From Multi-Context Business Collaboration Models to Context-specific ebXML BPSS

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

Abstract

UN/CEFACT's Modelling Methodology (UMM) is used to analyze and design B2B business processes. We extend UMM by a constraint mechanism for adding business environment-specific constraints to models. Now they might be shared by multiple business environments. The implementation of business process models is supported by choreography languages that can be process by applications. Choreography languages like ebXML BPSS are always specific to the business environment. Thus, it is the goal of this paper to map multi-context UMM models to context-specific BPSS.

1 Motivation

UN/CEFACT's vision is developing business process models for global e-business. The UN/CEFACT's modeling methodology (UMM) is a development process for a consistent analysis and design of inter-organizational business processes. 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. A generic model together with constraints for different business environments is a much more effective approach. Therefore, we extend UMM by a constraint-mechanism in order to develop multi-context business collaboration models.

Connecting different components - like Web Services or ebXML business service interfaces - together to create business processes is the goal of orchestration and choreography languages. The most prominent examples include: Business Process Execution Language (BPEL) Business Process Modeling Language (BPML), and ebXML Business Process Specification Schema (BPSS). The flow described by such an language is processed by a business application in order to execute and/or track a business process. Since a business application is always executed in a certain business environment, the flow must be context-specific to this environment.

In this paper we fill the gap between a multi-context B2B process model and a context-specific business choreography language. We base our approach on ebXML, which is specifically directed towards B2B infrastructure and recommends UMM as modeling methodology. Therefore, we want to derive context-specific ebXML BPSS (version 1.1) business processes from multi-context UMM (version 12) models.

2 Reference Concepts: UMM and BPSS

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 reflected 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 [6] that defines a coherent set of stereotypes, constraints, and tag definitions, i.e. a UML profile for the purpose of modeling business collaborations.

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 is recommending 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 artefacts that are relevant to BPSS are the activity graphs for a business transaction and for a business collaboration protocol. The mapping of these artefacts is described in the following sections. We do not care about any other UMM artefacts. The reader interested in UMM is referred to [1].

3 Mapping Business Transactions

Both UMM and BPSS use the concept of a business transaction in a very similar way. It is the basic building block to define a choreography of a business collaboration between collaborating business partners. Communication in a business collaboration is about aligning the information systems of the business partners. Aligning the information systems means that all relevant business objects (e.g. purchase orders, line items, etc.) are in the same state in each information system. 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. Business transactions are either one way or two way.

3.1. Business Transaction in UMM

In UMM a business transaction is defined by an activity graph that follows a certain pattern. Fig. 1 depicts the *search product* business transaction which follows the typical pattern of a two way transaction. 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* which outputs a first envelope, but does not necessarily end afterwards. The envelope is input to the *responding business activity* which is triggered by its receipt. In case of a two way transaction the requesting activity must still be alive to receive an 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 explained below—allows the initiator to determine the final state of the business object(s). Therefore, the completion of the initiating activity results in a deterministic end state.

UMM adopts the six types of business transactions identified by RosettaNet: *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*

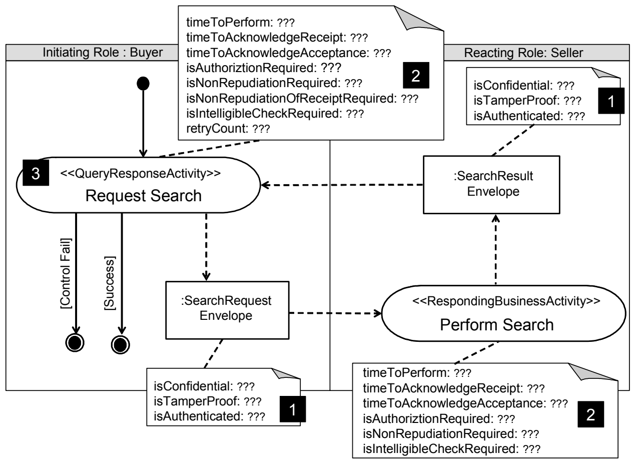

timeToPerform: ???
timeToAcknowledgeReceipt: ???
timeToAcknowledgeAcceptance: ???
isAuthoriztionRequired: ???
isNonRepudiationRequired: ???
isNonRepudiationOfReceiptRequired: ???
isIntelligibleCheckRequired: ???
retryCount: ???
2
Initiating Role : Buyer
Reacting Role: Seller
isConfidential: ???
isTamperProof: ???
isAuthenticated: ???
1
3
<<QueryResponseActivity>>
Request Search
:SearchResult Envelope
[Control Fail]
[Success]
:SearchRequest Envelope
<<RespondingBusinessActivity>>
Perform Search
isConfidential: ???
isTamperProof: ???
isAuthenticated: ???
1
timeToPerform: ???
timeToAcknowledgeReceipt: ???
timeToAcknowledgeAcceptance: ???
isAuthoriztionRequired: ???
isNonRepudiationRequired: ???
isIntelligibleCheckRequired: ???
2

Figure 1. UMM business transaction "search product"

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, a time value specified as tagged value defines the required flow of business signals for acks. An ack of receipt is sent after schema 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.*

The basic flow defined in the business transaction is independent of the business context. However, we identify the candidates that are business context-sensitive. These types of business context variations are marked by the corresponding number in Fig. 1. Additionally, the corresponding values appear as placeholders (*???*).

1 Tagged values for envelope

2 Tagged values for requesting and responding activities

3 Type of transaction

3.2. Business Transaction in BPSS

The transformation of the UMM business transaction *search product* of Fig. 1 to the XML syntax of BPSS is shown below. All values that depend on the business context are again marked by placeholders (*???*). The numbered square refers to the UMM counterpart mentioned above. 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 6 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 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)*. Additionally, the attributes *business document* and *business document IDREF* reference a *business document* element that is covered by the envelope.

```
<BusinessTransaction
  nameID="BT1" name="SearchProduct"
  pattern="???" isGuaranteedDeliveryRequired="true">   [3]
  <RequestingBusinessActivity
    nameID="RBA1" name="RequestSearch"
    isAuthorizationRequired="???" timeToAcknowledgeReceipt="???"   [2]
    timeToAcknowledgeAcceptance="???" isNonRepudiationRequired="???'
    isNonRepudiationReceiptRequired="???"
    isIntelligibleCheckRequired="true" retryCount="?" >
    <DocumentEnvelope
      nameID="DE1" name="SearchRequestEnvelope"
      businessDocument="SearchRequest" businessDocumentIDREF="D1"
      isAuthenticated="???" isConfidential="???"   [1]
      isTamperDetectable="???"/>
  </RequestingBusinessActivity>
  <RespondingBusinessActivity
    nameID="RBA2" name="PerformSearch"
    isAuthorizationRequired="???" timeToAcknowledgeReceipt="???"   [2]
    timeToAcknowledgeAcceptance="???"
    isNonRepudiationRequired="???" isIntelligibleCheckRequired="true">
    <DocumentEnvelope
      nameID="DE2" name="SearchResultEnvelope"
      businessDocument="SearchResult" businessDocumentIDREF="D2"
      isAuthenticated="???" isConfidential="???"   [1]
      isTamperDetectable="???"/>
  </RespondingBusinessActivity>
</BusinessTransaction>
```

3.3. From UMM to BPSS

The best way to describe a business environment is by the concept of business context as introduced by ebXML core components [5]. Business context is defined as a mechanism for qualifying and refining core components according to their use in a particular business environment. We apply this mechanism to business transactions and later on to business collaborations. The business context is specified by a set of eight categories and their associated values: *business process*, *product classification*, *industry classification*, *geopolitical*, *official constraints*, *business process role*, *supporting role*, and *system capabilities*. For demonstration purposes we assume two different business environments, a purchase management for books and one for tourism products. In the oversimplified examples we use only the categories *product classification* and *industry classification*.

In order to allow variations for different business contexts it is necessary to assign a constraint to an affected model element. The principal idea is checking the business context in an *if*-clause and adjusting the element's characteristics accordingly in a *then*-clause. Since UML's OCL [3] is not perfect for constraining tagged values of activity graphs, we developed a constraint language in the spirit of OCL. We use syntax (extended by an *elsif*-clause) and some properties of OCL. Attributes describing the business environment are defined as omnipresent.

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).

```
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 ::= = | > | < | >= | <= | <>
```

The first business context-sensitive variation ($\boxed{1}$) concerns the security parameters assigned to the envelopes. Therefore we define an invariant of the envelope. If there are no context-sensitive variations the invariant defines a boolean value for one or more of the security parameters *is confidential*, *is tamper proof*, and *is authenticated*. A logical *and* separates the assignments. Otherwise the variations are specified by checking the business environment in the *if* or *elsif*-clause and setting the booleans for the security parameters in the *then*-clause. A default value for each security parameter might be given in the *else*-clause.

```
EnvelopeInvariant ::=
context <Envelope> inv:
    <MultipleEnvelopeTaggedValueStatement> |
    [if <BusinessContextStatement>
    then <MultipleEnvelopeTaggedValueStatement>
    [elsif <BusinessContextStatement>
    then <MultipleEnvelopeTaggedValueStatement>
    ]*
    [else <MultipleEnvelopeTaggedValueStatement> ]?
    endif]

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

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

   Envelope TaggedValue ::= isConfidential | isTamperProof |
                            isAuthenticated
```

In our example we assume that the search result 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 Fig. 2.

A transformation of the business transaction including

```
Context SearchResultEnvelope inv:
if IndustryClassification = „Tourism“ AND ProductClassification = „Tourism Product“
then isConfidential = „true“ AND isTamperProof = „true“ AND isAuthenticated = „true“
else isConfidential = „false“ AND isTamperProof = „false“ AND isAuthenticated = „false“
endif
```

:SearchResult Envelope

```
<!-- Book Example -->
<DocumentEnvelope
      nameID="DE2" name="SearchResultEnvelope"
      businessDocument="SearchResult"
1     businessDocumentIDREF="BD2" isAuthenticated="none"
      isConfidential="none" isTamperDetectable="none"/>
<!-- Tourism Example -->
<DocumentEnvelope
      nameID="DE2" name="SearchResultEnvelope"
      businessDocument="SearchResult"
1     businessDocumentIDREF="BD2" isAuthenticated="transient"
      isConfidential="transient" isTamperDetectable="transient"/>
```

Figure 2. Contex-specific search result envelope

the constraint of Fig. 2 from UMM to BPSS will result in two different BPSS specifications. The BPSS specifications will differ in the attributes of the *document envelope* element for *search result envelope*. The resulting code fragments for the book example and the tourism example are also given in Fig. 2. It should be noted that, BPSS allows a more sophisticated instantiation of the security parameters: *none*, *transient*, *persistent*, and *transient-and-persistent*. UMM focuses on the delivery to the receiver, but not on a subsequent storage. The corresponding BPSS value is *transient*.

The second variation (2) based on different business environments concerns the characteristics of the requesting and the responding business activity. The structure of the constraint is very similar to the one for envelopes. Thus, we show just a rudimentary fragment of its syntax below. In Fig. 3 we present the constraints on the requesting activity *request search*. We assume that in the book case the search is a simple query on an existing catalog that everyone can run. In the tourism case the request is possible for paying customers. Consequently, the security requirements are higher and the tagged values differ considerably.

```
RequestingBusinessActivityTaggedValuesConstraint ::=
context <RequestingBusinessActivity> inv:
    <MultipleRequestingBusinessActivityTaggedValueStatement> |
    [if <BusinessContextStatement>
    then <MultipleRequestingBusinessActivityTaggedValueStatement>
    // rest of if-statement is truncated
    endif]
```

Again, the transformation of the initiating business activity including the constraint of Fig. 3 will result in two different BPSS instantiations. The resulting code fragments for the BPSS element *requesting business activity* are shown below. The attributes are instantiated according to the constraint, except for *Time to perform* which does not exist in BPSS. It should be noted that we include the *is authorization required* attribute for reasons of completeness, although ebXML software will currently ignore it.

The type of business transaction (3) is the third variation. In the book case, *search product* is a query against an existing catalog. The responder already has the information available before receiving a 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

```
context RequestSearch inv:
if ProductClassification = „Book“ AND IndustryClassification = „PrintMedia“
then retryCount = „3”  AND TimeToPerform = „4 hrs” AND  isAuthorizationRequired = „false” AND
        isIntelligibleCheckRequired = „true“ AND TimeToAcknowledgeReceipt = „Null” AND
        TimeToAcknowledgeAcceptance = „Null” AND isNonRepudiationRequired = „false” AND
         isNonRepudiationOfReceiptRequired = „false”
elsif ProductClassification = „Book“ AND IndustryClassification = „PrintMedia“
 then retryCount = „2”  AND TimeToPerform = „24 hrs” AND  isAuthorizationRequired = „true” AND
         isIntelligibleCheckRequired = „true“ AND TimeToAcknowledgeReceipt = „2 hrs” AND
         TimeToAcknowledgeAcceptance = „4 hrs” AND isNonRepudiationRequired = „true” AND
         isNonRepudiationOfReceiptRequired = „false”
endif
```

<<QueryResponseActivity>>
Request Search

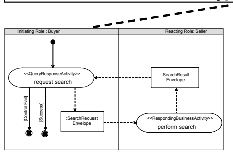
```
<!-- Book Example -->
<RequestingBusinessActivity
      nameID="RBA1" name="RequestSearch"
2     retryCount="3" isIntelligibleCheckRequired="true"
      isAuthorizationRequired="false" isNonRepudiationRequired="false"
      isNonRepudiationReceiptRequired="false">
            <DocumentEnvelope ... />
</RequestingBusinessActivity>

<!-- Tourism Example -->
<RequestingBusinessActivity
      nameID="RBA1" name="RequestSearch"
2     retryCount="2" isIntelligibleCheckRequired="true"
      isAuthorizationRequired="true" timeToAcknowledgeReceipt="PT2H"
      timeToAcknowledgeAcceptance="PT4H" isNonRepudiationRequired="true"
      isNonRepudiationReceiptRequired="false">
            <DocumentEnvelope ... />
</RequestingBusinessActivity>
```

Figure 3. Context on initiating activity request search

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. This time the constraint statement has a mandatory *else*-clause in the *if*-statement, defining a transaction type that corresponds to the stereotype of the requesting business activity. The constraint for the *search product* transaction of our example is depicted in Fig. 4. The two BPSS instances differ in the value for the *pattern* attribute according to their transaction type.

```
context <BusinessTransaction> inv:
    BusinessTransactionType = <BusinessTransactionType> |
    [if <BusinessContextStatement>
    then BusinessTransactionType = <BusinessTransactionType>
    // elsif is truncated
    else <MultipleEnvelopeTaggedValueStatement>
    endif]
```

```
context SearchProduct
if ProductClassification = „TourismProduct” AND
IndustryClassification = „Tourism”
then BusinessTransactionType = „RequestResponse ”
else BusinessTransactionType = „QueryResponse”
```



```
<!-- Book Example -->
<BusinessTransaction
      nameID="BT1" name="SearchProduct"
      pattern="QueryResponse" isGuaranteedDeliveryRequired="true">
3     ... </BusinessTransaction>

<!-- Tourism Example -->
<BusinessTransaction
      nameID="BT1" name="SearchProduct"
      pattern="RequestResponse" isGuaranteedDeliveryRequired="true">
3     ... </BusinessTransaction>
```

Figure 4. Context on business transaction search product

4 Mapping Business Collaborations

4.1. Business Collaboration Protocol in UMM

In UMM an inter-organizational business process is called business collaboration. Its choreography is defined by an activity graph called business collaboration protocol. Fig. 5 shows an extract of a business collaboration protocol. In practice, the business collaboration protocol describes collaborations between two partners only - although it is not limited to. In UMM v12 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.

We use business context-sensitive variations for the following parts of a business collaboration protocol:

- $\boxed{4}$ Tagged values, pre- and post-conditions of business transaction activities
- $\boxed{5}$ Transitions

4.2. Binary Collaboration in BPSS

BPSS 1.10 specifies only collaborations between two partners. However, 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*, *decision*. The code fragment listed below represents the XML equivalent to Fig. 5. The two *role* elements specify the two collaborating business partners. A

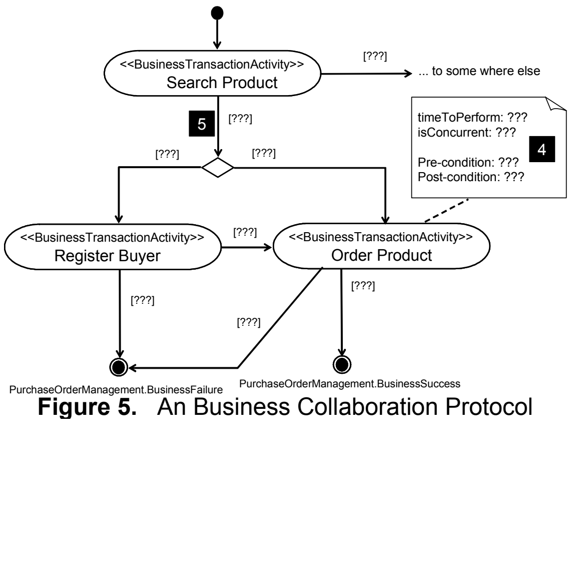


Figure 5. An Business Collaboration Protocol

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 might state an OCL constraint to check if a business object is in a given state or not.

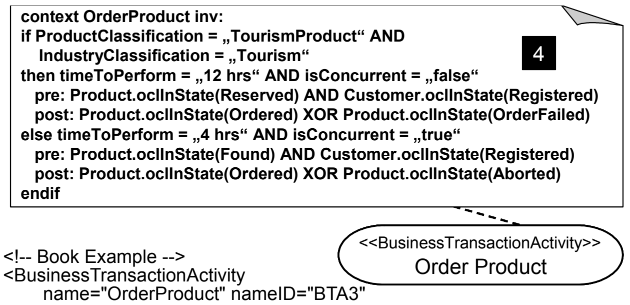
The *transition* element references its source (*from business state IDREF)* and its target (*to business state IDREF)*. Context-sensitive guards are specified in the *condition guard attribute* and the *condition expression* child element. Special types of transitions are the one from the initial state (*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.

```
<BinaryCollaboration ... >
  <Role name="buyer" nameID="R1"/>
  <Role name="seller" nameID="R2"/>
  <Start nameID="St1"
    toBusinessState="SearchProduct" toBusinessStateIDREF="BTA1"/>
  <BusinessTransactionActivity
    name="SearchProduct" nameID="BTA1"
    businessTransaction="SearchProduct" businessTransactionIDREF="BT1"
    fromRole="buyer" fromRoleIDREF="R1"
    toRole="seller" toRoleIDREF="R2"
    isConcurrent="???" timeToPerform="???"
    preCondition="???" beginsWhen="some text"
    endsWhen="some text" postCondition="???"/>
  <BusinessTransactionActivity
    name="RegisterBuyer" nameID="BTA2" />
  <BusinessTransactionActivity
    name="OrderProduct" nameID="BTA3" />
  <Success nameID="Su1" conditionGuard="???"
    fromBusinessState="OrderProduct" fromBusinessStateIDREF="BTA3"/>
  <Failure nameID="F1" conditionGuard="???"
    fromBusinessState="OrderProduct" fromBusinessStateIDREF="BTA3"/>
  <Failure nameID="F2" conditionGuard="???"
    fromBusinessState="RegisterBuyer"fromBusinessStateIDREF="BTA2"/>
  <Transition
    nameID="T-BTA1-D1" conditionGuard="???"
    fromBusinessState="SearchProduct" fromBusinessStateIDREF="BTA1"
    toBusinessState="Decision1" toBusinessStateIDREF="D1">
    <ConditionExpression
      expressionLanguage="OCL" expression="???"/>
  </Transition>
  <Transition nameID="T-D1-BTA2" ... > ...</Transition>
  <Transition nameID="T-D1-BTA3" ... > ...</Transition>
  <Transition nameID="T-BTA2-BTA3" ... > ...   </Transition>
  <Transition nameID=”T-BTA1-toSomewhereElse” ...> ... </Transition>
  <Decision name="Decision1" nameID="D1">
    <ConditionExpression expressionLanguage="OCL"
      expression="CustomerInformation.oclInState(Confirmed)"/>
  </Decision>
</BinaryCollaboration>
```

4.3. From UMM to BPSS

The context-sensitive constraints on the tagged values *time to perform* and *is concurrent* of business transaction activities ($\boxed{4}$) are very similar to the constraints $\boxed{1}$ and $\boxed{2}$. So we do not detail them any further. However, their pre- and post condition might vary as well. In our example a product might be ordered after it is listed in a search result. It follows that a pre-condition for *order product* is that the business object *product* is in state *found*. The tourism case is even more restrictive. It requires that *product* is in state *reserved*. Accordingly the state *found* is not sufficient. The OCL statement *object.oclInState(state : OclState) : Boolean* is used to check the state. The *customer* must be in state *confirmed* anyway. Thus an invariant for a business transaction activity - see example in Fig. 6 - includes constraints on tagged values and a block for pre- and post-conditions recognized by the key words *pre* or *post*:

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

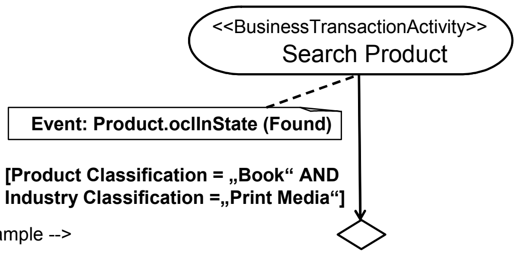


```
context OrderProduct inv:
if ProductClassification = „TourismProduct“ AND
   IndustryClassification = „Tourism“
then timeToPerform = „12 hrs“ AND isConcurrent = „false“
  pre: Product.oclInState(Reserved) AND Customer.oclInState(Registered)
  post: Product.oclInState(Ordered) XOR Product.oclInState(OrderFailed)
else timeToPerform = „4 hrs“ AND isConcurrent = „true“
  pre: Product.oclInState(Found) AND Customer.oclInState(Registered)
  post: Product.oclInState(Ordered) XOR Product.oclInState(Aborted)
endif
```

```
<!-- Book Example -->
<BusinessTransactionActivity
    name="OrderProduct" nameID="BTA3"
    businessTransaction="OrderProduct"
    businessTransactionIDREF="BT3"
    fromRole="buyer" fromRoleIDREF="R1"
    toRole="seller" toRoleIDREF="R2"
4   isConcurrent="true" timeToPerform="PT4H"
    preCondition="Product.oclInState(Found)
                  AND CustomerInformation.oclInState(Confirmed)"
    postCondition="Product.oclInState(Ordered) XOR
                  Product.oclInState(Aborted)"
    beginsWhen="some text" endsWhen="some text" />
<!-- Tourism Example -->
<BusinessTransactionActivity
    name="OrderProduct" nameID="BTA3"
    businessTransaction="OrderProduct"
    businessTransactionIDREF="BT3"
    fromRole="buyer" fromRoleIDREF="R1"
    toRole="seller" toRoleIDREF="R2"
4   isConcurrent="false" timeToPerform="PT12H"
    preCondition="Product.oclInState(Reserved) AND
                  Customer.oclInState(Registered)"
    postCondition="Product.oclInState(Ordered) XOR
                  Product.oclInState(Aborted)"
    beginsWhen="some text" endsWhen="some text" />
```

Figure 6. Context on Business Transaction Activity

Finally, transitions might be business context-sensitive as well (5). In this case we do not specify a constraint. However, if a guard is specified it must follow the syntax of our business context statement. In addition to the completion of the preceding activity a transition might require some business objects to be in a certain state. In this case an event for the transition must follow the syntax *object.oclInState(state : OclState) : Boolean.* Fig. 7 depicts the situation where the transition from search product to the decision node (later splitting into order product and register customer) is valid only in the book case and, furthermore, requires *product* to be in state *found*. Accordingly, the transition will not appear in the BPSS for tourism. The transition in the book case includes a condition guard element that is set to *business success* which is a BPSS pre-defined high level end state for a business transaction. The related object state *found* of *product* is defined in the sub-element *condition expression*.


<<BusinessTransactionActivity>>
Search Product
Event: Product.oclInState (Found)
[Product Classification = „Book“ AND
Industry Classification =„Print Media“]

```
<!-- Book Example -->
<Transition
5   nameID="T-BTA1-D1"
    fromBusinessState="SearchProduct"
    fromBusinessStateIDREF="BTA1"
    toBusinessState="Decision1"
    toBusinessStateIDREF="D1"
    conditionGuard="BusinessSuccess">
    <ConditionExpression
       expressionLanguage="OCL"
       expression="Product.oclInState(Found)"/>
</Transition>
```

Figure 7. Context-sensitive Guards on Transitions

5 Summary

In this paper we filled the gap between business process definitions in UMM as defined by standard organizations and business process definitions in BPSS executed by software tools. The former must fit multiple business environments, whereas the latter are specified in a specific business environment. Hence, a mapping from a single UMM model results in multiple BPSS files - one for each environment. We identified those elements of business transactions and business collaborations that depend on the business context. In order to allow context-specific variations we developed an appropriate constraint language. We demonstrated how these constraints influence the mapping.

References

[1] B. Hofreiter, C. Huemer, “Modeling Business Collaborations in Context”, *Proceedings of On The Move to Meaningful Internet Systems 2003: OTM 2003 Workshops*; Springer LNCS, Catania (Italy), Nov. 2003

[2] B. Hofreiter, C. Huemer, W. Winiwarter, “OCL-Constraints for UMM Business Collaborations”, *Proceedings the 5th Int’l Conf. on Electronic Commerce and Web Technologies (EC-Web 2004)*; Springer LNCS, Zaragoza (Spain), Aug. 2004

[3] OMG, “Object Constraint Language Specification”, http://www.omg.org/cgi-bin/doc?formal/03-03-13

[4] UN/CEFACT, “ebXML - BPSS v1.10”, Oct. 2003, http://www.unece.org/cefact/tmg/BPSS-v1pt10.pdf

[5] UN/CEFACT, “Core Components Technical Specification V2.01”, Nov. 2003, http://www.unece.org/cefact/tmg/cefact_ccts_v2_01.pdf

[6] UN/CEFACT, “UMM Meta Model, Revision 12”, Jan. 2003, http://www.unece.org/cefact/tmg/UMM-MM-V20030117.pdf