Business Collaboration Models and their Business Context-dependent Web Choreography in BPSS

BIRGIT HOFREITER, CHRISTIAN HUEMER
Institute of Distributed and Multimedia Systems, University of Vienna, Liebiggasse 4/3-4, 1010 Vienna, Austria

WERNER WINIWARTER
Institute of Scientific Computing, University of Vienna, Universitätsstraße 5/9, 1010 Vienna, Austria

Received: October 13 2004; accepted: February 1 2005

***Abstract*— Prior to conducting business via the Web, business partners agree on the business processes they are able to support. In ebXML, the choreography of these business processes is described as an instance of the so-called business process specification schema (BPSS). For execution purposes the BPSS must be defined in the exact business context of the partnership. Reference models for B2B processes developed by standard organizations usually span over multiple business contexts to avoid a multitude of similar processes. In this paper we present how business collaboration models following the UN/CEFACT Modeling Methodology (UMM) are expressed in ebXML BPSS. To allow a mapping from multi-context business collaboration models to a context-specific choreography in ebXML BPSS we extend UMM to capture constraints for different business contexts.**

***Index Terms*— web commerce and E-business, E-business models and architectures, business collaboration models**

I. MOTIVATION

For a long time e-business activities followed a pure data centric approach. Most attention was spent on business document types, messaging formats, and application interface descriptions. Recent standardization approaches take also business processes into account. The most prominent examples include: Business Process Execution Language (BPEL) [2], [8], Business Process Modeling Language (BPML) [1], and ebXML Business Process Specification Schema (BPSS) [11]. These standards provide an XML schema to describe the choreography of business processes. It is the vision that software tools will be able to process the choreography and execute the business process. For this purpose the business process must be defined in the exact business context. This means an enterprise describes a process exactly as it is executed in its organization and not in any other company. Similarly, in B2B, the inter-organizational process is described for the exact business environment of the partners and not for any other partnership.

Standard organizations in the B2B community have the goal to define commonly accepted inter-organizational business processes. There is a trend towards standardizing the business process independent of the underlying IT infrastructure. The first organizations following this approach were RosettaNet and UN/CEFACT. Both organizations defined a methodology based on UML in order to describe the semantics of an inter-organizational business process. In 2000 the company EDIFECS asked UN/CEFACT to maintain their methodology, which was used by RosettaNet. UN/CEFACT merged this approach into its own approach called UN/CEFACT's modeling methodology (UMM). Also other organizations, like SWIFT, EAN*UCC, and TM Forum aligned their methodologies with UMM.

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 specific business environment. Developing a new model for each variation would result in a multitude of models. Thus, a generic model together with constraints for different business contexts is a much more effective approach to ensure unambiguity.

In this paper we fill the gap between multi-context B2B process models and the context-specific business choreography language used for implementation in a Web environment. We base our approach on ebXML, which is specifically directed towards a B2B infrastructure for the Web 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. Since UMM currently does not say anything about how to specify context-dependent constraints, we developed our own templates [6].

The remainder of this paper is structured as follows: In Section II we introduce the two standards our approach is based upon: UMM and BPSS. The two key artefacts that have to be transformed from UMM to BPSS are business transactions and business collaborations. The former are described in Section III and the latter in Section IV. Both sections have the same structure. First we define the basic concepts. Then we describe how these concepts are modeled in UMM. Next we define how the model is generally mapped to BPSS.

Finally, we concentrate on those parts of the transformation that depend on the business context. A summary in Section V concludes the paper.

II. Reference Concepts: UMM and BPSS

UN/CEFACT started its work on the UN/CEFACT modeling methodology in 1997, as base technology for their next generation EDI standards succeeding UN/EDIFACT. UMM is based on the business operational view (BOV) of the ISO standard Open-edi [7], which 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 [13], which defines a coherent set of stereotypes, constraints, and tag definitions, i.e. a UML profile, for the purpose of modeling business collaborations.

The UMM meta model consists of 4 views in order to describe business collaboration models. In UMM the term "business collaboration" is used for a business process involving two or more business partners to accomplish a common business goal. Firstly, the Business Domain View (BDV) provides a framework for understanding existing business processes and categorizing these business processes into business areas and process areas. Secondly, the Business Requirements View (BRV) identifies possible business collaborations and further elaborates on these collaborations. It describes processes and resources used to achieve certain objectives and the resulting commitments. Thirdly, the Business Transaction View (BTV) presents the view of the business process analyst. It defines the choreography of the business collaboration and structures the business information exchanged. Finally, the Business Service View (BSV) considers the interaction sequences between network components in order to map the business collaboration semantics to collaborating application systems.

UMM is not a mandatory part of the ebXML specifications. The ebXML specifications comprise standards for messaging, registries, collaboration profiles/agreements, core components, and business processes. The ebXML standard to describe business processes is the business process specification schema (BPSS). The relationship to other ebXML standards is as follows: A BPSS instance will reference business document types that are constructed from building blocks called core components. A business partner will reference in its profile (Collaboration Protocol Profile) all the BPSS instances of the business processes it is able to support. Both the instances of the CPP and the BPSS will be stored in an ebXML registry so that potential business partners are able to locate them. Once business partners agree to execute a certain type of business process, they will reference the corresponding BPSS instance in their agreement (Collaboration Protocol Agreement).

ebXML does not require any specific modeling language or modeling methodology, but the architecture specification recommends to use UMM to develop business process definitions [3]. BPSS provides an XML schema to specify a collaboration between business partners, and to supply configuration parameters for the partners' runtime systems in order to execute that 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.

Like UMM, BPSS supports the specification of business transactions and the choreography of business transactions into business collaborations. It does not deal with the definition of business document types, it rather references them. Therefore, in the next section we concentrate on business transactions and in section IV on business collaborations. We do not care about any other UMM artefacts. The reader interested in UMM is referred to [5], [14]. For demonstration of the concepts in the next two sections we use a simple example of a purchase order management for ordering books and ordering tourism products. We presume that both processes are similar to share a common definition, but each includes some context-specific variations.

III. Mapping Business Transactions

A. Business transaction semantics

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.

We distinguish two very basic types of business transactions. Firstly, 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. This is the case of a one way business transaction, because business information (not including business signals for acks) flows only from the initiating to the reacting business partner. Secondly, 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. This is the case of 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 original state requires another business transaction. For example, 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.

B. Business transaction in UMM

In UMM a business transaction is defined by an activity graph that follows a certain pattern already used in RosettaNet [10]. 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 object flow from the initiating activity to the reacting business activity is mandatory. The object flow from the reacting activity to the initiating activity exists only for two way transactions. Figure 1 depicts the *search product* business transaction, which follows the typical pattern of a two way transaction.

The activity graph does not show any control flows between the activities. In fact, the UMM interpretation of the control flow is neither compliant to UML 1.4 nor to 2.0. However, the interpretation is accepted in the e-business community. The transaction starts with the initiating business activity, which outputs a first envelope, but does not necessarily end afterwards. The envelope is input to the reacting activity, which is triggered by its receipt. In case of a two way transaction the initiating activity must still be alive to receive an envelope produced by the reacting business activity. The requirement of synchronized states – which is realized both by 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 initiating activity results in a deterministic end state.

The reacting activity is always stereotyped as *responding business activity*. We prefer the term reacting activity because responding is misleading in the case of one way transactions. The stereotype of the initiating activity is named after the underlying transaction type. UMM adopts the six types of business transactions identified by RosettaNet. These cover all known legally binding interactions between two decision making applications as defined in Open-edi [7].

There exist two different types of one-way transactions. If the business information sent is a formal non-reputable notification, the transaction is called notification, otherwise it is called information distribution. Furthermore, there exist four different types of two-way transactions. If the responder already has the information available, it is a query/response transaction. If the responder does not have the information, but no pre-editor context validation is required before processing, the transaction is of type request/confirm. If the latter is required, the next question is whether the transaction results in a residual obligation between the business partners to fulfill terms of a contract. If the answer is "yes", it is a commercial transaction, otherwise it is a request/response transaction.

The different types of transactions also differ in their defaults for the tagged values characterizing the initiating and responding activity. The self-explaining tags are *time to perform*, *time to acknowledge receipt*, *time to acknowledge acceptance*, *is authorization required*, and *is non-repudiation required*. *Is non-repudiation of receipt required* and *retry Count* characterize initiating activities only. 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*.

A business transaction might be used in different business environments. The basic flow defined in the business transaction is independent of the business context. Therefore, the name of the transaction itself, of the initiating and reacting activity, of the envelopes, and of the business partners remain unchanged. However, we identify the candidates listed below for variations according to the business context. These types of business context variations are marked by the corresponding number in Figure 1. Additionally, the corresponding values appear as placeholders (*???*):

- $\langle 1 \rangle$ tagged values for envelopes,
- $\langle 2 \rangle$ tagged values for initiating and reacting activities,
- $\langle 3 \rangle$ type of transaction (one out of six).

C. Business transaction in BPSS

The transformation of a UMM business transaction to the BPSS equivalent seems to be straight forward. The transformation of the UMM business transaction 'search product' in 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 circle 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 are the 6 types also used in UMM/RosettaNet. 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 represent the initiating activity (*requesting business activity*) and reacting activity (*responding business activity*). The attributes correspond more or less to the context-sensitive tagged values of the corresponding UMM activity. The following differences exist. 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 initiating and reacting business activity cannot be mapped to BPSS. The *is authorization required* attribute is defined in BPSS, but deprecated, because it cannot be supported by current ebXML business service interfaces. Moreover, BPSS includes the flag *is intelligible check required*

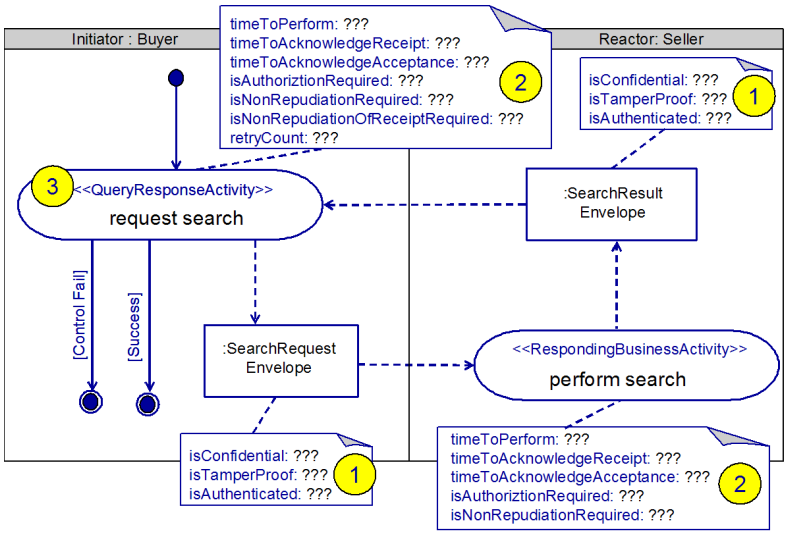


Fig. 1. UMM business transaction "search product".

```
<BusinessTransaction
    nameID="Transaction-001" name="SearchProduct"
3 pattern="???" isGuaranteedDeliveryRequired="true">
        <RequestingBusinessActivity
            nameID="Action-001" name="RequestSearch"
            retryCount="?" isIntelligibleCheckRequired="true" isAuthorizationRequired="???"
            timeToAcknowledgeReceipt="???" timeToAcknowledgeAcceptance="???"
            isNonRepudiationRequired="???" isNonRepudiationReceiptRequired="???"> 2
                <DocumentEnvelope
                    nameID="Envelope-001" name="SearchRequestEnvelope"
                    businessDocument="SearchRequest" businessDocumentIDREF="Document-001"
                    isAuthenticated="???" isConfidential="???" isTamperDetectable="???"/> 1
        </RequestingBusinessActivity>
        <RespondingBusinessActivity
            nameID="Action-002" name="PerformSearch"
            isIntelligibleCheckRequired="true" isAuthorizationRequired="???"
            timeToAcknowledgeReceipt="???" timeToAcknowledgeAcceptance="???" 2
            isNonRepudiationRequired="???" isNonRepudiationReceiptRequired="???">
                <DocumentEnvelope
                    nameID="Envelope-002" name="SearchResponseEnvelope"
                    businessDocument="SearchResponse" businessDocumentIDREF="Document-002"
                    isAuthenticated="???" isConfidential="???" isTamperDetectable="???"/> 1
        </RespondingBusinessActivity>
</BusinessTransaction>
```

to signify whether the ack of receipt is sent immediately upon receipt of the business information (*false*) or after sequence and schema validation (*true*). Since UMM always assumes the latter case, the flag is set to *true* if *time to acknowledge receipt* is set. It should be noted that *time to acknowledge receipt* and *time to acknowledge acceptance* are only specified if the corresponding ack is required, otherwise they are omitted.

The flow of business information is defined in BPSS by defining the envelope (*document envelope*) as child element of the 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. However, this *business document* element (not depicted in this paper) does not define a document structure, but specifies the URI of its schema definition.

D. Context-sensitive mappings from UMM to BPSS

The goal of this subsection is to replace those parts in UMM and BPSS that vary in different business environments and were marked with placeholders. The best way to describe a business environment is by the concept of business context as introduced by ebXML core components [12]. 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 categories and their associated values. In ebXML eight categories have been identified: business process, product classification, industry classification, geopolitical, official constraints, business process role, supporting role, and system capabilities. The context categories are not limited to the ones identified, but we do not recommend the usage of other categories. For demonstration purposes we assume two different business environments, one for ordering books and one for ordering tourism products. In the oversimplified example 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 UMM is based on UML, it seems that OCL [9] should be the constraint language of choice. OCL typically specifies invariant conditions that must hold for the system being modeled. In our case we do not want to specify invariants on the stereotypes of the metamodel (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 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. We use an extended version of the Bachus-Naur-Form (BNF). Note that, in practice, context drivers should refer to accepted code lists in practice, e.g. *industry classification* to the North American Industry Classification System (NAICS) or *product classification* to the United Nations Standard Products and Services Code (UNSPSC). For a better understanding we use human-readable literals in this paper.

```
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 ($\langle 1 \rangle$) concerns the security parameters assigned to the envelopes. Therefore we define an invariant for 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 search result envelope must be confidential, tamper proof, and authenticated. The resulting constraint is depicted in Figure 2.

A transformation of the business transaction including the constraint in Figure 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 response envelope*. The resulting code fragments for the book example and the tourism example are also given in Figure 2. It should be noted that BPSS allows a more sophisticated instantiation of the security parameters: *none*, *transient*, *persistent*, and *transient-and-persistent*. Transient security focuses on the delivery to the receiving message service handler. Persistent security applies as soon as the document leaves the receiving message handler. Transient security is what is considered by UMM. The meaning of *true* for a security parameter in UMM is equivalent to *transient* in BPSS.

The second variation ($\langle 2 \rangle$) based on different business environments concerns the characteristics of the initiating and the reacting business activity. Accordingly, we define an invariant for both initiating and reacting business activity by adding a constraint on each. 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 Figure 3 we present the constraints on the initiating activity *request a search* for our two business environments. We assume that in the book case the search is a simple query on an existing catalog, which anyone can execute. In contrary, in the tourism case the search result is assembled individually and the initiator gets billed for the request. As a consequence, the tagged values differ considerably.

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

Again, the transformation of the initiating business activity including the constraint of Figure 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. *Time to perform* does not exist in BPSS, and *is intelligible check required* is always set *true* according to the UMM semantics. We include the *is authorization required* attribute for reasons of completeness, although ebXML software will currently ignore it.

Finally, there exists a variation on the type of business transaction ($\langle 3 \rangle$). As we described before, the search product transaction is a query against an existing catalog in the book case. The responder already has the information available. By definition this is a query/response transaction. In the tourism case the search result is assembled individually. This is a request/response transaction, since the responder does not have the information available beforehand and pre-editor context validation is necessary.

Although the constraint does apply to the whole business transaction, UMM stereotypes the initiating business activity accordingly. We correct that in our approach by adding a tagged value *business transaction type* to the business transaction. The stereotype of the initiating business activity corresponds to the default business transaction type. This approach is valid, since the six stereotypes for the initiating business activity as defined in the UMM meta model differ only in the name, but provide the same characteristics. This time the constraint statement has a mandatory *else*-clause in the *if*-statement, since a default must be specified for the reasons mentioned above. The constraint for the search product transaction of our example is depicted in Figure 4.

```
context <BusinessTransaction> inv:
```

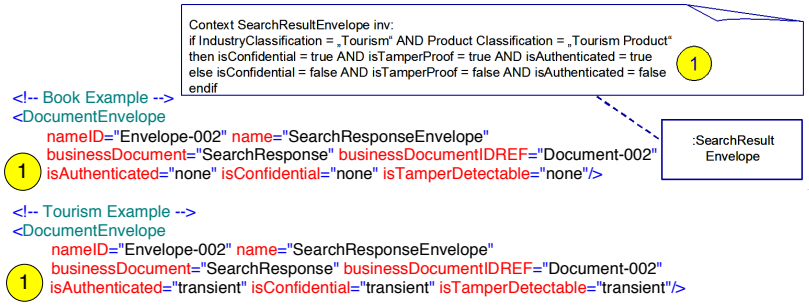


Fig. 2. Business context variation for the search result envelope.

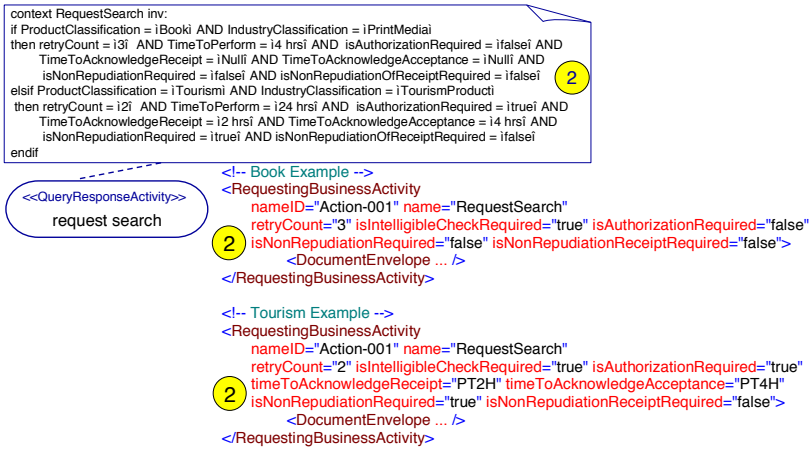


Fig. 3. Business context variation for initiating business activity “request search”.

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

The constraint leads to two different *business transaction elements* for *search product* in BPSS. They differ in the value for the *pattern* attribute according to their transaction type.

IV. Mapping Business Collaborations

A business process is defined as an organized group of related activities that together create customer value [4]. A business collaboration is a special kind of a business process, characterized by the fact that the activities are executed by two or more business partners. In case of two business partners the business collaboration is called a binary collaboration, and otherwise it is a multi-party collaboration. The business transaction as defined in the previous section is a special type of a binary collaboration. It is the basic building block for more complex business collaborations. Consequently, a business collaboration is built from several business transactions. It is important that the business collaboration defines an execution order for the business transactions, i.e. a business collaboration choreography.

A. Business collaboration protocol in UMM

In UMM, the choreography of a business collaboration is defined by an activity graph called business collaboration protocol. Figure 5 shows an extract of the business collaboration protocol for a simple purchase order management. In the current version 12 of UMM all activities of the business collaboration protocol must be stereotyped as business transaction activities. A business transaction activity must be refined by the activity graph of a business transaction. This means that a recursive nesting of business collaboration protocols is not possible in the moment. However, it is likely that future versions of UMM will allow a business collaboration activity that is refined by another business collaboration protocol.

For each business transaction activity the maximum performance time is documented by the *time to perform* property. If the underlying business transaction is not finished by this time, the initiating partner has to send a failure notice. Furthermore, the *is concurrent* property defines whether or not more than one business transaction activity can be open at one time.

As we described in Section III, a business transaction is an atomic unit changing one or more business objects from one state to another. This means, before a business transaction activity is started, the affected business objects must be in a defined state. After the business transaction activity is finished, the affected objects are transferred to a defined state. These states are described in the pre- and post-conditions of a business transaction activity.

The transition from one business transaction activity to another is triggered by two types of events that are defined in UML 1.4: the completion of the previous business transaction and the availability of a business object in a certain

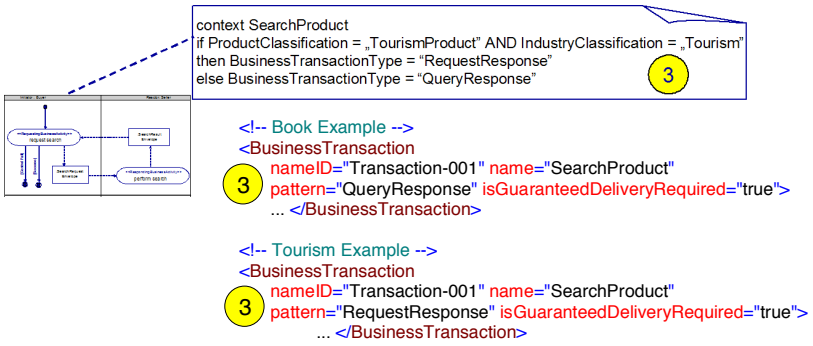


Fig. 4. Business context variation for business transaction "search product".

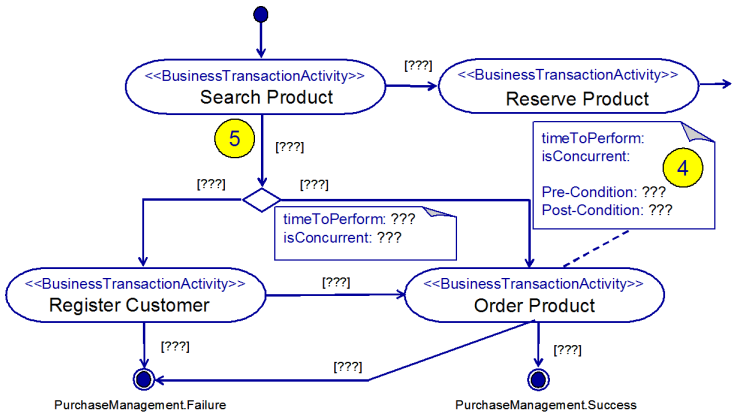


Fig. 5. Extract from business collaboration protocol "purchase management".

state. In addition to business transaction activities a business collaboration protocol includes the pseudo states used in UML activity diagrams: initial state, final state, decisions and merge (both of pseudo-state kind junction), as well as fork and join.

The business collaboration protocol is in principal not restricted to a binary collaboration. Different business transaction activities of the same business collaboration protocol might be performed by a different pair of business partners. However, it must be noted that nested business transaction activities are not allowed due to the transition semantics. Nested transactions are common in multi-party transactions. For example, verify credit between seller and bank is nested in register customer between buyer and seller, because it is started after the registration request, but ended before the registration response. In practice, the business collaboration protocol is used to describe binary collaborations – the primary focus of UMM.

In our approach the basic choreography of the business collaboration is constant. We use business context-sensitive variations for the following elements:

- $\langle 4 \rangle$ tagged values, pre- and post-conditions of business transaction activities,
- $\langle 5 \rangle$ transitions (guards and events).

B. Business collaboration protocol in BPSS

BPSS 1.10 uses binary collaborations only. The schema definition includes elements for multi-party transactions. However, these are marked as deprecated, since they might change considerably in future versions. Hence, we concentrate on binary collaborations. A BPSS binary collaboration might not only include business transaction activities, but also collaboration activities. Collaboration activities are refined by referencing another binary collaboration element. Since UMM business collaboration protocols do not yet use this concept, it is not considered in our transformation. The attributes of the binary collaboration element are most useful in such a recursive structure, so we do not concentrate on them either.

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. For education purposes we do not keep the sequence of the child elements in our explanation. The binary collaboration includes by definition exactly two *role* elements specifying the two business partners involved through the two standard element attributes. States are presented by one ore more *business transaction activity* elements, as well as zero to more elements equivalent to the UML pseudo states *fork*, *join* and *decision*. The transitions are defined by optional *transition* elements as well as the mandatory *start*, *success* and optional *failure* elements.

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. Both must refer to the two roles identified

```
<BinaryCollaboration ... >
        <Role name="customer" nameID="Role-001"/>
        <Role name="seller" nameID="Role-002"/>
        <Start
            nameID="Start-001"
            toBusinessState="SearchProduct" toBusinessStateIDREF="Activity-001"/>
        <BusinessTransactionActivity
            name="SearchProduct" nameID="Activity-001"
            businessTransaction="SearchProduct" businessTransactionIDREF="Transaction-001"
            fromRole="buyer" fromRoleIDREF="Role-001"
            toRole="seller" toRoleIDREF="Role-002"
            isConcurrent="???" timeToPerform="???"                    (4)
            preCondition="???" beginsWhen="some text" endsWhen="some text" postCondition="???"/>
        <BusinessTransactionActivity
            name="RegisterCustomer" nameID="Activity-002" />
        <BusinessTransactionActivity
            name="OrderProduct" nameID="Activity-003" />
        <BusinessTransactionActivity
            name="ReserveProduct" nameID="Activity-004" />
        <Success
            nameID="Success-001"
            fromBusinessState="OrderProduct" fromBusinessStateIDREF="Activity-003"
            conditionGuard="???"/>
        <Failure
            nameID="Failure-001"
            fromBusinessState="OrderProduct" fromBusinessStateIDREF="Activity-003"
            conditionGuard="???"/>
        <Failure nameID="Failure-002"
            fromBusinessState="RegisterCustomer" fromBusinessStateIDREF="Activity-002"
            conditionGuard="???"/>
        <Transition
            nameID="Transition-001-D001"
            fromBusinessState="SearchProduct" fromBusinessStateIDREF="Activity-001"
            toBusinessState="Decision1" toBusinessStateIDREF="Decision-001"
            conditionGuard="???">
                <ConditionExpression
                    expressionLanguage="OCL" expression="???"/>  (5)
        </Transition>
        <Transition nameID="Transition-D001-002" ... > ... </Transition>
        <Transition nameID="Transition-D001-003" ... > ... </Transition>
        <Transition nameID="Transition-002-003" ... > ... </Transition>
        <Transition nameID="Transition-001-004" ... > ... </Transition>
        <Decision name="Decision1" nameID="Decision-001">
            <ConditionExpression
                expressionLanguage="OCL" expression="CustomerInformation.oclInState(Confirmed)"/>
        </Decision>
</BinaryCollaboration>
```

within the binary collaboration. 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* in order to state the condition. Transformation from UMM models will specify this condition by OCL as *expression language*. The *expression* attribute refers to a call of the OCL statement *object.oclInState(state : OclState) : Boolean*, which results in *true* if the business object is in the specified state.

The *transition* element is used for transitions between two of the following types of states: business transaction activity, fork, join, and decision. It references these states in *from business state IDREF* and *to business state IDREF*. Conditions that guard the transition are business context sensitive and 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 similarly to that of the transition element, except that there exists either the *from business state* or the *to business state* attribute.

C. From UMM to BPSS

In this subsection we deal with the transformation of those parts of a business collaboration that are sensitive to the business environment. Candidates are the business transaction activities and the transitions in between. First we have a detailed look on the business transaction activity ($\langle 4 \rangle$). The tagged values *time to perform* and *is concurrent* might be subject to business context variations. In our example in Fig. 6 we assume that the order product activity takes 4 hours at most and that this activity is concurrent. Exceptionally, in the tourism case the maximum time is extended to 12 hours without any concurrency. A constraint statement similar to the ones before seems appropriate.

However, business transaction activities have additional characteristics that change according to the business context: pre- and post-conditions. In our example we assume that a product cannot be ordered right away. A product found in a previous search can be ordered. A reservation of products found is also possible, and a reserved product might be ordered as well. It follows that a pre-condition for *order product* is that the business object *product* is either in state *found* or *reserved*. The tourism case is even more restrictive. One cannot order any product that has not been reserved. Accordingly, the pre-condition requires the business object *product* to be in state *reserved*, the state *found* is not sufficient.

A constraint assigned to a business transaction activity must govern tagged values as well as pre- and post-conditions. A block of tagged values and pre-/post-conditions is assigned either generally to the business transaction activity, or an if-statement is used to assign a block for each different business environment. Within the block we decided to specify the tagged values first. The end of the tagged value assignment within the block is recognized by the keyword *pre:* (or *post:*) that starts the pre-condition (or post-condition) statement. Each pre- and post-condition follows the syntax of the pre-defined OCL statement *object.oclInState(state : OclState) : Boolean*.

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

The constraint given in Figure 6 follows the assumptions we described above. It results again in two different BPSS instantiations. The two instantiations of the business transaction element for order product differ only in the attributes *time to perform*, *is concurrent*, *pre-condition*, and *post-condition*. The values are simply copied from the assignments of the constraint statement.

The last business context variation is directed towards the transitions ($\langle 5 \rangle$) between business transaction activities/pseudo states. A transition requires the completion of the source business activity. Furthermore, the transition is triggered by the availability of a business object in a certain state, which is an end state of the source business activity. In the example in Figure 7, the transition from search product to the decision node (later splitting into order product and register customer) is triggered by the event that the product was found, i.e. the business success of search product. In UML a transition might be guarded. In our approach the guards refer to the business context. Consequently, we do not specify any constraint statement for our last context variation. Yet, we require the

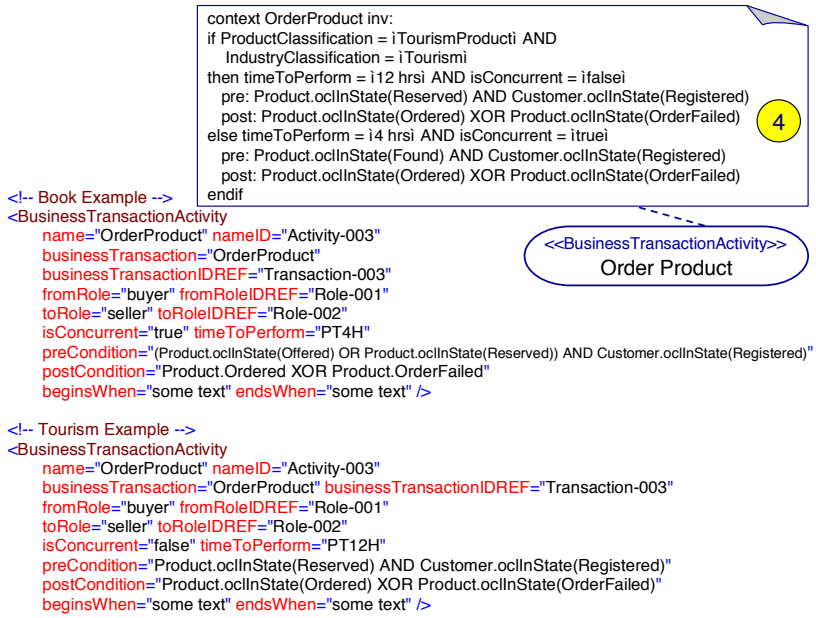


Fig. 6. Business context variation for business transaction activity “order product”.

following: If an event for a transition is specified, it must be in the syntax of *object.oclInState(state : OclState) : Boolean*. Furthermore, if a guard is specified, it must follow the syntax of our business context statement.

In our example, a transition from search product to the decision node that might lead to order products is only valid in the book case. Therefore, the transition is guarded by the corresponding business context statement. This transition does not make sense for the tourism case, since in tourism, products must be reserved prior to ordering. A transformation of the guarded transition will result in the effect that the transition from search product to the decision point (*Transition-001-D001*) will only appear in the BPSS file for the book case, but is not included in the BPSS file for the tourism case. The UMM event is mapped to the condition guard attribute of element transition. The latter is set to *business success*, which is a BPSS pre-defined high level end state for a business transaction. The related object state *found* for *product* is defined in the sub-element *condition expression*.

V. Conclusion

In this paper, we fill the gap between business process definitions as defined by standard organizations and business process definitions executed by software tools. The former are defined on the model level and must fit multiple business environments. The later use a syntax processable by the software tools and are specified in a specific business environment. Since we place our approach in the ebXML environment, we use UMM as a standard for the former category and BPSS as a standard for the later one.

UMM defines much more concepts than needed by BPSS. Only the concepts of business transactions and their choreography into business collaborations are relevant for creating a BPSS instance from a UMM model. In this paper, we have carefully analyzed the semantics of both concepts and we showed how these semantics are represented in a UMM model. In a next step we have demonstrated how BPSS handles these semantics and how the UMM elements are mapped to BPSS elements. Furthermore, we have identified those elements of business transactions and business collaborations that depend on the business context. In UMM, context-specific restrictions must be assigned as constraints to the UMM elements that are defined in multi-context by nature. For this purpose, we have developed a constraint language that follows the spirit of OCL and fits our needs. Finally, we have demonstrated how multi-context UMM elements assigned with context-dependent constraints will lead to different instantiations of context-specific BPSS elements.

In our future work we want to verify whether our approach will also fit into the Web Services environment by transforming multi-context UMM to context-specific BPEL. Furthermore, we will concentrate on the transformation of the business information from multi-context UMM class diagrams to context-specific document types.

References

[1] A. Arkin (2002) *Business Process Modeling Language (Version 1.0)*, November 2002, http://www.bpmi.org/bpml-spec.esp.

[2] T. Andrews *et al.* (2003) *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.

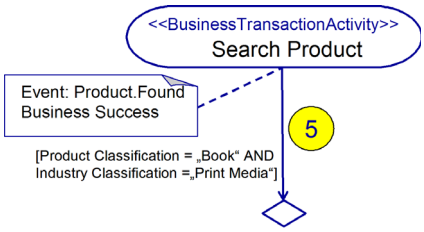
[3] B. Eisenberg and D. Nickull (2001) ebXML Technical Architecture Specification v1.0.4, *ebXML Specifications*, http://www.ebxml.org/specs/ebTA.pdf.

[4] M. Hammer and J. Champy (1993) *Reengineering the Corporation: Manifesto for Business Revolution*. Harper Business.

[5] B. Hofreiter and C. Huemer (2003) Modeling Business Collaborations in Context. *Proceedings of On The Move to Meaningful Internet Systems 2003: OTM 2003 Workshops*. Springer-Verlag, November 2003.

[6] B. Hofreiter, C. Huemer and W. Winiwarter (2004) OCL-Constraints for UMM Business Collaborations. *Proceedings of the 5th International Conference on Electronic Commerce and Web Technologies (EC-Web)*. Springer-Verlag, September 2004.

[7] ISO (1995) Open-edi Reference Model. *ISO/IEC JTC 1/SC30 ISO Standard 14662*.




```
<!-- Book Example -->
<Transition
    nameID="Transition-001-D001"
    fromBusinessState="SearchProduct"
    fromBusinessStateIDREF="Activity-001"
    toBusinessState="Decision1"
    toBusinessStateIDREF="Decision-001"
    conditionGuard="BusinessSuccess">
    <ConditionExpression
        expressionLanguage="OCL"
        expression="Product.oclInState(Found)"/>
</Transition>

<!-- Tourism Example -->
<!-- No Transition-001-D001 exists -->
```

Fig. 7. Business context-sensitive guards on transitions.

[8] F. Leymann, D. Roller and M.-T. Schmidt (2002) Web Services and Business Process Management. *IBM Systems Journal*, **41**(2).
[9] OMG (2003) *Object Constraint Language Specification*, http://www.omg.org/cgi-bin/doc?formal/03-03-13.
[10] RosettaNet (2002) *RosettaNet Implementation Framework: Core Specification, V02.00.01*, March 2002, http://www.rosettanet.org/rnif.
[11] UN/CEFACT (2003) *ebXML – Business Process Specification Schema v1.10*, October 2003, http://www.untmg.org/downloads/General/approved/ebBPSS-v1pt10.zip.
[12] UN/CEFACT (2003) *Core Components Technical Specification V2.01*, November 2003, http://www.untmg.org/downloads/General/approved/CEFACT-CCTS-Version-2pt01.zip.
[13] UN/CEFACT (2003) *UMM Meta Model, Revision 12*, January 2003, http://www.untmg.org/downloads/General/approved/UMM-MM-V20030117.zip.
[14] UN/CEFACT (2003) *UMM User Guide, Revision 12*, September 2003, http://www.untmg.org/downloads/General/approved/UMM-UG-V20030922.zip.

Birgit Hofreiter Birgit Hofreiter received her MSc in Business Informatics from the Vienna University of Technology, Austria. Since then she is employed as a research assistant at the Institute of Distributed and Multimedia Systems at the University of Vienna. Her research interests cover the fields E-Commerce (especially B2B), Business Process Engineering and the Semantic Web. Birgit Hofreiter participated in the ebXML Initiative and contributes to UN/CEFACT. Besides her research work she is currently giving lectures on Electronic Commerce, Modeling Techniques and Methods, and Information Systems.

Christian Huemer Christian Huemer is Associate Professor at the Institute of Distributed and Multimedia Systems at the University of Vienna, Austria. He finished his master in business informatics at Vienna University of Technology in 1993. In 1997 he received a doctors degree from the University of Vienna. In 2002 the University of Vienna conferred the Venia Docendi in business informatics on Christian Huemer. His research interests are methodologies for modelling e-business transactions as well as meta data support for electronic business and EDI environments. Currently, he is Vice Chair of UN/CEFACT's Techniques and Methodologies Group (TMG). For a long time he has been a contributer to UN/CEFACT's Modelling Methodology (UMM). Furthermore, he has been a member of the ebXML initiative from its beginning and organized the last ebXML Meeting in May 2001. He was involved in several international and national projects.

Werner Winiwarter Prof. Dr. Werner Winiwarter holds a tenured position at the Institute of Scientific Computing, University of Vienna. He received an MS degree in 1990, an MA degree in 1992, and a PhD in 1995, all from the University of Vienna. His main research interest is human language technology. In addition, Prof. Winiwarter also works on data mining, and machine learning, Semantic Web, information retrieval and filtering, electronic business, digital libraries, and education systems.