ebXML Business Processes - Defined both in UMM and BPSS

Birgit Hofreiter[1] and Christian Huemer[1,2]

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

[2] Department of Information Systems, University Duisburg-Essen
Universitätsstrasse 9, 45141 Essen, Germany
huemer@wi-inf.uni-essen.de

Abstract. The ebXML framework consists of eight specifications for conducting e-Business. The loosely coupled specifications span over the topics of messaging, registries, profiles & agreements, business processes, and core (data) components. The choreography of business processes is defined by instances of the business processes specification schema (BPSS). The BPSS is defined as an XML schema used by business systems to support the execution of business collaborations. It is based on concepts introduced by the UN/CEFACT Modelling Methodology (UMM), or the UMM meta model to be more specific. Thus, BPSS provides the bridge between e-business process modeling and specification of e-business software components. In this paper we show how an ebXML business process is represented in both UMM and BPSS.

1 Introduction

In November 1999 UN/CEFACT and OASIS started the ebXML initiative. The vision of ebXML is to create an electronic marketplace, where businesses can find each other, agree to become trading partners and conduct business. All operations are performed automatically by exchanging XML documents. In order to support the needs of small and medium enterprises (SMEs), ebXML envisions that software industries will deliver commercial off-the-shelf software (COTS) for Business-to-Business (B2B) scenarios to the SMEs. This goal is expressed in the ebXML scenario between a large corporation (Company A) and a SME (Company B) as illustrated in Figure 1. The scenario is described in the ebXML technical architecture specification [UO01].

Company A requests business details from the ebXML registry (step 1) and decides to build its own ebXML-compliant application. Company A submits its own business profile information to the ebXML registry. The business profile submitted to the ebXML registry describes the company's ebXML capabilities and constraints, as well as its supported business scenarios. Company B, which uses an ebXML-compliant shrink-wrapped application, discovers the business scenarios supported by Company A in the registry (step 4). Company B sends a request to Company A stating that they would like to engage in a business scenario (step 5). Before engaging in the scenario, company B submits a proposed business arrangement directly to Company A's ebXML-compliant software interface. The proposed business arrangement outlines the mutually agreed upon business scenarios and specific agreements. Company A then accepts the business agreement. Company A and B are now ready to engage in e-business using ebXML (step 6).

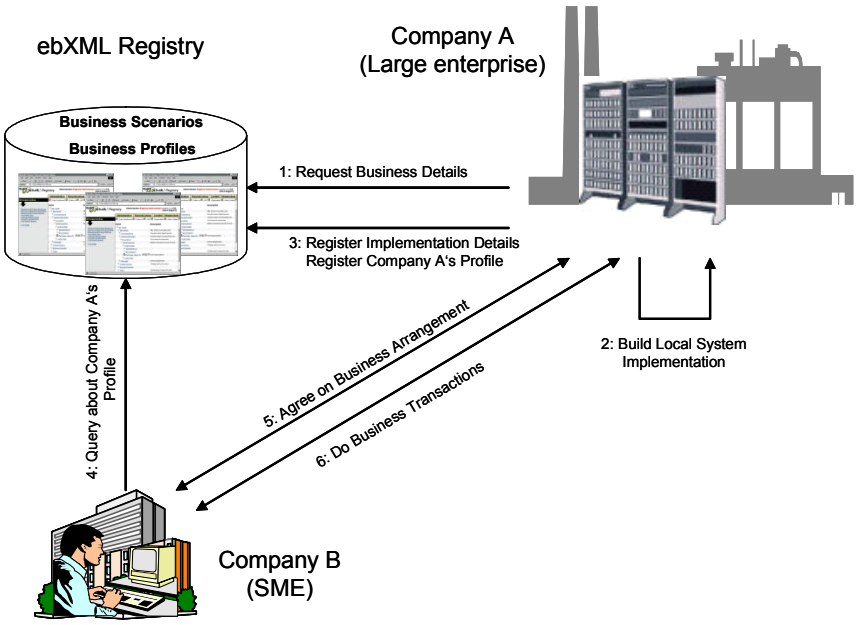

Figure 1. ebXML Scenario

In order to support this secenario ebXML offers a modular suite of specifications. These specifications provide a standard method to exchange business messages, conduct trading relationships, communicate data in common terms and define/register business processes [HHK02]. The current set of specification comprises the ebXML requirements, technical architecture, messaging service, registry services specification, registry information model, collaboration protocol profile and agreement specification, business process specification schema, and core components.

In this paper we concentrate on the ebXML business process specification schema (BPSS) in its current version 1.10 [UN03c]. It is defined as an XML schema. The BPSS is used to describe standard B2B business processes by defining a choreography of activites amongst business partners. It usally refers to standard business documents that are exchanged in the business process. A partner's profile references the BPSS of a supported process and the role(s) therein. Similarily, a business partners' agreement references the BPSS of a business process in which the business partners will collaborate.

Creating a BPSS does not require a specific process modeling methodology. However, BPSS is based on concepts introduced by UN/CEFACT's Modelling Methodology (UMM). UMM is a methodology for defining the business aspects of a B2B collaboration. It is based on UML [BJR98]. BPSS can be considered as an XML representation of a subset of UMM's meta model. In Section 2 we define an ebXML business process by the means of UMM. The graphical UML syntax helps to quickly understand the basic concepts. In Section 3 we show the equivalent business process in the XML syntax of BPSS. We show how the UMM concepts are mapped to BPSS and point out the rare cases where BPSS extends the UMM concepts. We conclude with a summary in Section 4.

2 UN/CEFACT's Modelling Methodology

ebXML does not require any specific modeling language or modeling methodology. However, the architecture specification recommends that if implementers and users decide to apply business process modeling, they shall use UMM [UO01]. The UMM concentrates on the business semantics of B2B partnerships. It captures the commitments that are made by business partners when agreeing to a certain type of business processes. These commitments are reflected in the resulting orchestration of the business process involving information exchanges. UMM defines a procedure [UN03b] that describes the necessary steps to create business collaboration models. These business collaboration models are independent of the technology (e.g. ebXML) used to implement them. The term "business collaboration" is used in UMM for a business process involving two or more business partners to accomplish a common business goal.

In addition to the procedure, the reference ontologies, and the patterns, UMM delivers a meta model [UN03a]. The UMM meta model puts the UML meta model into a small corset defining those diagram types that are specific for B2B. The most important diagram types are presented in the subsections below. The UMM procedure [UN03b] leads to business collaboration models that are valid instances of the UMM meta model. These business collaboration models contain more information than what is required for configuring ebXML compliant software, but not less than that. BPSS is based on the UMM meta model. It uses only those concepts that are important for the configuration of the ebXML software. Thus, BPSS is a subset of the UMM meta model, but expressed as XML schema. One might use any methodology to create an BPSS instance. However, UMM guarantees a consistent way for developing a business collaboration model that is a superset of a BPSS instance.

Beforehand, we introduce the UMM by means of a simple example. In the next section the resulting business collaboration model is mapped to the BPSS equivalent. It is not our goal to introduce UMM in all the details. We limit ourselves to those features that help to understand the concepts of BPSS.

The UMM procedure as well as the UMM meta model consists of 4 views in order to describe business collaboration models. 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. In other words, the BRV focuses on the economics of a system. Thirdly, the **Business Transaction View (BTV)** presents the view of the business process analyst. It defines the orchestration 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. The business service view does not add any new information. Its artefacts are automatically created from the information gained in the previous process steps. The BSV artefacts are not mapped to the BPSS. Therefore, we do not detail the BSV in the following subsections.

2.1 Business Domain View (BDV)

The first workflow of UMM is used to gather existing knowledge. It identifies the business processes in the domain of the business problems that are important to stakeholders. It is important at this stage that business processes are not constructed, but discovered. Stakeholders might describe intra-organizational as well as inter-organizational business processes. Both types are recorded. However, the description concentrates on so-called business interface tasks, where an organization communicates with its partners. All the discovered business processes are classified according to a pre-defined classification schema. The final result of the business domain view allows a business process analyst to find opportunities for business collaborations that are constructed in the following view.

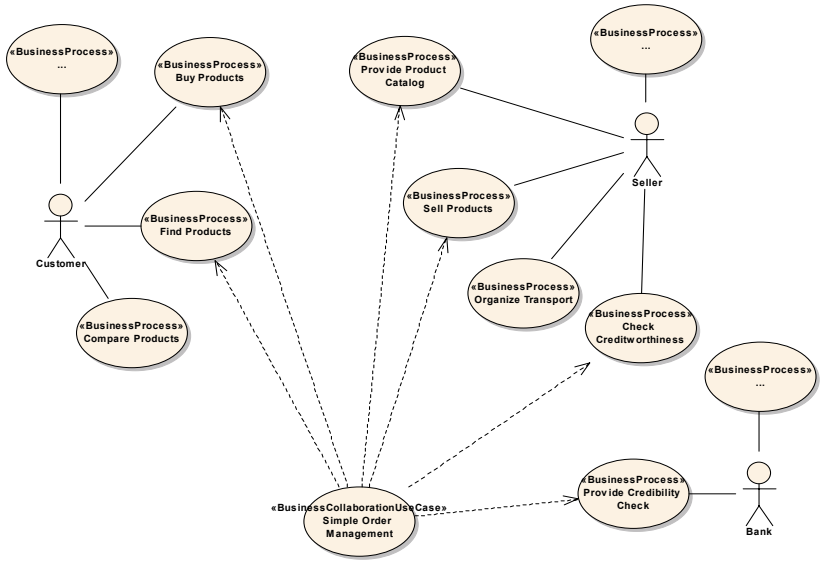

Figure 2. Business Processes to be Considered in Simple Order Mangement

To demonstrate UMM and BPSS we use the example of a simple order management process (see also [HH03, HHN04]). Note that this example does not include all the complexity that might be involved in an order management process. The simplified process still allows us to show the main concepts of UMM. In the first workflow the stakeholders in the simple order management process are interviewed. Customers, sellers, and banks (amongst others not included in the example) describe their business processes that are important in the domain under consideration. These business processes are documented by UML use cases. The resulting use case diagram is depicted in Figure 2. Each actor (customer, seller, and bank) is associated with those business processes that are described by the respective role. The details of each business process are described by completing a worksheet. UMM provides a worksheet type designed to capture all the relevant aspects of a business process. Due to space limitation we do not further concentrate on these worksheets.

2.2 Business Requirements View (BRV)

The goal of the business requirements view is to identify possible business collaborations in the considered domain and to detail the requirements of these collaborations. Business collaborations span over multiple business processes discovered in the previous workflow. Thus, a use case for a business collaboration must consider the views of different stakeholders. The description of the use case must present an harmonized view on the business collaboration being developed.

In our example we identify a *simple order management* as a possible business collaboration between a customer, a seller, and a bank. This *simple order management* depends on business processes described in the previous workflow. The simple order management is described again by a use case together with a worksheet especially designed for business collaboration use cases. We show the *simple order management* use case already at the bottom of Figure 2. This allows us to show the dependency of the business collaboration use case on the business processes of the BDV.

Possible business collaborations identified in the BRV are multiparty as well as binary business collaborations. Binary collaborations are between two business partners only. More than two business partners participate in a multiparty collaboration. Business collaborations 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 a business transaction.

Use cases and special worksheets document the requirements of multiparty business collaborations, binary business collaborations and business transactions. So-called business collaboration protocol use cases define the needs of both multiparty and binary business collaborations. The term “protocol” appears in the stereotype to express that a complex protocol is needed to choreograph the activities of the collaboration. Less surprisingly, a business transaction use case describes the requirements of a business transaction. A business collaboration protocol use case usually requires other business collaborations and/or business transactions to be included as part of it. This fact is denoted by “include”-relationships between the respective use cases (c.f. Figure 4). In the BRV all the business collaboration use cases are decomposed recursively until the lowest level, which is a business transaction use case, is reached.

The *simple order management* collaboration is a multiparty collaboration since three business partners (customer, seller, bank) are participating in this collaboration. BPSS allows multiparty collaborations only if they are synthesized from two or more binary collaborations. A multiparty collaboration is not able to reflect dependencies between intermediate states of different business collaborations. For a better understanding we anticipate what we will see later on: The purchasing process between a customer and a seller involves multiple choreographed steps. Somewhere in the course of this process a customer requests registration. The seller might check the credibility of the customer with the customer’s bank before responding to the customer. This means that some activities of one binary collaboration are performed during the course of the other. By strictly separating the two binary collaborations we loose this choreography information.

UMM is a little bit less restrictive than BPSS. A choreography for a collaboration might include more than two parties. Important in UMM is that a business collaboration is finally decomposed into business transactions, which are by default binary. These business transactions do not overlap. This means that UMM does not support nested business transactions. Note, that the UMM reference guide (currently only available as Chapter 8 of the old UMM Revision 10) mentions that business transactions can be nested - however the UMM meta model does not support it. Coming back to the example above, the credibility check is nested in the registration transaction. This cannot be expressed according to the UMM meta model. Nesting business transactions usually appear in multiparty collaborations. Only, binary collaborations are choreographed without the need for nesting transactions. In our example, we split the multiparty collaboration simple order management into two binary collaborations: (1) the *simple purchase management* performed by customer and seller and (2) the *credibility check* performed by seller and bank. Figure 3 shows this decomposition.

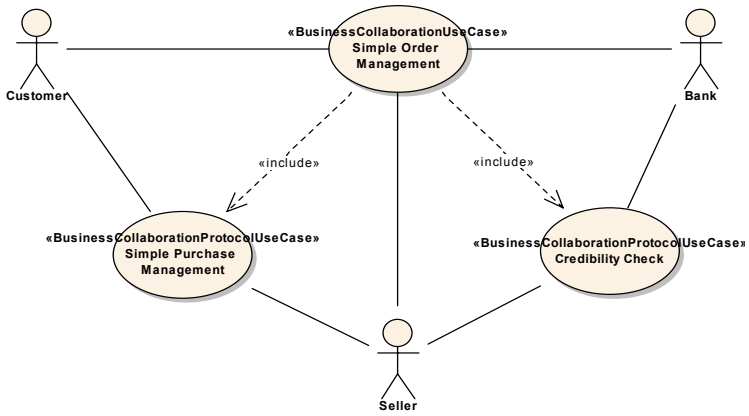

Figure 3. Decomposing Multiparty Collaborations into Binary Collaborations

On first glance one might consider the weak support of multiparty collaborations as a weak point of UMM. It is necessary to understand that UMM is especially designed for modelling B2B partnerships focusing on the aspects regarding the making of business decisions and commitments among the business partners. Therefore, a UMM-compliant business collaboration model designs all the steps necessary to fulfill a contract. The business collaboration considers all the parties fulfilling the contract. Most of the contracts in real business life are between two parties. Returning to the example above, a contract is made between the customer and the seller. This contract is independent of whether the seller decides to check the credibility or not. UMM does not standardize any internal processes or decisions.

The binary collaboration *simple purchase management* includes - according to the worksheet - the use cases of *register customer*, *search product*, and *order products* among others not detailed here. Figure 4 shows the decomposition. The requirements of these use cases are documented by corresponding worksheets. By completing the worksheet it is easy to detect that each of these use cases can be realized by a business transaction. Consequently, the use cases are stereotyped as business transaction use cases. It follows, that the bottom layer is reached and no more decomposition is necessary.

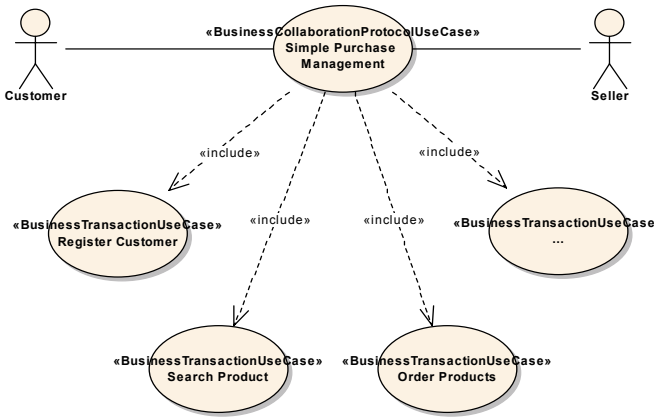

Figure 4. Decomposing a Binary Collaboration into Business Transactions

According to UMM, a business collaboration is best designed by a choreography of business state changes. Thus, it is important to analyze the effects of activities on the business state of the collaboration or, better, on the business states of business entities, which have a life-cycle during the business collaboration. Figure 5 depicts the business state changes in the business transaction *register customer*. It defines preconditions, post-conditions and inter-mediate states of the *customer information* during the transaction.

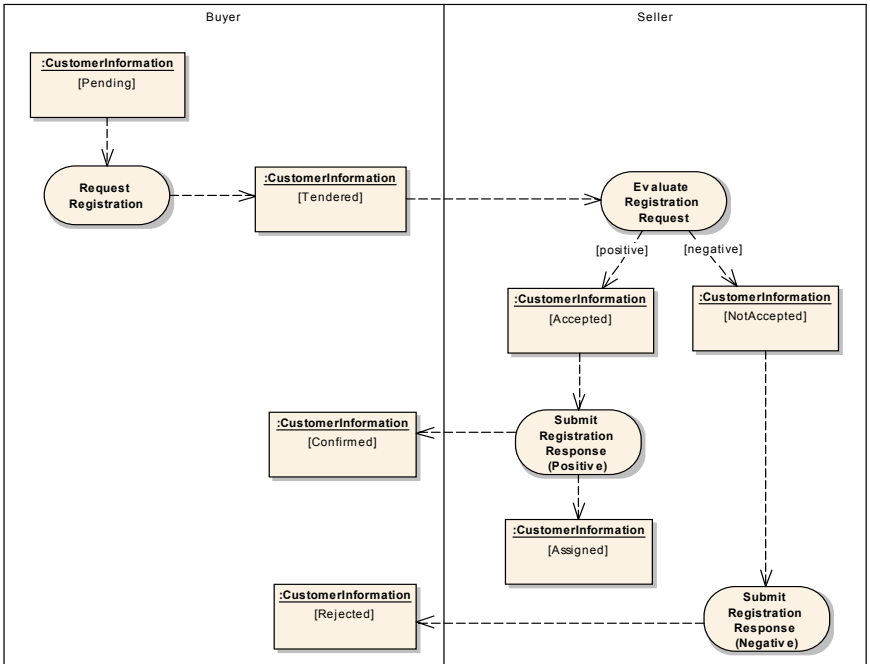

Figure 5. Business Entity States for Customer Information

2.3 Business Transaction View (BTV)

The BTV defines the choreography for the business collaboration as well as the structure of the business information exchanges. The business transaction view covers three main types of artefacts: the business collaboration protocol, the business transaction diagram and the class diagram on business information structures.

The first step of the BTV models the business collaboration according to the corresponding use case description. The resulting activity graph is called business collaboration protocol. An extract of the business collaboration protocol for the binary collaboration *simple purchase management* is presented in Figure 6. The use case diagram of the BRV in Figure 4 defines that simple purchase management includes *register customer*, *search products* and *order products* among others. All the included use cases are mapped in the BTV to activities of the business collaboration protocol. Business transaction use cases are mapped to business transaction activities and business collaboration protocol use cases to collaboration activities. Each business transaction activity is further detailed by the activity graph for a business transaction (see further below). A collaboration activity is refined by another business collaboration protocol. However, UMM mentions the concept of collaboration activity, but does not support it by the current meta model. It follows that a collaboration activity must be recursively replaced by its detailing business collaboration protocol. Finally, a business collaboration protocol is built by business transactions activities only. We think that the concept of collaboration activities - which is supported by BPSS - is certainly useful and should be considered in future revisions of the UMM meta model.

For each business transaction activity the maximum performance time is documented by the *timeToPerform* property. If this time is exceeded, the initiating partner has to send a failure notice. Furthermore, the *isConcurrent* property defines whether or not more than

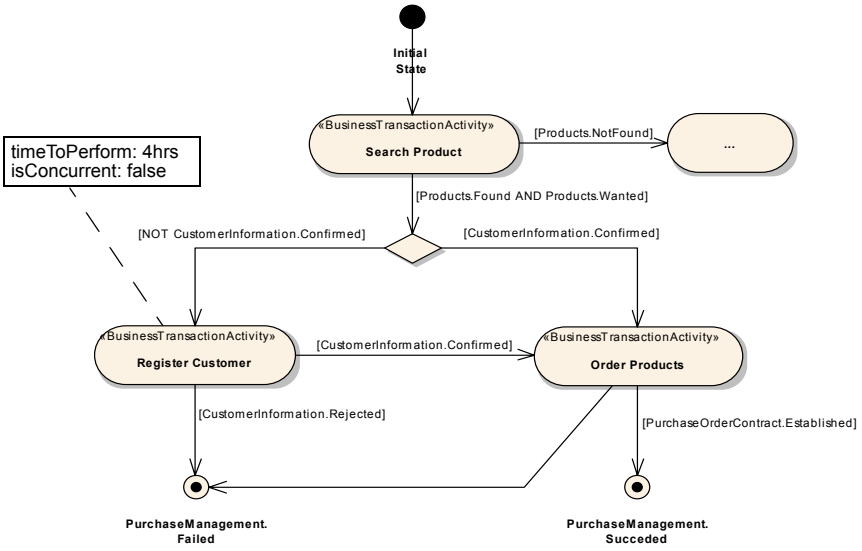

Figure 6. Business Collaboration Protocol for Simple Purchase Management

one business transaction activity can be open at one time. In our example *register customer* is not concurrent and most be performed in 4 hours at most.

The business collaboration protocol defines the choreography amongst the business transaction activities. The transitions between the activities are guarded by business states of business entities. For example, a transition from *search product* to *order product* requires that products were found and that these products are wanted (by the customer). Furthermore, the customer must be registered at the seller, i.e. she must have a confirmed customer information.

The next step in the BTV is to detail each business transaction activity by its own activity graph. This graph defines the choreography of a business transaction. The activity graph of a business transaction is always composed of two business actions, a requesting business activity performed by the initiator and a responding business activity performed by the other business partner. We stick here to the UMM terms, although initiating and reacting business activities would be better terms, since not each transaction is a request / response one. In the UML notation, a business action is assigned to the swimlane of the respective business partner. In a one-way transaction business information is exchanged only from the requesting business activity to the responding business activity. In case of two-way transaction the responding business activity returns business information to the requesting business activity. The exchange of business information is shown by object flows. Figure 7 shows the business transaction *register customer*.

In UMM we distinguish two different types of one-way transactions. If the business information sent is a formal non-reputable notification, the transaction is called notification. Otherwise the transaction is known as information distribution. Furthermore, there exist four different types of two-way transactions. If the responder already has the information

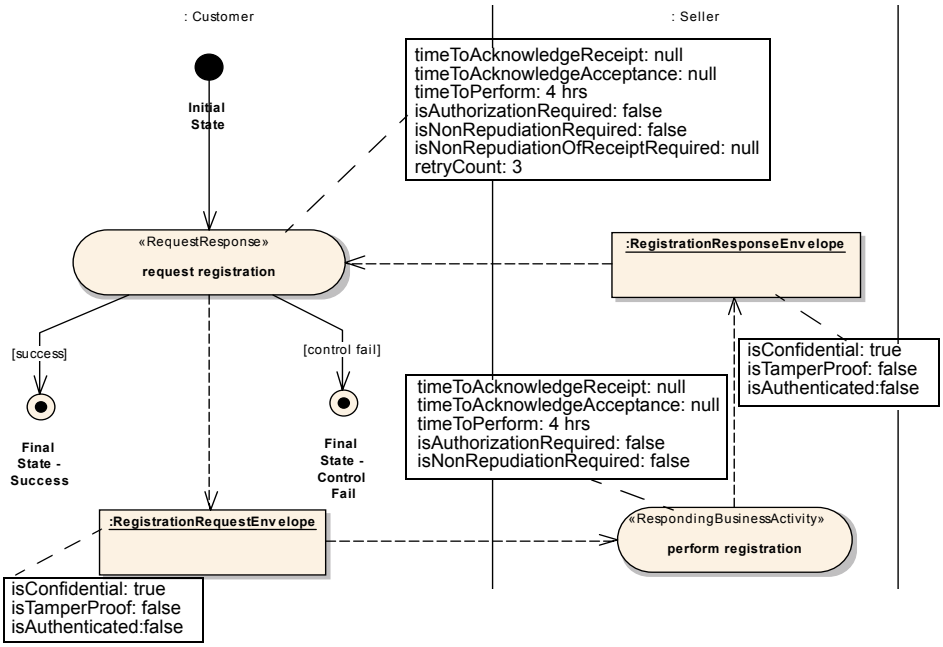

Figure 7. Business Transaction for Register Customer

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 a request/confirm one. 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. In case of a "yes" it is a commercial transaction and a request/response transaction otherwise. These types of business transactions cover all known legally binding interactions between two decision making applications as defined in Open-edi [ISO95]. They have proven to be useful in RosettaNet. In the UML notation the requesting business activity is stereotyped according to the transaction type.

The different types of business transaction patterns differ in the default values for the parameters that characterize the activities in the business transaction: *timeToAcknowledgeReceipt*, *timeToAcknowledgeAcceptance*, *timeToPerform*, *isAuthorizationRequired*, and *isNonRepudiationRequired*. The values for *isNonRepudiationOfReceiptRequired* and for *retryCount* are only defined for the requesting business activity. Note, an acknowledge of receipt is sent after grammar validation, sequence validation, and schema validation. An acknowledge of acceptance is sent after an additional content validation. Retry count is the number of retries in case of control failures.

The object flow states of a business transaction refer to instances of business information envelopes. The business information envelope and the included business information is modeled in a class diagram. The business information is commonly called a business document. Since UMM is not a document-centric approach and only exchanges information necessary for a business state change, we stick to the term business information.

In order to guarantee reusability, the business information must be built by common building blocks. Unfortunately, the current version of UMM does not reflect this requirement. The meta model only defines that the business information exchanged is built by recursively structured information entities. Recently it was agreed that the business information should be based on ebXML core components. A core components is defined as "a building block that contains pieces of business information that belong to a single concept. Core components are characterized by the fact that they appear in many different circumstances of business information and in many different areas of business." [UN03d] Currently, UN/CEFACT is building a library of core components which will become the richest and cross-industry harmonized source for assembling business information.

In order to model the class diagram for a business information exchange, one must determine the business entities that are effected by the transaction. Each business entity is described by the information needed to change its business state. This information is built by re-using core components of the library. Thus, one must select suitable core components from the library and customize them to the needs of the business transaction. Customizing means setting the core components into the context of the business transaction. This is when the core component becomes a so called business information entity. In the resulting class diagram a basic business information entity (BBIE) is represented by an attribute and an aggregate business information entity by a class or a group of classes.

Figure 8 presents a class diagram for the registration request. The registration request envelope includes the business information, i.e. the registration request. The registration request aggregates the header details and the affected business entities. In this case the only business entity is the customer information. The customer information must include

all the information necessary to change its state from “tendered” to “accepted” and finally “confirmed”. Therefore existing core component types are set into the context of our registration transaction. Setting into context means e.g. selecting only those attributes from the more than 20 attributes that are assigned to the core component address. Due to space limitation we do not explain this process in detail but refer to the core components specification [UN03d].

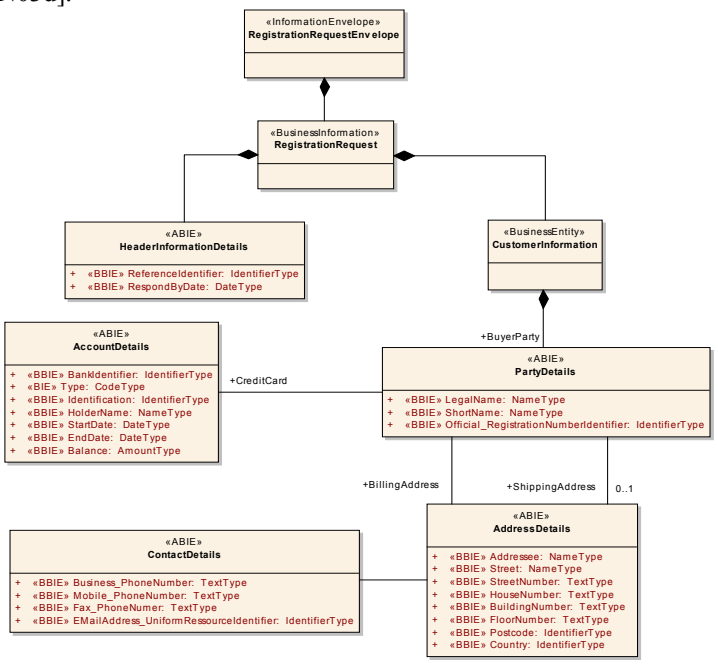

Figure 8. Class Diagram for Registration Request

3 From UMM to BPSS

In this section we describe how the UMM model developed in the previous section is equally represented in a BPSS instance. As mentioned earlier BPSS is based on a subset of the UMM meta model. This subset is more or less identical to the Business Transaction View (BTV). Thus, BPSS might be viewed as an “XML-ification” of the BTV artefacts: business collaboration protocol and business transaction with references to exchanged documents. The UMM is an approach intended to specify business collaborations from top down, re-using existing lower level content as much as possible. In BPSS it does not matter whether an top-down or bottom-up approach is used. For educational purposes we use a bottom up approach to describe the concepts of BPSS.

3.1 Business Documents / Business Information

In BPSS, the business information exchanged in a business transaction is called a business document. Thus the UMM concept of business information must be mapped to an

BPSS equivalent. BPSS does not by itself support the definition of business document types, nor does any other ebXML specification. It is assumed that the various business document standard organizations will define the rules on how to map the business information entities to their specific document types.

A BPSS instance points to the resulting business document types. If the document type is XML-based, BPSS will use the business document element to point to an external schema. We assume that this is the case in our example below. However, it is possible to define documents of any other structure, e.g. UN/EDIFACT, or completely unstructured documents as attachments. The following two code fragments show the XML schema for the attribute group *name* used in *BusinessDocument* and many other BPSS elements, and the XML schema for *BusinessDocument*. Each defined business document carries a globally unique Id (GUID) in the *nameID* attribute. The logical name of the business document, which corresponds to the class name for business information in UMM, is defined in the *name* attribute. Either *specificationLocation* or *specificationID* are used to point to the external schema.

```
<xsd:attributeGroup name="name">
    <xsd:attribute name="name" type="xsd:string" use="optional"/>
    <xsd:attribute name="nameID" use="required">
        <xsd:simpleType> <xsd:restriction base="GUID"/> </xsd:simpleType>
    </xsd:attribute>
</xsd:attributeGroup>


<xsd:element name="BusinessDocument">
    <xsd:complexType>
        <snip/>
        <xsd:attributeGroup ref="name"/>
        <xsd:attribute name="specificationLocation" type="xsd:anyURI"/>
        <xsd:attribute name="specificationID" type="xsd:anyURI"/>
        <snip/>
    </xsd:complexType>
</xsd:element>
```

The valid instances for pointing to external XML schemas that define the document types for registration request and registration response are the following:

```
<BusinessDocument nameID="BPSS-XML4BPM-Document-001" name="RegistrationRequest"
specificationLocation="http://www.xml4bpm.org/RegistrationRequest.xsd"/>
<BusinessDocument nameID="BPSS-XML4BPM-Document-002" name="RegistrationResponse"
specificationLocation="http://www.xml4bpm.org/RegistrationResponse.xsd"/>
```

3.2 Business Transactions

BPSS uses the concept of business transactions more or less identically to UMM. This means a business transaction is built by two business actions - the requesting business activity and the responding business activity. The request document flow is mandatory and the responding one is optional. We recommend to read this subsection in conjunction with Figure 7 showing a UMM business transaction.

The information exchanged in a UMM business transaction is represented by an object flow state. This object flow state is an instance of a business information envelope which covers the business information. For this purpose, BPSS defines the element *DocumentEnvelope*. The envelope has a GUID (attribute *nameID*) and a logical name (attribute *name*). Its attribute *businessDocumentIDREF* references one of the business documents

described above. Furthermore, the attribute *businessDocument* contains the logical name of this document. In the example further below the business document envelope *RegistrationRequestEnvelope* is created with its unique Id (*BPSS-XML4BPM-Envelope-000001*). This envelope references the business document *RegistrationRequest* (*BPSS-XML4BPM-Document-000001*).

Both BPSS and UMM define the security parameters *isAuthenticated*, *isConfidential*, and *isTamperDetectabel* (in UMM: *isTamperProof*) for the information exchanged. In UMM these parameters are booleans. BPSS uses a more sophisticated differentiation with four possible values: *none*, *transient*, *persistent*, and *transient-and-persistent*. Transient security focuses on the delivery to the receiving message service handler. Persistence security applies as soon as the document leaves the receiving message handler. Transient security is what is considered by UMM. Therefore, a UMM *true* for a security parameter maps to *transient* in BPSS. Persistence security cannot be mapped automatically. In UMM, each subpart within a business information envelope might have its own security parameters. In BPSS the security parameters are the same for the envelope and its content. By mapping UMM to BPSS the highest security identified for a subpart must be set for the whole envelope. In our UMM example we already used the security parameters only on the envelope level. Therefore, the code fragment for the *registration request envelope* looks as follows (c.f. Figure 7): *isConfidential* is set to *transient*, whereas *isAuthenticated* and *isTamperDetectable* are set to *none*.

```
<xsd:element name="DocumentEnvelope">
    <xsd:complexType>
        <snip/>
        <xsd:attributeGroup ref="name"/>
        <xsd:attribute name="businessDocument" type="xsd:string" use="required"/>
        <xsd:attribute name="businessDocumentIDREF" type="GUIDREF"/>
        <xsd:attribute name="isPositiveResponse" type="xsd:boolean"/>
        <xsd:attributeGroup ref="documentSecurity"/>
    </xsd:complexType>
    <snip/>
</xsd:element>
<xsd:attributeGroup name="documentSecurity">
    <xsd:attribute name="isAuthenticated">
        <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
                <xsd:enumeration value="none"/> <xsd:enumeration value="transient"/>
                <xsd:enumeration value="persistent"/><xsd:enumeration value="transient-and-persistent"/>
            </xsd:restriction>
        </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="isConfidential">
        <xsd:simpleType> <snip>identical to isAuthenticated</snip></xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="isTamperDetectable">
        <xsd:simpleType> <snip>identical to isAuthenticated</snip></xsd:simpleType>
    </xsd:attribute>
</xsd:attributeGroup>

<DocumentEnvelope nameID="BPSS-XML4BPM-Envelope-001" name="RegistrationRequestEnvelope"
businessDocument="RegistrationRequest" businessDocumentIDREF="BPSS-XML4BPM-Document-001"
isAuthenticated="none" isConfidential="transient" isTamperDetectable="none"/>
```

Next, we consider the requesting activity and the responding activity. Both are subtypes of business action in UMM and BPSS. Each business action has a GUID and a name. In BPSS, the requesting business activity includes the child element document envelope created by the requesting business activity itself. Since the requesting flow is mandatory, the child element is mandatory as well. The responding activity includes an optional document envelope. Note, there is a bug in the BPSS 1.10, since a responding business activity

might output even more document envelopes. We corrected this in the code fragment below. In BPSS there is no explicit link between a business document envelop and the business action that receives this input. The receiving business action is always the one that does not create the business document envelope.

The most significant difference to UMM is the fact, that BPSS does not assign any roles to the business actions on the transaction level. In addition, the parameters characterizing the business actions differ in UMM and BPSS. *timeToAcknowledgeReceipt* and *timeToAcknowledgeAcceptance* are used similar. A value is assigned to each of these attributes, if the acks are required. In UMM an acknowledgment of receipt is sent after schema validation. The BPSS equivalent requires the flag *isIntelligibleCheck* to be set, otherwise the ack is sent after receiving the information without any validation. In BPSS there is no attribute like *timeToPerform*, since the only significant time is the one for the overall transaction that is defined in the business transaction activity. Both *isNonRepudiationRequired* and *isNonRepudiationOfReceiptRequired* are attributes of both business action types - in UMM the latter does not exist for the responding business activity. The *retryCount* for the requesting business activity is also used as in UMM. The *isAuthorizationRequired* attribute is defined, but deprecated, because it cannot be supported by current ebXML business service interfaces.

In our example (c.f. Figure 7) the requesting business activity is *request registration*. Both non-repudiation requirements do not apply. The retry count is set to 3. Time values for the acknowledgments must not be specified, since no acks are sent. *Request registration* outputs the *registration request envelope*. The responding business activity is *perform registration*. Non-repudiation is not required. The acks attributes must be omitted as well. Since both acknowledgments do not require acknowledgments of receipt, the *isIntelligibleCheckRequired* parameter is not useful.

```
<xsd:complexType name="BusinessAction">
    <snip/>
    <xsd:attributeGroup ref="name"/>
    <xsd:attribute name="isAuthorizationRequired" type="xsd:boolean" default="false">
        <xsd:annotation> <xsd:documentation>deprecated</xsd:documentation> </xsd:annotation>
    </xsd:attribute>
    <xsd:attribute name="isIntelligibleCheckRequired" type="xsd:boolean" default="false"/>
    <xsd:attribute name="isNonRepudiationRequired" type="xsd:boolean" default="false"/>
    <xsd:attribute name="isNonRepudiationReceiptRequired" type="xsd:boolean" default="false"/>
    <xsd:attribute name="timeToAcknowledgeReceipt" type="xsd:duration"/>
    <xsd:attribute name="timeToAcknowledgeAcceptance" type="xsd:duration"/>
</xsd:complexType>
<xsd:element name="RequestingBusinessActivity">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="BusinessAction">
                <xsd:sequence><xsd:element ref="DocumentEnvelope"/></xsd:sequence>
                <xsd:attribute name="retryCount" type="xsd:int"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>
<xsd:element name="RespondingBusinessActivity">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="BusinessAction">
                <xsd:sequence>
                    <xsd:element ref="DocumentEnvelope" minOccurs="0" maxOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>
```

```
<RequestingBusinessActivity nameID="BPSS-XML4BPM-Action-001" name="RequestRegistration" retryCount="3"
isNonRepudiationRequired="false" isNonRepudiationReceiptRequired="false">
    <DocumentEnvelope nameID="BPSS-XML4BPM-Envelope-001" name="RegistrationRequestEnvelope" ... />
</RequestingBusinessActivity>
<RespondingBusinessActivity nameID="BPSS-XML4BPM-Action-002" name="PerformRegistration"
isNonRepudiationRequired="false">
    <DocumentEnvelope nameID="BPSS-XML4BPM-Envelope-002" name="RegistrationResponseEnvelope" ... />
</RespondingBusinessActivity>
```

A business transaction is defined by the sequence of the requesting business activity and the responding business activity. In UMM each business transaction follows one out of six legally binding business patterns. The attribute *pattern* of the BPSS element *BusinessTransaction* specifies the underlying pattern type. Furthermore, the attribute *isGuaranteedDeliveryRequired* signals that reading partners must employ a delivery channel that provides a delivery guarantee. UMM always assumes guaranteed delivery if necessary.

The business transaction *register customer* (c.f. Figure 7) includes the requesting business activity *request registration* and the responding business activity *perform registration*. Its underlying pattern is a *query/response transaction*. Furthermore, we assume a guaranteed delivery channel.

```
<xsd:element name="BusinessTransaction">
    <xsd:complexType>
        <xsd:sequence>
            <snip/>
            <xsd:element ref="RequestingBusinessActivity"/>
            <xsd:element ref="RespondingBusinessActivity"/>
        </xsd:sequence>
        <xsd:attributeGroup ref="name"/>
        <xsd:attribute name="pattern" type="xsd:anyURI"/>
        <xsd:attribute name="isGuaranteedDeliveryRequired" type="xsd:boolean" default="false"/>
    </xsd:complexType>
    <snip/>
</xsd:element>

<BusinessTransaction  nameID="BPSS-XML4BPM-Transaction-001" name="RegisterCustomer"
pattern="QueryResponseActivity" isGuaranteedDeliveryRequired="true">
    <RequestingBusinessActivity nameID="BPSS-XML4BPM-Action-001" name="RequestRegistration" ... >
        <snip/>
    </RequestingBusinessActivity>
    <RespondingBusinessActivity nameID="BPSS-XML4BPM-Action-002" name="PerformRegistration" ... >
        <snip/>
    </RespondingBusinessActivity>
</BusinessTransaction>

<BusinessTransactionActivity name="RegisterCustomer" nameID="BPSS-XML4BPM-Activity-001"
businessTransaction="RegisterCustomer" businessTransactionIDREF="BPSS-XML4BPM-Transaction-001"
fromRole="customer" fromRoleIDREF="BPSS-XML4BPM-Role-001" toRole="seller" toRoleIDREF="BPSS-
XML4BPM-Role-002" isConcurrent="false" timeToPerform="PT4H" preCondition="some text" beginsWhen="some
text" endsWhen="some text" postCondition="some text"/>
```

3.3 Binary Collaboration

A binary collaboration is always conducted by two roles. They perform one or more business activities. In BPSS, a business activity is either a business transaction activity or a collaboration activity. We first concentrate on business transaction activities. Collaboration activities are briefly explained at the end of this subchapter. A business transaction activity points to a business transaction. The semantic of a business transaction activity is the same as in UMM (c.f. Figure 6). The business transaction activity is identified by its own *nameID* and *name*. It is quite common that the name of the referenced business

transaction is identical. However, this is not a must. The IDREF to the business transaction ensures unambiguouty. For each business transaction activity an initiating role (*fromRole*) and an reacting role (*toRole*) together with IDREFs to the role definitions are specified. The *fromRole* will perform the requesting business activity of the business transaction. The responding business activity is performed by the *toRole*. The UMM parameters *isConcurrent* and *timeToPerform* characterizing a business transaction activity exist in BPSS as well. Further attributes assigned to the BPSS business transaction activity are *preCondition*, *beginsWhen*, *endsWhen* and *postCondition*. Values for these attributes are usually defined in the transaction use case template (worksheet) of UMM.

The code fragment further below shows the definition of the register customer business transaction activity (c.f. Figure 6). It is a non-concurrent business transaction activity which must be executed within 4 hours. The customer initiates the referenced business transaction, which is defined in the previous subsection. The seller is the responder.

```
<xsd:complexType name="BusinessActivity">
    <xsd:attributeGroup ref="name"/>
    <xsd:attribute name="fromRole" type="xsd:string" use="required"/>
    <xsd:attribute name="fromRoleIDREF" type="GUIDREF"/>
    <xsd:attribute name="toRole" type="xsd:string" use="required"/>
    <xsd:attribute name="toRoleIDREF" type="GUIDREF"/>
    <xsd:attribute name="beginsWhen" type="xsd:string"/>
    <xsd:attribute name="endsWhen" type="xsd:string"/>
    <xsd:attribute name="preCondition" type="xsd:string"/>
    <xsd:attribute name="postCondition" type="xsd:string"/>
</xsd:complexType>

<xsd:element name="BusinessTransactionActivity">
        <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="BusinessActivity">
                <snip/>
                <xsd:attribute name="businessTransaction" type="xsd:string" use="required"/>
                <xsd:attribute name="businessTransactionIDREF" type="GUIDREF"/>
                <xsd:attribute name="isConcurrent" type="xsd:boolean" default="true"/>
                <snip/>
                <xsd:attribute name="timeToPerform" type="xsd:duration"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<BusinessTransactionActivity name="RegisterCustomer" nameID="BPSS-XML4BPM-Activity-001"
businessTransaction="RegisterCustomer" businessTransactionIDREF="BPSS-XML4BPM-Transaction-001"
fromRole="customer" fromRoleIDREF="BPSS-XML4BPM-Role-001" toRole="seller" toRoleIDREF="BPSS-
XML4BPM-Role-002" isConcurrent="false" timeToPerform="PT4H" preCondition="some text" beginsWhen="some
text" endsWhen="some text" postCondition="some text"/>
```

In UMM the business collaboration protocol choreographs the business transaction activities. This choreography is defined in BPSS by a binary collaboration. Figure 9 shows the subelement structure for a binary collaboration. Due to space limitation we do not present the XML schema code for each of these elements. The details of the binary collaboration are best explained by means of the example depicted in the business collaboration protocol of Figure 6 and mapped to the code fragment further below.

The binary collaboration simple purchase management is identified by a unique id (BPSS-XML4BPM-BinaryCollaboration-001). The attribute *isInnerCollaboration* is a flag indicating that the binary collaboration can only be used as subpart of another binary collaboration. Since this is not the case in our example it is set to false. The *initiatingRoleIDREF* references the role that initiates the first business transaction activity. This attribute is only relevant if a binary collaboration is subpart of another one (see further

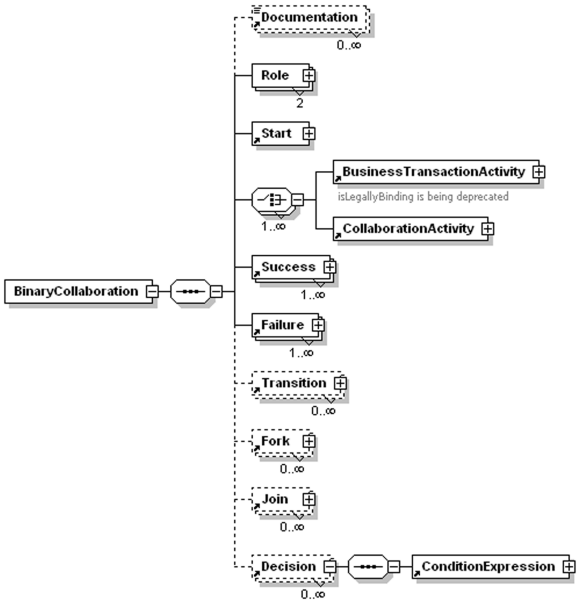

Figure 9. XML Schema for a Binary Collaboration

below). Another attribute indicates a pattern of a binary collaboration. Since these types of patterns are currently in development, this attribute is for future use only. A maximum time for a binary collaboration is defined by *timeToPerform* - a concept that does not exist on this level in UMM. The attributes *preCondition*, *beginsWhen*, *endsWhen* and *postConditions* are gathered in UMM by the business collaboration protocol use case.

Exactly two roles participate in a binary collaboration. In our example these roles are a customer and a seller, which are defined by *Role* elements. The *nameID* of a role must be unique for all binary collaborations regardless the fact that a role with the same name exists in another binary collaboration. The next element *Start* maps the UMM transition from the initial pseudo state to the first business transaction activity, which is *Search Products*. This business transaction activity is referenced by name (*toBusinessState*) and id (*toBusinessStateIDREF*). The referenced business transaction activity must be in the following list of all business transaction activities of the binary collaboration. Thus, the next elements define the business transaction activity *search products* as well as *order products* and *register customer*. The definition of business transaction activities was explained in detail above.

Next, the *Success* element specifies the successful completion, i.e. the transition from *Order Products* to the successful end state. The last business transaction activity *Order Products* is referenced by name (*from BusinessState*) and id (*fromBusinessStateIDREF*). The value *BusinessSuccess* for *conditionGuard* of the transition is taken from an enumerated list of different success and failure types. Similarly, the *Failure* elements specify the transitions from *Register Customer* and from *Order Products* to the failing end state.

Only then all the other transitions between the business activities are declared. A transition has its own id and references both source and target business transaction activity by

name and id. Again a transition guard from the enumerated list of success and failure types might be specified in *conditionGuard*. In addition the child element *ConditionExpression* enables a more complex specification reflecting the UMM business entity states. In our example we use an OCL syntax to denote the guards on the transitions. The Decision element - which follows the transition declarations - is the equivalent to the UML decision (depicted as rhombus). The underlying decision must be given in the subelement *ConditionExpression*; e.g. the OCL statement on whether the customer information is in state *Confirmed* or not. Other BPSS elements - not used in our example - include *Fork* for alternative flows and *Join* for parallel flows. Transitions reference not only business transaction activities as source and target, but also *Fork*, *Join*, and *Decision*. For example, the first transition statement is from Search Product to the decision node guarded by the fact that products are in state *Found* and substate *Wanted*.

UMM business collaboration protocols are always built by business transaction activities and do not include collaboration activities. These might be used in BPSS only. A collaboration activity references another binary collaboration which thereby becomes part of the parent binary collaboration. The other attributes of the collaboration activity are the same as for the business transaction activity except that *timeToPerform* and *isConcurrent* are not appropriate. The *fromRoleIDREF* of the collaboration activity binds to the *initiatingRole* of the included binary collaboration. Note, that the id values must be different and the names of the roles might be different. This allows to bind e.g. a buyer role from the parent collaboration to a customer role of the included collaboration.

```
<BinaryCollaboration name="SimplePurchaseManagement" nameID="BPSS-XML4BPM-BinaryCollaboration-001"
initiatingRoleIDREF="BPSS-XML4BPM-Role-001" isInnerCollaboration="false" pattern="nopatterns exist yet"
timeToPerform="P1D" preCondition="some text" beginsWhen="some text" endsWhen="some text"
postCondition="some text">
    <Role name="customer" nameID="BPSS-XML4BPM-Role-001"/>
    <Role name="seller" nameID="BPSS-XML4BPM-Role-002"/>
    <Start nameID="BPSS-XML4BPM-Start-001" toBusinessState="SearchProducts"
    toBusinessStateIDREF="BPSS-XML4BPM-Activity-002"/>
    <BusinessTransactionActivity name="SearchProducts" nameID="BPSS-XML4BPM-Activity-002" ... />
    <BusinessTransactionActivity name="OrderProducts" nameID="BPSS-XML4BPM-Activity-003" ... />
    <BusinessTransactionActivity name="RegisterCustomer" nameID="BPSS-XML4BPM-Activity-001"... />
    <Success nameID="BPSS-XML4BPM-Success-002" fromBusinessState="OrderProducts"
    fromBusinessStateIDREF="BPSS-XML4BPM-Activity-003" conditionGuard="BusinessSuccess"/>
    <Failure nameID="..." fromBusinessState="OrderProducts" ... conditionGuard="Failure"/>
    <Failure ... fromBusinessState="RegisterCustomer" ... conditionGuard="Failure"/>
    <Transition nameID="Transition-002-F001" fromBusinessState="SearchProduct"
    fromBusinessStateIDREF="BPSS-XML4BPM-Activity-002" toBusinessState="Decision1"
    toBusinessStateIDREF="BPSS-XML4BPM-Decision-001" conditionGuard="BusinessSuccess">
        <ConditionExpression expressionLanguage="OCL"
        expression="Products.oclInState(Found::Wanted) = TRUE"/>
    </Transition>
    <Transition ... fromBusinessState="Decision1" ... toBusinessState="RegisterCustomer" ...>
        <ConditionExpression ... expression="CustomerInformation.oclInState(Confirmed) = FALSE"/>
    </Transition>
    <Transition ... fromBusinessState="Decision1" ... toBusinessState="OrderProducts" ... >
        <ConditionExpression ... expression="CustomerInformation.oclInState(Confirmed) = TRUE"/>
    </Transition>
    <Transition ... fromBusinessState="RegisterCustomer" ... toBusinessState="OrderProducts" ...
    ConditionGuard="BusinessSuccess">
        <ConditionExpression ... expression="CustomerInformation.oclInState(Confirmed) = TRUE"/>
    </Transition>
    <Decision name="Decision1" nameID="BPSS-XML4BPM-Decision-001">
            <ConditionExpression ... expression="CustomerInformation.oclInState(Confirmed)"/>
    </Decision>
</BinaryCollaboration>
```

3.4 Mulitparty Transaction

The current version of BPSS 1.10 uses binary collaborations only. The XML schema allows the specification of multiparty transaction, but it is not recommended to use them. The concepts for multiparty collaborations might change considerably within the next revision. Multiparty collaborations are always synthesized by binary collaborations. However, the multiparty statement does not list the included binary collaborations. Instead the multiparty statement list the business partner roles that participate in the multiparty collaboration. Each business partner role includes one or more *Performs* elements to bind roles of binary collaborations. Since the id of a role is unique for all binary collaborations, the binary collaborations are also unambiguously identified.

The business partners in our example *simple order management* are buyer, seller and bank. The buyer takes on the role of customer in the simple purchase management. The seller binds the role of seller in the simple purchase management and the role of the credibility requestor in check credibility. The bank supports the role of a bank in check credibility. Furthermore, transitions between business transaction activities of different binary collaborations are supported. Since the business transaction activities of our example are nested this concept is not used in the code fragment below.

```
<MultiPartyCollaboration name="SimpleOrderManagement" nameID="BPSS-XML4BPM-MultiParty-001">
    <BusinessPartnerRole name="buyer" nameID="BPSS-XML4BPM-MultiRole-001">
        <Performs nameID="" role="customer" roleIDREF="BPSS-XML4BPM-Role-001"/>
    </BusinessPartnerRole>
    <BusinessPartnerRole nameID="BPSS-XML4BPM-MultiRole-002" name="seller">
        <Performs nameID="" role="seller" roleIDREF="BPSS-XML4BPM-Role-002"/>
        <Performs nameID="" role="credibilityRequestor" roleIDREF="BPSS-XML4BPM-Role-003"/>
    </BusinessPartnerRole>
    <BusinessPartnerRole nameID="" name="bank">
        <Performs nameID="" role="bank" roleIDREF="BPSS-XML4BPM-Role-004"/>
    </BusinessPartnerRole>
</MultiPartyCollaboration>
```

4 Summary

The BPSS specification states that its goal is to provide the bridge between e-business process modeling and specification of e-business software components. BPSS does not require any particular e-business process modeling methodology. Nevertheless, main concepts of BPSS are based on UMM, or better its meta model. Thus it is close at hand to model e-business collaborations with UMM and to map these models to XML-based BPSS. The gap between UMM and BPSS is quite close. This approach enables e-business software to interpret the choreography specified by UMM business collaboration models.

In this paper we first presented UMM as a top-down approach. Apart from introducing UMM's methodology, this section helps to understand the background of many BPSS concepts. Then the UMM business collaboration models are mapped to BPSS. We explained the mapping the other way round by a bottom-up approach. It is demonstrated that most of the UMM semantics - except those for requirements gathering - result in a equivalent BPSS counterpart. Most of them use the same terms and structures. Only very few UMM concepts are not supported by BPSS. Additional concepts available in BPSS but not supported in UMM are rare exceptions. These might be considered in future revisions of UMM.

From the arguments above it becomes obvious that alignment of BPSS and UMM is desirable. However, BPSS must also be aligned with the other ebXML specifications. The BPSS team has done a good job to deal with these sometimes conflicting interests. At the end of the 18-month ebXML initiative UN/CEFACT and OASIS agreed that ebXML business process specifications are maintained by UN/CEFACT. UN/CEFACT has signaled steps toward UMM BRV alignment in BPSS 2.0 and towards BDV alignment in BPSS 3.0. OASIS has formed its new ebXML business process technical committee in October 2003. Therefore, we stick with the current version 1.10 and await BPSS's future.

5 References

[BJR98] Booch, G., Jacobson, I., Rumbaugh J.: The Unified Modeling Language User Guide. Addison Wesley Object Technology Series, Reading, (1998)

[HHK02] Hofreiter, B., Huemer, C., Klas, W.: ebXML: Status, Research Issues and Obstacles. Proc. of 12th Int. Workshop on Research Issues on Data Engineering (RIDE02), San Jose (2002)

[HHN04] Hofreiter, B., Huemer, C., Naujok, K.-D.: UN/CEFACT's Business Collaboration Framework - Motivation and Basic Concepts. Proc. of MKWI'04 Track on Co-ordination in Value Creation networks / Agent Technology for Business Applications, LNI GITO (2004)

[HH03] Hofreiter, B., Huemer, C.: Modeling Business Collaborations in Context. Proc. of On The Move to Meaningful Internet Systems 2003: OTM 2003 Workshops, Springer LNCS, Catania (2003)

[ISO95] ISO: Open-edi Reference Model. ISO/IEC JTC 1/SC30 ISO Standard 14662 (1995)

[UN03a] UN/CEFACT TMG: UN/CEFACT Modelling Methodology - Meta Model, Revision 12. (2003), http://www.untmg.org/downloads/General/approved/UMM-MM-V20030117.zip

[UN03b] UN/CEFACT TMG: UMM User Guide, Revision 12. (2003) http://www.untmg.org/downloads/General/approved/UMM-UG-V20030922.zip

[UN03c] UN/CEFACT TMG: UN/CEFACT – ebXML Business Process Specification Schema, Version 1.10, http://www.untmg.org/downloads/General/approved/ebBPSS-v1pt10.zip

[UN03d] UN/CEFACT TMG: Core Components Technical Specification – Part 8 of the ebXML Framework, Version 2.01, (2003), http://www.untmg.org/downloads/General/approved/CEFACT-CCTS-Version-2pt01.zip

[UO01] UN/CEFACT, OASIS: ebXML Technical Architecture Specification v1.0.4. (2001), http://www.ebxml.org/specs/ebTA.pdf