product*. The customer performs *request a search* as initiating activity that produces a *search request* document. This document is input to the reacting activity *perform search* which is executed by the seller. The reacting activity outputs the *search result* document that is returned to the initiating activity. Since there is a response that does not immediately result in a contractual obligation and the responder has the information (about the product) already available, the transaction is of type *query/response*. The initiating activity is stereotyped accordingly.

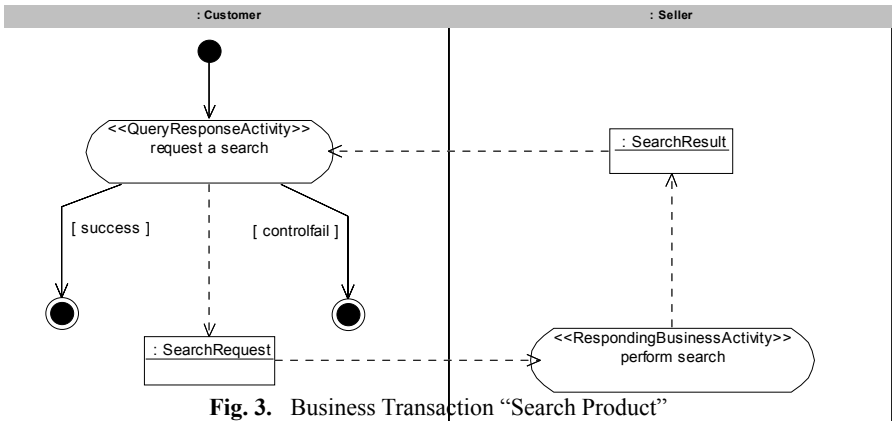

Fig. 3. Business Transaction "Search Product"

First, we define the business environment for the business transaction, which covers both of our example cases. The constraint statement is similar to that for the business collaboration protocol. The business environment is defined as a string of name-value-pairs for the context categories connected by boolean operators.

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

Example:
context SearchProduct inv:
BusinessCollaboration = “OrderManagement” AND BusinessTransaction = “SearchProduct”
AND (Product Classification = “Book” OR Product Classification = “Tourism Product”)
AND (IndustryClassification = “PrintMedia” OR IndustryClassification = “Tourism”)

Both the initiating business activity and the responding business activity are characterized by a well-defined set of tagged values. Again the instances of the tagged values might vary for different subsets of the overall business environment. The code fragments below refer to constraints for tagged values on the initiating business activity. The ones for the responding business activity are quite similar. In our example we define that for the book case the maximum time to perform is 4 hours. There is no need for acknowledgments, authorization and non-repudiation. In case of control failures the initiating activities restarts the transaction 3 times before giving up.

InitiatingBusinessActivityTaggedValuesConstraint ::=
context <*InitiatingBusinessActivity*> **inv:**
<MultipleInitiatingBusinessActivityTaggedValueStatement> |
[**if** <BusinessContextStatement>
then <MultipleInitiatingBusinessActivityTaggedValueStatement>
rest of if-clause is truncated
endif]

MultipleInitiatingBusinessActivityTaggedValueStatement ::=
<InitiatingBusinessActivityTaggedValueStatement>
[**AND** <MultipleInitiatingBusinessActivityTaggedValueStatement>]?

InitiatingBusinessActivityTaggedValueStatement ::=
<InitiatingBusinessActivityTaggedValue> = <literal>

InitiatingBusinessActivityTaggedValue ::= **TimeToAcknowledgeReceipt** |
TimeToAcknowledgeAcceptance | **TimeToPerform** | **AuthorizationRequired** |
NonRepudiationOfOriginAndContent | **NonRepudiationOfReceipt** | **Recurrence**

Example:
context RequestASearch inv:
if ProductClassification = “Book” AND IndustryClassification = “PrintMedia”
then TimeToAcknowledgeReceipt = “Null” AND TimeToAcknowledgeAcceptance = “Null” AND
TimeToPerform = “4 hrs” AND AuthorizationRequired = “false” AND NonRepudiationOfOriginAndContent = “false”
AND NonRepudiationOfReceipt = “false” AND Recurrence = “3”
else ... endif

Finally, there might exist variations in the business transaction type according to the business environment. As mentioned above search product is by default a query/ response transaction, since there are no contractual obligations involved and the responder has the information already available. Imagine that in tourism the information is not already available, but must be calculated by the responder. Accordingly, the transaction type changes to request/response. This is shown in the example below. A more radical variation can happen in case of tacit approval when a commercial transaction (two-way) changes to a notification (one-way).

```
context <BusinessTransaction> inv:
    if  <BusinessContextStatement>
    then BusinessTransactionType = <BusinessTransactionType>
    ### rest of if-clause truncated ###
    endif
```

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

Example:
context SearchProduct inv:
if ProductClassification = “TourismProduct” AND IndustryClassification = “Tourism”
then BusinessTransactionType = “RequestResponse”
else BusinessTransactionType = “QueryResponse”

5 Summary

B2B e-Commerce standardization is more and more directed towards business processes. Most approaches are in the area of Web Services. Their goal is to describe a choreography for an executable business process that is assembled from a set of Web Services. For this purpose the process must be defined in a specific business environment. In contrary, UMM is a methodology that starts from gathering user requirements and develops business process and information models that are independent of the underlying B2B technology (Web Services, ebXML, EDI, etc.). UMM’s goal are unambiguous business process models for global e-business. For the sake of reusability, a business process model must be generic enough to adopt to different business environments. Nevertheless, it must be specific enough to unambiguously describe a business process execution in a given business environment.

In order to fulfill this pretension UMM must deliver generic models that exactly define the constraints for adopting to a certain business environment. This requires a constraint language that is adjusted to the UMM meta-model. Since UMM is UML-based it seems to be straight forward to use OCL for this purpose. In the same way as UMM restricts the UML meta-model, we must restrict the flexibility of OCL. Thus, this paper defines OCL templates specially designed for UMM artefacts. The constraints for business collaboration protocols are: (1) definitions of applicable business environments, (2) invariants for tagged values of business transaction activities, (3) pre- and post-conditions of business transaction activities, and (4) invariants for transitions. The templates for business transactions are: (1) definitions of applicable business environments, (2) invariants of tagged values for initiating and reacting business activities (3) invariants for business transaction types.

We also started to develop OCL templates for adopting the business information exchanged in a business transaction to different business environments. We plan to summarize this complex topic in the near future. In our paper [6] we map UMM models (developed for a specific business environment) to ebXML BPSS. It is our goal to demonstrate mapping for other choreography languages as well. Moreover, we want to show how a generic UMM model including constraint statements for multiple business environments will map to different choreographies in the same choreography language.

References

1. Arkin, A.; Business Process Modeling Language (Version 1.0); November 2002; http://www.bpmi.org/bpml-spec.esp
2. Andrews, T., Curbera, F., Dholakia, H., Goland Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weerawarana, S.; Business Process Execution Language for Web Services, Version 1.1, May 2003 http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbizspec/html/bpel1-1.asp
3. Grosof, B.N., Labrou, Y.; An Approach to using XML and a Rule-based Content Language with an Agent Communication Language"; Proceedings of the IJCAI-99 Workshop on Agent Communication Languages (ACL-99); Stockholm (Sweden), August 1999
4. Hammer, M., Champy, J.; Reengineering the Corporation: Manifesto for Business Revolution; Harper Business; 1993
5. Hofreiter, B., Huemer, C.; Modeling Business Collaborations in Context; Proceedings of On The Move to Meaningful Internet Systems 2003: OTM 2003 Workshops; Springer; November 2003
6. Hofreiter, B., Huemer; Transformation of UMM Models to ebXML BPSS; to appear in: Proceedings of XML4BPM Workshop, Marburg (Germany) March 2003; http://www.ifs.univie.ac.at/~ch/UMM2BPSS.pdf
7. ISO; Open-edi Reference Model; ISO/IEC JTC 1/SC30 ISO Standard 14662; 1995
8. Leymann, F.; Web Services Flow Language (WSFL 1.0); May 2001 http://www-306.ibm.com/software/solutions/webservices/pdf/WSFL.pdf
9. Leymann, F., Roller, D., Schmidt, M.-T.; Web Services and Business Process Management; IBM Systems Journal, Vol. 41, No. 2, 2002
10. RosettaNet; RosettaNet Implementation Framework: Core Specification V02.00.01; March 2002; http://www.rosettanet.org/rnif
11. Thatte, S.; XLANG - Web Services for Business Process Design; June 2001; http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm
12. OMG; Object Constraint Language Specification; http://www.omg.org/cgi-bin/doc?formal/03-03-13
13. UN/CEFACT; ebXML - Business Process Specification Schema v1.10; October 2003; http://www.untmg.org/downloads/General/approved/ebBPSS-v1pt10.zip
14. UN/CEFACT; Core Components Technical Specification V2.01; November 2003; http://www.untmg.org/downloads/General/approved/CEFACT-CCTS-Version-2pt01.zip
15. UN/CEFACT; UMM Meta Model, Revision 12; January 2003; http://www.untmg.org/downloads/General/approved/UMM-MM-V20030117.zip
16. UN/CEFACT; UMM User Guide, Revision 12; September 2003; http://www.untmg.org/downloads/General/approved/UMM-UG-V20030922.zip
17. White, S; Business Process Modeling Notation Working Draft (1.0); August 2003; http://www.bpmi.org/bpmn-spec.esp
18. W3C; Web Service Choreography Interface (WSCI) 1.0; August 2002; http://www.w3.org/TR/wsci/