Noname manuscript No.
(will be inserted by the editor)

Choreography of ebXML business collaborations

Birgit Hofreiter, Christian Huemer, Ja-Hee Kim

Institute of Distributed and Multimedia Systems, University of Vienna, at Liebiggasse 4/3-4, 1010 Vienna, Austria (e-mail: `birgit.hofreiter@univie.ac.at`, `christian.huemer@univie.ac.at`, `kim@mminf.univie.ac.at`)

The date of receipt and acceptance will be inserted by the editor

Abstract The ebXML framework consists of eight loosely coupled specifications for conducting e-Business. The choreography of ebXML business processes is defined by instances of the business process specification schema (BPSS). The BPSS is defined as an XML schema. It specifies elements describing the inter-organizational business processes, called business collaborations, but does not concentrate on intra-organizational business processes. Most of the underlying semantics were derived from the meta model of the UN/CEFACT Modeling Methodology (UMM). In this paper we describe the characteristics of ebXML business collaborations. We demonstrate how these concepts are captured by UMM and BPSS. These concepts must be supported by any proposed alternative business process interchange format in order to bridge between ebXML and other approaches, or even to replace BPSS.

Key words B2B e-Commerce – Business process management – Interchange Formats – XML – ebXML

1 Introduction

The exchange of business data between applications of different business partners goes back to the 1960s. These types of interchanges became known as Electronic Data Interchange (EDI) (Hill and Ferguson, 1989; Schatz, 1988). The most significant revolution in this area happened when XML entered the market. A lot of XML-based business document standards have been developed (Li, 2000; Kotok, 2000) and more and more companies became interested in business-to-business electronic commerce (B2B).

In addition to standardizing business document types, approaches supporting more advanced aspects of a B2B partnership appeared. ISO's Open-edi reference model (ISO, 1997) suggests the separation of the business logic in the *Business Service View* from its implementation in the *Functional Service View*. The most important standard of the *Business Service View* is UN/CEFACTs modelling methodology (UMM)(UN/CEFACT TMG, 2003 d). It is a methodology to define business collaboration models for B2B based on UML (Booch *et al.*, 1998). In this paper we focus on the choreography of UMM business collaborations.

The two most popular approaches of the *Functional Service View* are ebXML and Web Services. ebXML is directed towards e-Business in specific, whereas Web

Services originated from a remote procedure call background. Nevertheless, Web Services might be used in a B2B environment as well.

The Web Services base standards are WSDL, UDDI and SOAP. In a B2B environment, a WSDL file describes a business partner's interfaces to a collaborative business process. A UDDI registry is used to publish these services in order to inform prospective business partners of supported business processes. Once business partners with complementary interfaces have found each other, they collaborate by exchanging SOAP messages. Usually a business collaboration is a complex process that consists of many operations performed between the business partners. Therefore, a choreography amongst these operations must be defined. For this purpose different languages have been proposed: Web Services Choreography Interface (WSCI) (W3C, 2002 a), Web Services Conversation Language (WSCL) (W3C, 2002 b), Business Process Execution Language (BPEL) (OASIS, 2003; Leymann and Roller, 2004), and Web Services Choreography Description Language (WS-CDL) (W3C, 2004). Since BPEL seems to be the most commonly accepted language, we demonstrated how a UMM business collaboration is supported by BPEL in another paper (Hofreiter and Huemer, 2004).

In this paper we do not go into the details of a Web Services environment, but concentrate on ebXML business collaborations. ebXML offers a set of specifications in order to define business processes, building blocks for business documents, profiles of business partners and agreements between business partners. Furthermore, it offers a B2B-specific SOAP messaging and a registry/repository. Since we are interested in the choreography of business processes, we concentrate on the corresponding specification, that is the Business Process Specification Schema (BPSS). We show how a UMM version 12 business collaboration is implemented in ebXML by BPSS version 1.1.

Since both UMM and BPSS are specifically targeted towards B2B, and BPSS development was based on the UMM meta model, it is not surprising that both specifications share a lot of common concepts. These concepts should be supported by any other language considered as alternative to describe business collaboration choreographies. Alternatives might be the above mentioned Web Services languages, and languages like the Petri Net Markup Language (PNML) (Kindler, 2004), the Event-Driven Process Chains Markup Language (EPML) (Mendling and Nüttgens, 2004), and the Graph eXchange Language (GXL) (Winter and Simon, 2004).

The remainder of this paper is structured as follows: In Section 2 we introduce the two standards that our approach is based upon: ebXML and UMM. The overlapping concepts in both standards are business transactions and business collaborations. We introduce these key concepts in Section 3 and Section 4. Both sections have the same substructure. First, we introduce the relevant part of the UMM meta model. Next we present a UMM example demonstrating the key concepts. The third subsection shows how the UMM concepts are mapped to the BPSS schema definition. For a better understanding the UMM example is expressed in a BPSS equivalent in the fourth subsection. In case of the business collaboration we provide an additional subsection on complex choreographies as compared to well-known workflow patterns. A short summary in Section 5 concludes the paper.

2 Background information on UMM and ebXML BPSS

2.1 UN/CEFACT's Modeling Methodology (UMM)

In the mid-1990s UN/CEFACT had successfully introduced their UN/EDIFACT standards. Since UN/EDIFACT implementations resulted in high start-up costs, UN/CEFACT already started to look for their next generation EDI standards. For this purpose UN/CEFACT followed the idea of Open-edi. The UN/CEFACT vision was to develop business collaboration models that do not have the problem of ambiguity; instead they would describe the complete processes and their information requirements, including constraints, options in execution, exceptions, etc. Of course, it was recognized that the issue of businesses wishing to do things differently would not go away. However, it was envisioned that software providers would implement the most common scenarios in their software packages for small and medium enterprises (SMEs). This should lead to a critical mass of SME users in order to attract large companies to support these "common" scenarios instead of insisting on their proprietary scenarios.

Following this vision, UN/CEFACT started its work on UMM in 1997. UMM concentrates on the business semantics of a B2B partnership and is, thus, a BOV-related standard. Its goal is capturing the commitments made by business partners. These commitments are then reflected in the resulting choreography of the business collaboration. Another standards organization following a similar approach is RosettaNet. In 2000, the company EDIFECS transferred their copyrights of the RosettaNet modeling approach to UN/CEFACT. UN/CEFACT merged this approach into UMM. Also other organizations that participated in the development of UMM, like SWIFT, EAN*UCC, and TM Forum aligned their methodologies with UMM.

All the UMM artifacts are documented by the means of UML. Moreover, the methodology is based on the UMM meta model (UN/CEFACT TMG, 2003 c) that defines a coherent set of stereotypes, constraints, and tag definitions, i.e. a UML profile for the purpose of modeling business collaborations. It should be noted that most of the terminology used in this paper refers to the definitions in the UMM meta model.

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.

2.2 electronic business XML (ebXML)

The business collaboration models defined in the BOV must be supported by an IT-infrastructure on the FSV layer. UN/CEFACT recognized the growing demand for XML-based solutions and teamed up with OASIS to start the ebXML initiative in November 1999. The ebXML vision is best explained by the ebXML scenario between a large corporation (Company A) and a SME (Company B) as illustrated in Figure 1 (UN/CEFACT and OASIS, 2001).

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 the business scenarios it supports. Company B, which uses an ebXML-

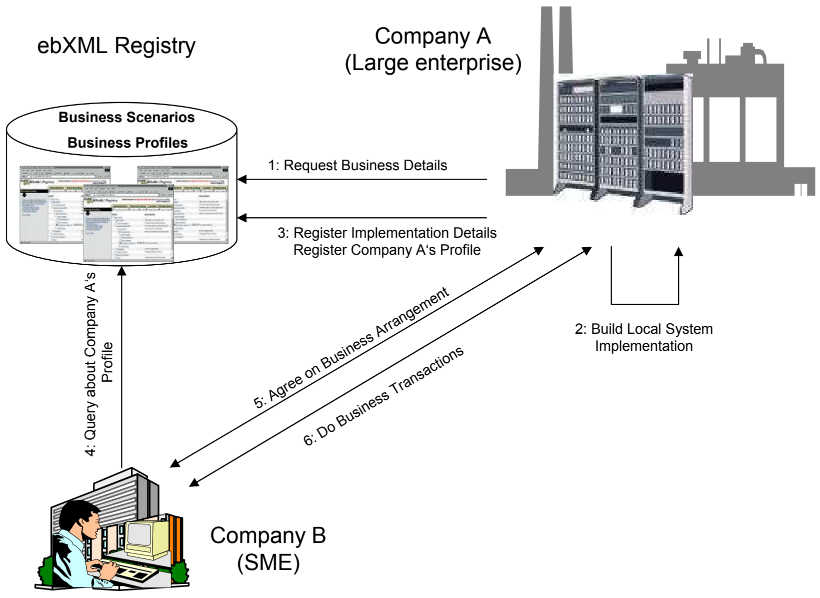
ebXML Registry
Business Scenarios
Business Profiles
Company A
(Large enterprise)
1: Request Business Details
3: Register Implementation Details
Register Company A's Profile
2: Build Local System
Implementation
4: Query about Company A's
Profile
5: Agree on Business Arrangement
6: Do Business Transactions
Company B
(SME)

Fig. 1 ebXML scenario.

compliant software package, 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. Then, Company A accepts the business agreement. Company A and B are ready to engage in e-business using ebXML (step 6).

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 (Hofreiter *et al.*, 2002). The current set of specifications comprises the ebXML requirements (ebREQ), technical architecture (ebTA), messaging service (ebMSG), registry information model (ebRIM), registry services specification (ebRS), collaboration protocol profile and agreement specification (CPPA), business process specification schema (BPSS), and core components (CC).

The ebXML standard to describe business collaborations 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 core components (CC). A business partner will reference in its profile (CPP), 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. This guarantees 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 (CPA).

The goal of the BPSS is to provide the bridge between e-business process modeling and specification of e-business software components (UN/CEFACT TMG, 2003 b). It provides an XML schema to specify a collaboration between business partners, and to provide 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. BPSS identified those UMM modeling elements that are relevant for the runtime systems and discarded the rest. The relevant modeling elements have been expressed in XML schema.

Frankly speaking, the BPSS can be regarded as an "XML-ification" of UMM's BTV. The most important artifacts of the BTV are business transactions and their choreography into business collaborations. Thus, we concentrate in the next two sections on each of these artifacts. We do not introduce in detail any other UMM artifacts. The reader interested in UMM is referred to (UN/CEFACT TMG, 2003 d; Hofreiter and Huemer, 2003). Please note that 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 a top-down or bottom-up approach is used. For educational purposes we use a bottom up approach to describe the concepts of both UMM and BPSS.

3 Business Transactions

Both UMM and BPSS use the concept of a business transaction in a very similar way. It is the basic building block to define a choreography of a business collaboration between collaborating business partners. Communication in a business collaboration is about aligning the information systems of the business partners. Aligning the information systems means that all relevant business objects (e.g. purchase orders, line items, etc.) are in the same state in each information system. If a business partner recognizes an event that changes the state of a business object, it initiates a business transaction to synchronize with the collaborating business partner. It follows that a business transaction is an atomic unit that leads to a synchronized state in both information systems.

We distinguish two very basic types of business transactions: In the first type, 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 acknowledgments) flows only from the initiating to the reacting business partner. In the second type, 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. E.g., once a purchase order is agreed upon in a business transaction a rollback is not allowed anymore, but requires the execution of a *cancel order* business transaction.

3.1 Business Transactions in UMM

In Figure 2 we present an extract of the UMM meta model showing those meta classes and associations that relate to business transactions. Note, that the semantics presented in this figure span over multiple diagrams of the original UMM meta model (UN/CEFACT TMG, 2003 c).

In the BRV, the requirements of a business transaction are documented by a business transaction use case. UMM provides a worksheet that is assigned to a business transaction use case. The worksheet documents the following requirements: *pre-conditions, begins when, definition, ends when, exceptions, post-conditions, requesting business function, responding business function.* Usually business experts

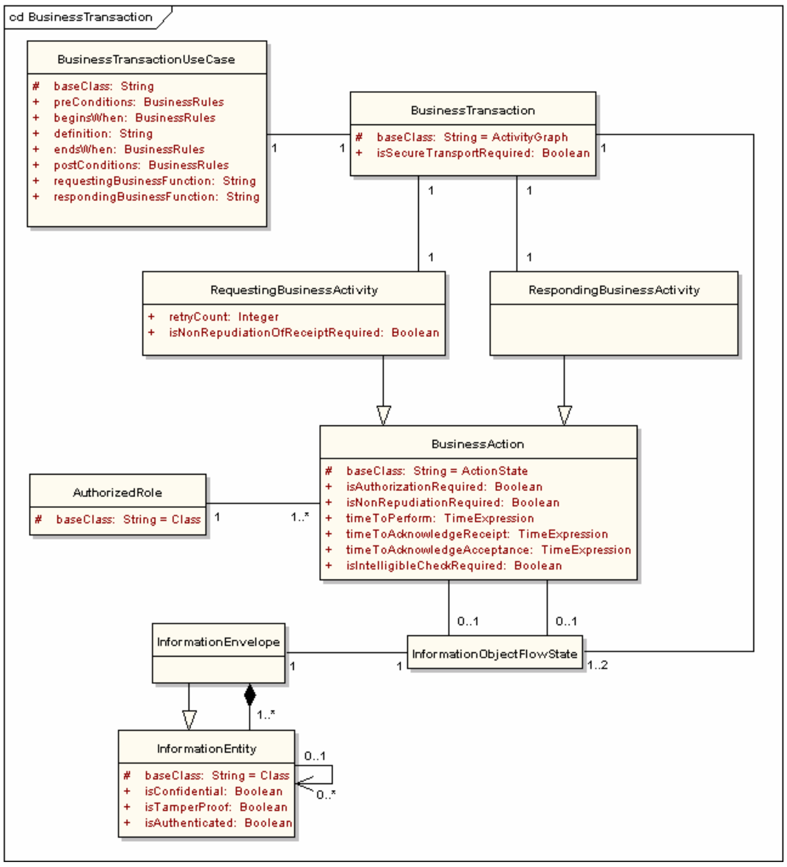
cd BusinessTransaction
BusinessTransactionUseCase
# baseClass: String
+ preConditions: BusinessRules
+ beginsWhen: BusinessRules
+ definition: String
+ endsWhen: BusinessRules
+ postConditions: BusinessRules
+ requestingBusinessFunction: String
+ respondingBusinessFunction: String
BusinessTransaction
# baseClass: String = ActivityGraph
+ isSecureTransportRequired: Boolean
RequestingBusinessActivity
+ retryCount: Integer
+ isNonRepudiationOfReceiptRequired: Boolean
RespondingBusinessActivity
BusinessAction
# baseClass: String = ActionState
+ isAuthorizationRequired: Boolean
+ isNonRepudiationRequired: Boolean
+ timeToPerform: TimeExpression
+ timeToAcknowledgeReceipt: TimeExpression
+ timeToAcknowledgeAcceptance: TimeExpression
+ isIntelligibleCheckRequired: Boolean
AuthorizedRole
# baseClass: String = Class
InformationEnvelope
InformationObjectFlowState
InformationEntity
# baseClass: String = Class
+ isConfidential: Boolean
+ isTamperProof: Boolean
+ isAuthenticated: Boolean
1
1
1
1
1
1
1
1
1..*
0..1
0..1
1
1
1..2
1..*
0..1
0..*

Fig. 2 UMM meta model extract related to business transactions.

will use human language to complete the worksheets. According to the semantics of business transactions described above, the properties for *pre-conditions* and *post-conditions* reference business states of business entities. A business process analyst is responsible for guiding the business experts to define these states.

Each *business transaction use case* results in exactly one *business transaction* which is represented by an activity graph. The *is secure transport required* flag signals that both business partners must agree to exchange business information using a secure transport channel. The activity graph of a *business transaction* is always built by exactly *two business actions*, a *requesting business activity* and a *responding activity*. Each *business action* is performed by exactly one *authorized role* executed by a business partner. The assignment of the business action to an authorized role is realized by the UML concept of swimlanes. The requesting business activity is assigned to the initiating role and the responding activity is assigned to the reacting role. Note, we stick here to the UMM terms, although initiating and reacting business activities would be better terms, since in one way transactions there exist a responding activity, but no response is sent.

In UMM we distinguish two types of one-way transactions. If the business information sent is a formal non-repudiatable 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 available beforehand, 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. If so, it is a commercial transaction. Otherwise it is a request/response transaction. These types of business transactions cover all known legally binding interactions between two decision making applications as defined in Open-edi (ISO, 1997). They have proven to be useful in RosettaNet. In UMM the *requesting business activity* is stereotyped according to the transaction type. This means that there exists a subclass of requesting business activity for each of the six patterns. These subclasses do not have any additional attributes. The subclass structure is not shown in Figure 2, in order to keep it simple.

The different types of business transaction patterns differ in the default values for the attributes that characterize *business actions*: *is authorization required, is non-repudiation required, time to perform, time to acknowledge receipt, and time to acknowledge acceptance.* The values for *is non-repudiation of receipt required* and for *retry count* are only defined for the *requesting business activity.* Most of these attributes are self-explanatory. An acknowledgment of receipt is usually sent after grammar validation, sequence validation, and schema validation. However, if the *is intelligible check required* flag is set to *false*, the acknowledgment is sent immediately after receipt without any validation. An acknowledgment of acceptance is sent after validating the content against additional rules to ensure that the content is processable by the target application. *Retry count* is the number of retries in case of control failures.

In a one-way transaction business information is exchanged only from the *requesting business activity* to the *responding business activity.* In case of a two-way transaction the *responding business activity* returns business information to the *requesting business activity.* The exchange of business information is shown by an object flow. One *business action* sets an *information object flow state* that is consumed by the other *business action.*

An *information object flow state* refers to an *information envelope* exchanged between the *business actions.* Within an *information envelope* there are one or more information entities representing the business information, which might be structured recursively. Each *business information entity* is characterized by three security parameters: *is confidential, is tamper proof,* and *is authenticated.* Since an *information envelope* is a specialization of an *information entity*, the security parameters characterize the *information envelope* as well.

The structure of the information envelope's content is modeled in a class diagram. In order to guarantee reusability, the structure must be built using common building blocks. Hence, it was recently agreed that *information entities* should be based on ebXML core components (UN/CEFACT TMG, 2003 a). Currently, UN/CEFACT is building a library of core components which will become the richest and most cross-industry harmonized source for assembling business information. Since the structure of the information exchanged does not influence the choreography - which is the primary focus of this paper - we do not further concentrate on this topic.

3.2 UMM business transaction example

In order to demonstrate UMM business transactions we take the example of a *request for quote* transaction. This business transaction is depicted in the activity diagram of Figure 3. The white numbers with black background refer to line numbers of the BPSS code in section 3.4 presenting the BPSS equivalent. In a *request* for quote transaction the buyer takes on the role of the initiator and the seller performs the reacting role. The buyer starts a *request for quote* activity. This activity creates

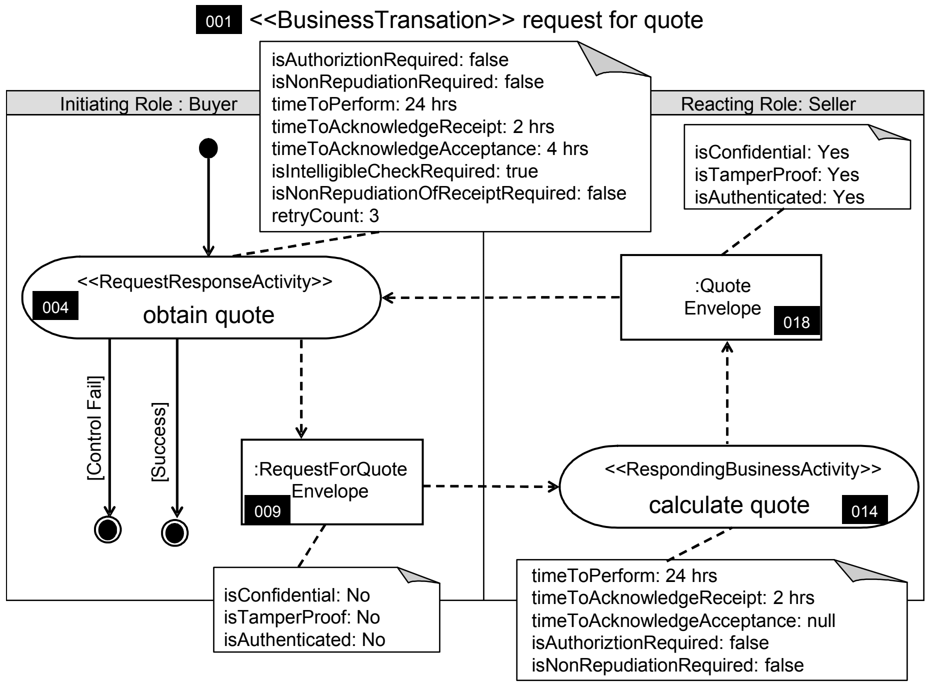
001 <<BusinessTransation>> request for quote
isAuthoriztionRequired: false
isNonRepudiationRequired: false
timeToPerform: 24 hrs
timeToAcknowledgeReceipt: 2 hrs
timeToAcknowledgeAcceptance: 4 hrs
isIntelligibleCheckRequired: true
isNonRepudiationOfReceiptRequired: false
retryCount: 3
Initiating Role : Buyer
Reacting Role: Seller
isConfidential: Yes
isTamperProof: Yes
isAuthenticated: Yes
<<RequestResponseActivity>>
004 obtain quote
:Quote Envelope 018
[Control Fail]
[Success]
:RequestForQuote Envelope 009
<<RespondingBusinessActivity>>
calculate quote 014
isConfidential: No
isTamperProof: No
isAuthenticated: No
timeToPerform: 24 hrs
timeToAcknowledgeReceipt: 2 hrs
timeToAcknowledgeAcceptance: null
isAuthoriztionRequired: false
isNonRepudiationRequired: false

Fig. 3 Business transaction 'request for quote'.

a *request for quote envelope* that triggers the *calculate quote* activity of the seller. According to UMM business transaction semantics, the *obtain quote activity* does not end after sending the envelope - it is still alive. The *calculate quote* activity outputs the *quote envelope*, which is returned to the *obtain quote* activity. The resulting activity graph - which does not show any control flows and is built by loosely coupled, collaborating activities - does not follow the UML 1.4 semantics. However, this interpretation is well accepted by the e-business community and all the patterns are based on this concept.

The *obtain quote* activity requires neither authorization nor non-repudiation. This holds also for the *calculate quote* activity. Both activities have to be executed within 24 hours. They expect an acknowledgment of receipt after performing intelligible checks within 2 hours. Furthermore, the *obtain quote* requires a acknowledgment of acceptance within 4 hours. In case of a time-out the *obtain quote* activity tries to restart 3 times. All the security parameters do not apply to the *request for quote* envelope, but for the *quote envelope*.

It becomes obvious that the activity graph of a *business transaction* shows only the exchange of business information in the corresponding envelopes, but does not show any business signals for acknowledgements. These acknowledgments are specified in the tagged values of the *business actions*. However, this approach covers all information necessary for the business service interfaces (BSI) on each partner's side. In case of a successful business transaction, the flow of messages will be as depicted in Figure 4.

3.3 Business Transaction in BPSS

BPSS defines an XML element *business transaction* to cover the same semantics as defined in UMM. In Figure 5 we present the business transaction element's structure. This structure includes only the important elements and omits elements for documentation, etc. We use a notation originally used by the software XML Spy,

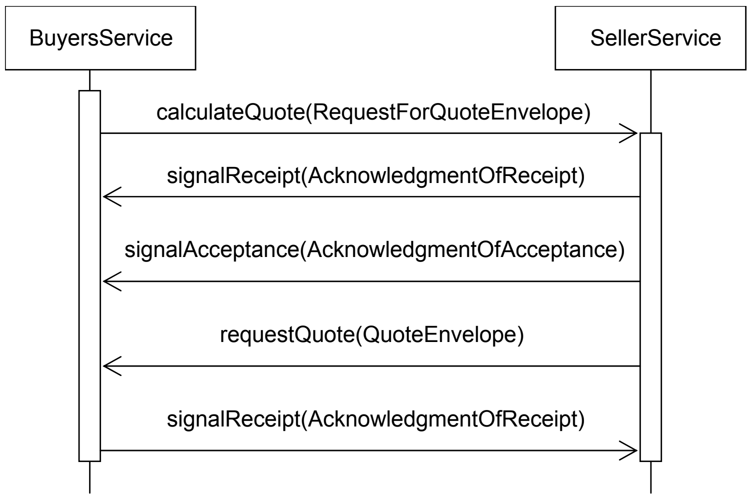
BuyersService
SellerService
calculateQuote(RequestForQuoteEnvelope)
signalReceipt(AcknowledgmentOfReceipt)
signalAcceptance(AcknowledgmentOfAcceptance)
requestQuote(QuoteEnvelope)
signalReceipt(AcknowledgmentOfReceipt)

Fig. 4 The messages flow 'request for quote'.

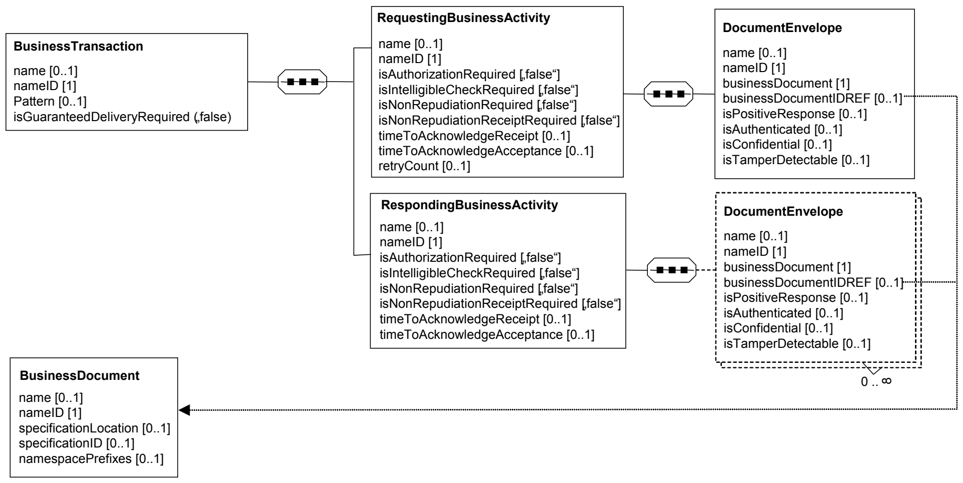
BusinessTransaction
name [0..1]
nameID [1]
Pattern [0..1]
isGuaranteedDeliveryRequired („false)
RequestingBusinessActivity
name [0..1]
nameID [1]
isAuthorizationRequired [„false"]
isIntelligibleCheckRequired [„false"]
isNonRepudiationRequired [„false"]
isNonRepudiationReceiptRequired [„false"]
timeToAcknowledgeReceipt [0..1]
timeToAcknowledgeAcceptance [0..1]
retryCount [0..1]
RespondingBusinessActivity
name [0..1]
nameID [1]
isAuthorizationRequired [„false"]
isIntelligibleCheckRequired [„false"]
isNonRepudiationRequired [„false"]
isNonRepudiationReceiptRequired [„false"]
timeToAcknowledgeReceipt [0..1]
timeToAcknowledgeAcceptance [0..1]
DocumentEnvelope
name [0..1]
nameID [1]
businessDocument [1]
businessDocumentIDREF [0..1]
isPositiveResponse [0..1]
isAuthenticated [0..1]
isConfidential [0..1]
isTamperDetectable [0..1]
DocumentEnvelope
name [0..1]
nameID [1]
businessDocument [1]
businessDocumentIDREF [0..1]
isPositiveResponse [0..1]
isAuthenticated [0..1]
isConfidential [0..1]
isTamperDetectable [0..1]
0 .. ∞
BusinessDocument
name [0..1]
nameID [1]
specificationLocation [0..1]
specificationID [0..1]
namespacePrefixes [0..1]

Fig. 5 BPSS business transaction element.

which is well-accepted in the XML community. However, we extend this notation to show the elements' attributes as well. For each attribute we define whether it is required (marked [1]) or optional [0..1]. If a default value is assigned to an optional element we specify the default value ["default"] instead. Furthermore we use a dotted arrow to depict logical reference to other elements.

In general all BPSS elements include an attribute for a unique id (*nameID*) and another one for a human readable identification (*name*). The business transaction element further comprises two more attributes. The domain of the *patterns* attribute covers the six types of business transactions defined in UMM. The pattern is correctly placed on the transaction level (as opposed to stereotyping the requesting business activity in UMM). The attribute *is guaranteed delivery required* is used to declare that the business partners must employ only a delivery channel that provides a delivery guarantee. Note, that this attribute is not semantically congruent with *is secure transport required* which characterizes business transactions in UMM. The latter does not exist in BPSS.

The element *business transaction* includes the two child elements *requesting business activity* and *responding business activity*. Their attributes are similar to those of their UMM equivalents. A value of type duration for *time to acknowledge receipt* and *time to acknowledge acceptance* signals that the corresponding acknowledgment is needed. If no acknowledgment is needed the attribute is simply omitted in the instance. Both *is non repudiation required* and *is non repudiation of receipt required* are attributes of both business action types - in UMM the latter does not exist for the responding business activity. The *retry count* for the requesting business activity is also used as in UMM. The *is authorization required* attribute is defined, but deprecated, because it cannot be supported by current ebXML business service interfaces. In BPSS no attribute *time to perform* exists, because it is not considered relevant for the business service interfaces.

In BPSS the element *document envelope* is equivalent to *information envelope* in UMM. The *document envelop* element is a child element of the business action that creates it. The *requesting business activity* includes a mandatory *document envelope*, whereas the *document envelope* is an optional child element of *responding business activity.* No link exists from the *document element* to the business action that consumes the envelope. Since a business transaction includes exactly two collaborating business actions, it is clear that it must be the other business action (not including the document as child element). A *responding business activity* might include more *document envelope* children. However, only one of them will be sent at run-time. In BPSS each different option for a response document type has its own *document envelope.* This is contrary to UMM where the same *information envelope* is used for different options of documents. The attribute *is positive response* is set *true* for all envelopes that carry documents that lead to a success from a business perspective.

The document envelope element also includes attributes for the security parameters: *is authenticated, is confidential*, and *is tamper detectable* (in UMM: *is tamper proof*). 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.

Furthermore the *document envelope* includes the attributes *business document* and *business document IDREF* to reference a *business document element*, which is sent as the envelope's content. The *business document* element does not include any substructure to define a document type. Instead its attributes *specification location* or *specification ID* reference the schema of the corresponding document type.

3.4 BPSS Business Transaction Example

The code fragment below presents the BPSS equivalent to Figure 3. Note that we do not use realistic unique IDs for the *nameID* attributes. Instead we build IDs by the first characters of the concept followed by a sequential number (e.g. BA2 for the second business action) in order to allow easier tracing for the reader.

The business transaction *request for quote* starts at line 001. It includes the requesting business activity *obtain quote* (004) and the responding business activity *calculate quote* (014). The *request for quote envelope* (009) is a child of *obtain quote* that creates the envelope. Similarly, *calculate quote* includes the child element *quote envelope.* The business documents included in the envelopes (025 and 028) are referenced by their unique id (BD1 and BD2). All the attributes of all the BPSS elements correspond more or less to the tagged values of the UMM business transaction.

```
001    <BusinessTransaction
002        nameID="BT1" name="request for quote"
003        pattern="RequestResponse" isGuaranteedDeliveryRequired="true">
004            <RequestingBusinessActivity
```

```
            nameID="BA1" name="obtain quote"
            retryCount="3" isIntelligibleCheckRequired="true" isAuthorizationRequired="false"
            isNonRepudiationRequired="false" isNonRepudiationReceiptRequired="false"
            timeToAcknowledgeReceipt="PT2H" timeToAcknowledgeAcceptance="PT4H">
                <DocumentEnvelope
                    nameID="DE1" name="RequestForQuoteEnvelope"
                    businessDocument="RequestForQuote" businessDocumentIDREF="BD1"
                    isAuthenticated="none" isConfidential="none" isTamperDetectable="none"/>
        </RequestingBusinessActivity>
        <RespondingBusinessActivity
            nameID="BA2" name="calculate quote"
            isIntelligibleCheckRequired="true" isAuthorizationRequired="false"
            isNonRepudiationRequired="false" timeToAcknowledgeReceipt="PT2H" >
                <DocumentEnvelope
                    nameID="DE2" name="QuoteEnvelope"
                    businessDocument="Quote" businessDocumentIDREF="BD2"
                    isAuthenticated="transient" isConfidential="transient"
                    isTamperDetectable="transient"/>
        </RespondingBusinessActivity>
</BusinessTransaction>
<BusinessDocument
    nameID="BD1" name="RequestForQuote"
    specificationLocation="http://www.example.org/RequestForQuote.xsd"/>
<BusinessDocument
    nameID="BD2" name="Quote"
    specificationLocation="http://www.example.org/Quote.xsd"/>
```

4 Business Collaborations

A business process is defined as an organized group of related activities that together create customer value (Hammer and Champy, 1993). 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. A multi-party collaboration implies that three or more partners collaborate. 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 out of a number of business transactions. It is important that the business collaboration defines an execution order for the business transactions, i.e. a business collaboration choreography.

4.1 Business Collaborations in UMM

An extract of the UMM meta model for business collaborations and their relations to business transactions is shown in Figure 6. The requirements of a business collaboration are documented in a business collaboration use case. In UMM there exist two subtypes of business collaboration use cases: the business transaction use case - we already introduced in the previous section - and the business collaboration protocol use case for more complex business collaborations. The worksheet assigned to the latter addresses *pre-conditions, begins when, definition, ends when, exceptions* and *post-conditions.*

Each *business collaboration protocol use case* specifies the requirements for the choreography that is defined by an activity graph called *business collaboration protocol.* Currently, all activities of the *business collaboration protocol* must be stereotyped as *business transaction activity.* 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 at 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

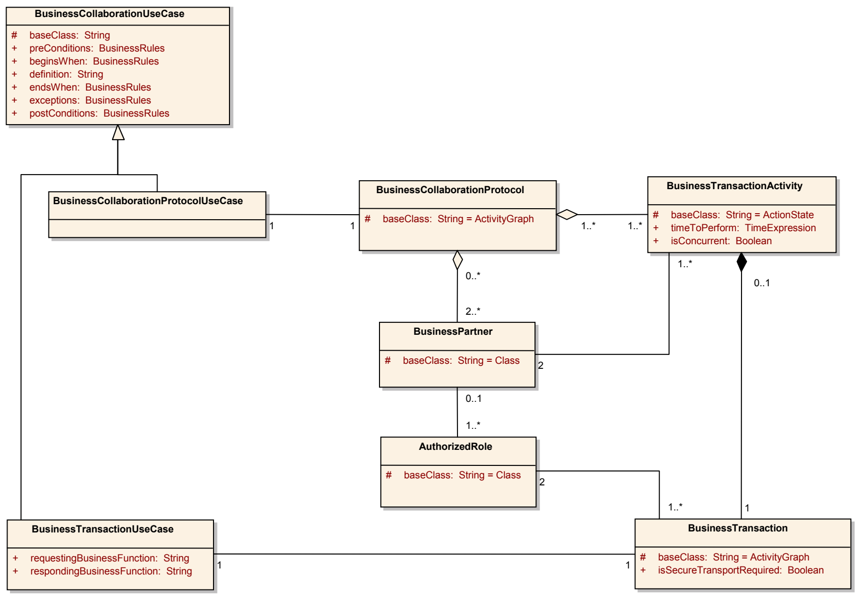
BusinessCollaborationUseCase
# baseClass: String
+ preConditions: BusinessRules
+ beginsWhen: BusinessRules
+ definition: String
+ endsWhen: BusinessRules
+ exceptions: BusinessRules
+ postConditions: BusinessRules
BusinessCollaborationProtocolUseCase
1
1
BusinessCollaborationProtocol
# baseClass: String = ActivityGraph
1..*
1..*
BusinessTransactionActivity
# baseClass: String = ActionState
+ timeToPerform: TimeExpression
+ isConcurrent: Boolean
0..*
2..*
1..*
0..1
BusinessPartner
# baseClass: String = Class
2
0..1
1..*
AuthorizedRole
# baseClass: String = Class
2
1..*
1
BusinessTransactionUseCase
+ requestingBusinessFunction: String
+ respondingBusinessFunction: String
1
1
BusinessTransaction
# baseClass: String = ActivityGraph
+ isSecureTransportRequired: Boolean

Fig. 6 UMM meta model extract related to business collaboration.

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.

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 (in addition) the availability of a business object in a certain 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 type *junction*), as well as *fork* and *join.*

A *business collaboration protocol* specifies the choreography of activities among two or even more *business partners.* Due to the 1:1-relationship of *business transaction activity* and *business transaction*, the number of *business partners* participating in a *business transaction activity* is exactly two. Different *business transaction activities* of the same *business collaboration protocol* might be performed by a different pair of *business partners.* However, nested *business transaction activities* are not allowed owing to the transition semantics. Nested transactions are common in multi-party transactions. E.g., *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.

4.2 UMM Business Collaboration Protocol Example

For the purpose of demonstrating a UMM business collaboration protocol we look at the example of a simple purchase order management. The over-simplified choreography is shown in Figure 7. Again, the white numbers with black background refer to lines of the BPSS code in section 4.4 presenting the same concept.

The business collaboration in question is a binary collaboration between a buyer and a seller. The first activity is *request for quote.* This business transaction activity is refined by the homonymous business transaction that we detailed in section 3. It

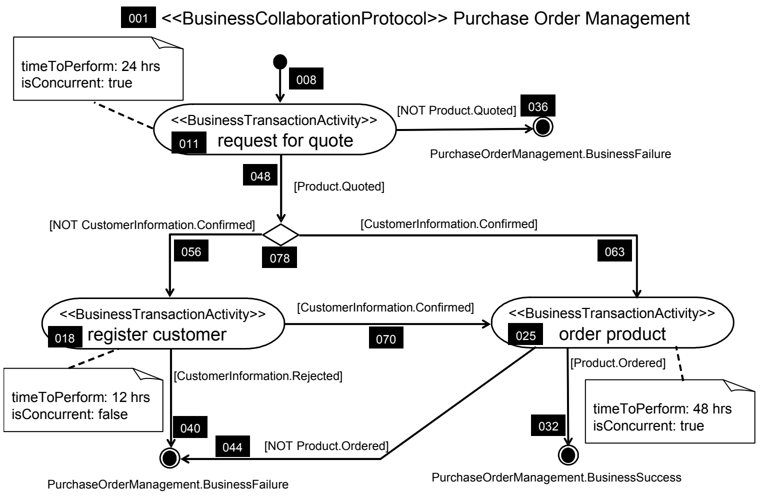
001 <<BusinessCollaborationProtocol>> Purchase Order Management
timeToPerform: 24 hrs
isConcurrent: true
008
<<BusinessTransactionActivity>>
011 request for quote
[NOT Product.Quoted]
036
PurchaseOrderManagement.BusinessFailure
048
[Product.Quoted]
[NOT CustomerInformation.Confirmed]
[CustomerInformation.Confirmed]
056
078
063
<<BusinessTransactionActivity>>
018 register customer
[CustomerInformation.Confirmed]
070
<<BusinessTransactionActivity>>
025 order product
[Product.Ordered]
[CustomerInformation.Rejected]
timeToPerform: 12 hrs
isConcurrent: false
timeToPerform: 48 hrs
isConcurrent: true
040
044
[NOT Product.Ordered]
032
PurchaseOrderManagement.BusinessFailure
PurchaseOrderManagement.BusinessSuccess

Fig. 7 Business collaboration protocol 'purchase order management'.

must be executed in 24 hours and might be concurrent. If *request for quote* does not result in a quote for the product, i.e. the business entity *product* is in state *quoted*, the business collaboration ends with a *business failure*. If a quote for the product is received, the buyer is able to order the product. However, ordering requires the buyer to be registered at the seller. Hence, a decision node splits for either going directly to the *order product* activity or first to the *register customer* activity. *Register customer* must be performed in 12 hours and concurrent registrations do not make any sense. If the customer is not registered by the seller, the collaboration ends with a *business failure*. If the customer information is confirmed by the seller the product is ready to be ordered. *Order product* must be realized in 48 hours and concurrent instantiations are possible. A successful order process, which sets the business object *product* into state *ordered*, leads to a business success of the purchase order management.

4.3 Business Collaboration in BPSS

The choreography of a business collaboration is described in BPSS by the element *binary collaboration*. As the name of this element suggests, it defines business collaborations between two business partners only. The BPSS schema file includes an element for multi-party collaboration. However, the element was deprecated and it will be subject to major changes in future BPSS versions. Consequently, we do not consider multi-party collaborations in this paper. The support of recursively structuring binary collaborations is a key difference compared to UMM. Accordingly, an activity of a *binary collaboration* is either a *business transaction activity*, which is refined by a *business transaction*, or a *collaboration activity*, which is refined by another *binary collaboration*.

The *binary collaboration* element includes attributes as defined in a UMM business collaboration use case: *begins when*, *ends when*, *pre-condition*, and *post-condition*. A binary collaboration must be completed within the time frame specified in the *time to perform* attribute. The business partner which initiates the first activity of the binary collaboration is referenced by *initiating role IDREF*. We already explained that binary collaborations might be nested. If a binary collaboration cannot be a top level one, but must be nested in another one, the value of *is inner collaboration* is set *true*. The *pattern* attribute is reserved for future use, when UN/CEFACT has developed reference models for negotiation, order-fulfillment-settlement, long term contract with periodic releases, etc.

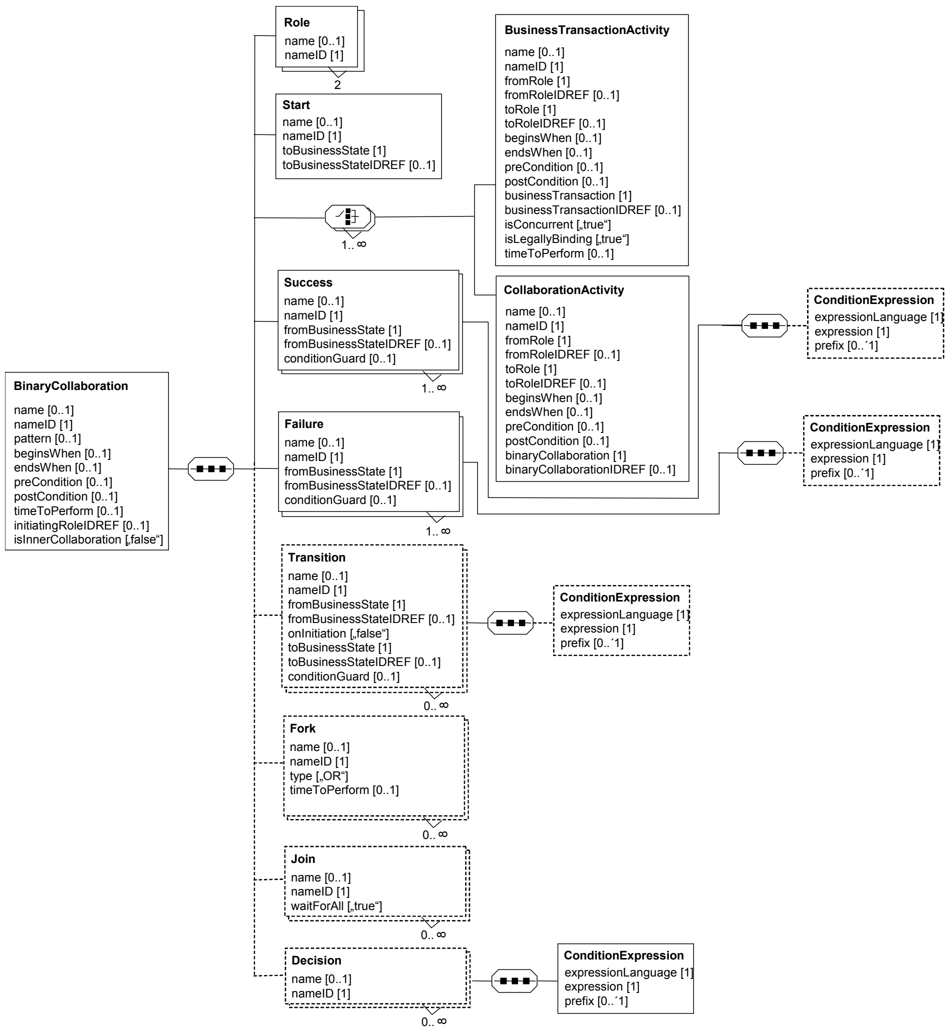

Fig. 8 BPSS Binary Collaboration element .

The *binary collaboration* element includes child elements that represent the components of the corresponding UMM activity graph. These elements can be classified into the categories business partners, activities, pseudo states and transitions. The *binary collaboration* element includes two *role* elements, one for each participating business partner type.

A binary collaboration is built by one or more activities. Each activity is described either by a *business transaction activity* element or a *collaboration activity* element. Each one uses the *from Role* and *from Role IDREF* attributes to reference the initiating role, and the *to role* and *to role IDREF* attributes to reference the reacting role. The referenced *role* elements must be the ones included in the same binary collaboration. Again, the *begins when*, *ends when*, *pre-condition* and *post-condition* attributes refer to corresponding use case descriptions. A *business transaction activity* element references the refining *business transaction* element by the attributes *business transaction* and *business transaction IDREF*. Similarly, a *collaboration activity* element references a refining *binary collaboration* element by the attributes *binary collaboration* and *binary collaboration IDREF*. Only *business transaction activity* elements contain the attributes *is concurrent* and *time to per-*

form as defined in UMM. Furthermore, they include an attribute to signal that performing the business transaction is *legally binding*, which is *true* by default.

The *binary collaboration* contains also elements for pseudo states known from UML: *fork*, *join*, and *decision*. However, the semantics of the BPSS elements is not always identical to UML. If the *type* attribute of *fork* is *XOR*, only one of the successors will become active and the first one wins. If the *type* is *OR*, the interpretation corresponds to UML, where all successors occur in parallel. A *fork* has a *time to perform* attribute. If this time interval is succeeded, the state will be automatically moved to the corresponding *join*. The *join* element has a *wait for all* attribute, which is *true* by default and corresponds in this case to the UML semantics. If it is *false*, only the first transition reaching the *join* state activates the successor. Furthermore, the condition of a *decision* element is given in the child element *condition expression*. The use of pseudo states in order to define complex choreographies is explained in more detail in section 4.5.

The *transition* element specifies the transition from one state to another by using the attributes *from business state*, *from business state IDREF*, as well as *to business state* and *to business state IDREF*. A referenced business state might be a *business transaction activity* element, a *collaboration activity* element, as well as one of the pseudo states *fork*, *join*, and *decision*. The *on initiation* attribute is set *true*, if the transition occurs already when the requesting document is sent in the source transaction. This enables nesting transactions. The guard of a transition is given in the *condition guard* attribute. Values correspond to an enumeration of different business/protocol success and failure indicators. More complex guards, like that of business entity states, have to be specified in the child element *condition expression*.

The elements *start*, *success*, and *failure* are a combination of the corresponding pseudo state and a transition to/from it. There is only one *start* element, but one ore more *success* and *failure* elements. The attributes *from business state*, *from business state IDREF*, *to business state*, and *to business state IDREF*, and *condition guard* are identical to those in the *transition* element. Similarly, a *condition expression* is an optional child element.

4.4 BPSS Binary Collaboration Example

In the code fragment below we show the BPSS definition of the binary collaboration *purchase order management* that is equivalent to the business collaboration protocol of Figure 7. For a better understanding we refer to both line numbers and element IDs. The binary collaboration is performed by a *buyer* (006) and a *seller* (007). The binary collaboration includes three activities: *request for quote* (BTA1, 011), *register customer* (BTA2, 018), and *order product* (BTA3, 025). Additionally, it contains a *decision* state (D1, 078).

The collaboration starts off with *request for quote*. Thus, the start element (008) points to this business transaction activity element (BTA1). In case that this transaction results in a failure and no quote is made, no matter whether this is due to a protocol failure or a business failure, the collaboration is terminated. This is specified by a failure element (E2, 036). A business success of *request for quote*, which sets the object *product* into state *quoted*, results in a transition (048) from *request for quote* (BTA 1) to the decision node (D1). The decision (078) is based on whether the object *customer information* is in state *confirmed* or not. If so, it results in a transition (056) from the decision (D1) to *register customer* (BTA2), and in a transition (063) from decision (D1) to *order product* (BTA3) otherwise.

A business success of *register customer*, i.e. the *customer information* is in state *confirmed*, leads to a transition (070) from *register customer* (BTA2) to *order product* (BTA3). In case of any failure of *register customer*, the *purchase order man-*

agement is terminated. The failure element (E3, 040) defines the transition from *register customer* (BTA2) to the end state. Similarly, any failure of *order product* terminates the binary collaboration. The transition from *order product* (BTA3) to the failure state is defined by another failure element (E4, 044). Only if *order product* results in a business success, i.e. the *product* is in state *ordered*, the *purchase order management* collaboration results in an overall business success. A success element (E1, 032) defines this transition from *order product* (BTA3) to the successful end state.

```
001<BinaryCollaboration
002       nameID="BC1" name="PurchaseOrderManagement"
003       initiatingRoleIDREF="R1" isInnerCollaboration="false"
004       pattern="nopatterns exist yet" timeToPerform="P4D"
005       preCondition="X" beginsWhen="X" endsWhen="X" postCondition="X">

006           <Role nameID="R1" name="buyer" />
007           <Role nameID="R2" name="seller"/>

008           <Start
009               nameID="S1"
010               toBusinessState="request for quote" toBusinessStateIDREF="BTA1"/>

011           <BusinessTransactionActivity
012               nameID="BTA1" name="request for quote"
013               businessTransaction="request for quote" businessTransactionIDREF="BT1"
014               fromRole="buyer" fromRoleIDREF="R1"
015               toRole="seller" toRoleIDREF="R2"
016               isConcurrent="true" timeToPerform="P1D"
017               preCondition="X" beginsWhen="X" endsWhen="X" postCondition="X "/>
018           <BusinessTransactionActivity
019               nameID="BTA2" name="register customer"
020               businessTransaction="register customer" businessTransactionIDREF="BT2"
021               fromRole="buyer" fromRoleIDREF="R1"
022               toRole="seller" toRoleIDREF="R2"
023               isConcurrent="false" timeToPerform="PT12H"
024               preCondition="X" beginsWhen="X" endsWhen="X" postCondition="X"/>
025           <BusinessTransactionActivity
026               nameID="BTA3" name="order product"
027               businessTransaction="order product" businessTransactionIDREF="BT3"
028               fromRole="buyer" fromRoleIDREF="R1"
029               toRole="seller" toRoleIDREF="R2"
030               isConcurrent="true" timeToPerform="P2D"
031               preCondition="X" beginsWhen="Xt" endsWhen="X" postCondition="X"/>

032           <Success
033               nameID="E1"
034               fromBusinessState="order product" fromBusinessStateIDREF="BTA3"
035               conditionGuard="BusinessSuccess"/>
036           <Failure
037               nameID="E2"
038               fromBusinessState="request for quote" fromBusinessStateIDREF="BTA1"
039               conditionGuard="Failure"/>
040           <Failure
041               nameID="E3"
042               fromBusinessState="register customer" fromBusinessStateIDREF="BTA2"
043               conditionGuard="Failure"/>
044           <Failure
045               nameID="E4"
046               fromBusinessState="order product" fromBusinessStateIDREF="BTA3"
047               conditionGuard="Failure"/>

048           <Transition
049               nameID="T1"
050               fromBusinessState="request for quote" fromBusinessStateIDREF="BTA1"
051               toBusinessState="Decision1" toBusinessStateIDREF="D1"
052               conditionGuard="BusinessSuccess">
053                   <ConditionExpression expressionLanguage="OCL"
054                       expression="Product.oclInState(Quoted) = TRUE"/>
055           </Transition>
056           <Transition
057               nameID="T2"
058               fromBusinessState="Decision1" fromBusinessStateIDREF="D1"
059               toBusinessState="register customer" toBusinessStateIDREF="BTA2" >
060                   ConditionExpression expressionLanguage="OCL"
061                       expression="CustomerInformation.oclInState(Confirmed) = FALSE"/>
062           </Transition>
```

```
        <Transition
            nameID="T3"
            fromBusinessState="Decision1" fromBusinessStateIDREF="D1"
            toBusinessState="order product" toBusinessStateIDREF="BTA3" >
                <ConditionExpression expressionLanguage="OCL"
                    expression="CustomerInformation.oclInState(Confirmed) = TRUE"/>
        </Transition>
        <Transition
            nameID="T4"
            fromBusinessState="register customer" fromBusinessStateIDREF="BTA2"
            toBusinessState="order product" toBusinessStateIDREF="BTA3"
            conditionGuard="BusinessSuccess">
                <ConditionExpression expressionLanguage="OCL"
                    expression="CustomerInformation.oclInState(Confirmed) = TRUE"/>
        </Transition>

        <Decision
            nameID="D1" name="Decision1" >
                <ConditionExpression expressionLanguage="OCL"
                    expression="CustomerInformation.oclInState(Confirmed)"/>
        </Decision>
</BinaryCollaboration>
```

4.5 Complex choreographies in UMM and BPSS

Our purchase order management example specifies a rather simple choreography. In real world situations a choreography will be much more complex. In this subsection we analyze the expressive power of BPSS in order to handle complex choreographies. This analysis is based on well-known workflow patterns. Aalst *et al.* propose the pattern based approach for comparison of the expression powers of commercial workflow management systems (Van der Aalst *et al.*, 2003). These patterns have also been used to analyze the expression power of UML activity diagrams (Dumas and Ter Hofsteded, 2001). The patterns are categorized into six classes - basic control patterns, advanced branching and synchronization patterns, structural patterns, patterns involving multiple instances, state-based patterns, and cancelation patterns. In total, 20 patterns have been identified. Table 1 summarizes the workflow patterns that UMM and BPSS support. Support is shown as +. - means that UMM/BPSS does not support the pattern. +/- means that a pattern is partially supported. If the pattern is not implemented by activity structures, but is realized by tagged values, we denote this fact by the character *t*. Furthermore, those patterns that are supported by BPSS are depicted in Figure 9.

The first class of patterns we consider are the basic control patterns. These patterns are *sequence, parallel split, synchronization, exclusive choice*, and *simple merge*. In general, they are similar to definitions of elementary control flow concepts provided by WfMC (WfMC, 1999). **Sequence** pattern (Figure 9a) means that an activity starts after the completion of its predecessor. This is the most simple case. **Parallel split** pattern (Figure 9b) is the behavior of an AND-fork. This means a single activity has multiple successors that occur in parallel. In UMM the pseudo state *fork* is used to realize this pattern. In BPSS there exists a *fork* element as well. However, its *type* attribute must be set to *OR*. **Synchronization** pattern (Figure 9b) forms an antithesis to the parallel split pattern. A successor starts only if all its predecessors are completed. In UMM this pattern is realized by the pseudo state *join*. The graphical notation of *join* is identical to that of *fork* and is often called *synchronization bar* due to its shape. The *join* element exists in BPSS as well. It is important to set its *wait for all* attribute to *true*. **Exclusive choice** pattern (Figure 9c) chooses only one transition from several alternatives based on a decision. It is realized in UMM by the pseudo state *decision* and mutually exclusive guards on the transitions from the *decision* state to the alternatives. In BPSS, the element *decision* fulfills the same function as the *decision* state in UMM. **Simple merge** pattern (Figure 9c) makes an antithesis of the exclusive choice pattern. Multiple

	UMM v. 12	BPSS v. 1.10
Sequence	+	+
Parallel Split	+	+
Synchronization	+	+
Exclusive Choice	+	+
Simple Merge	+	+
Multi Choice	+	+
Synchronizing Merge	+	+
Multi Merge	-	-
Discriminator	-	+
Arbitrary Cycles	+/-	+
Implicit Termination	t	+
MI without Synchronization	t	+
MI with a Priori Design Time Knowledge	-	-
MI with a Priori Runtime Knowledge	-	-
MI without a Priori Runtime Knowledge	-	-
Deferred Choice	+	+
Interleaved Parallel Routing	-	-
Milestone	t	+
Cancel Activity	-	-
Cancel Case	+	+

Table 1 Comparison of UMM and BPSS

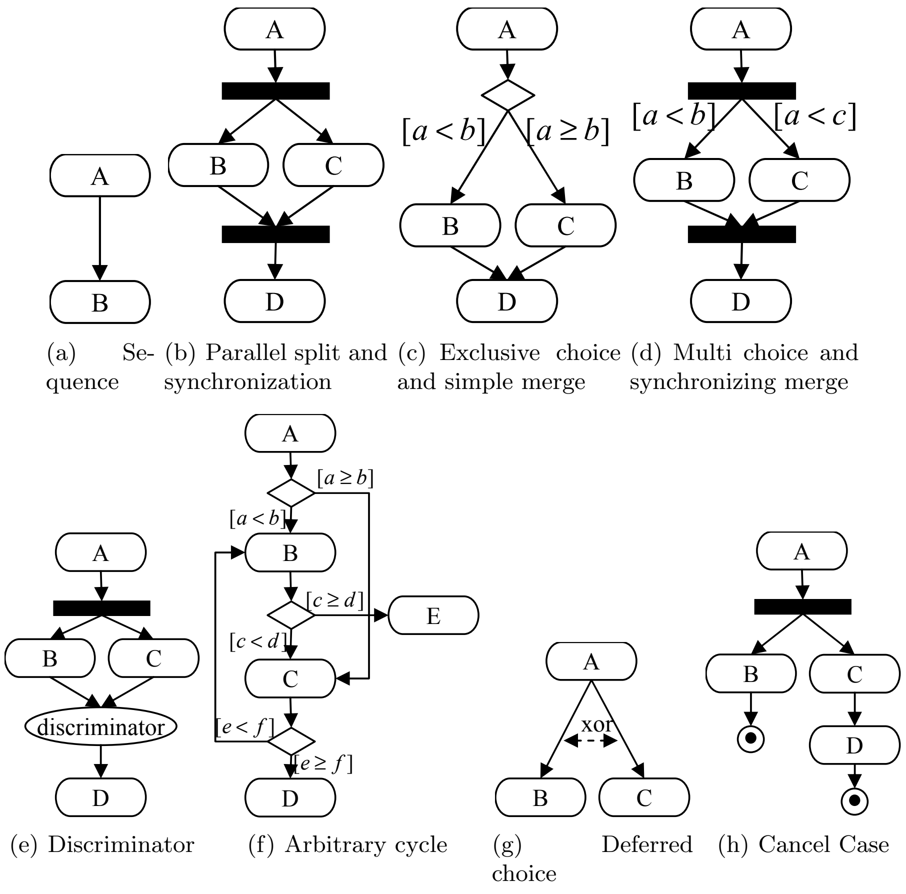

Fig. 9 Patterns supported by BPSS.

branches of which only one can be active due to a previous exclusive choice are merged into a single activity. There is no need for synchronization. This pattern is realized in UMM and BPSS by direct transitions from the last activity of each branch to the single successor. Only in UMM a merge state (notation is identical to the decision state) might be alternatively used.

The second category are the advanced branching and synchronization patterns. **Multi choice** pattern (Figure 9d) allows one or more threads to continue according to given conditions. In UMM the realization is similar to the parallel split. However, conditions must guard the transitions from the fork to a successors. BPSS uses the same concept. **Synchronizing merge** pattern (Figure 9d) is the antithesis of the multiple choice. Those branches which fulfilled the conditions of the multi choice converge into one continuing activity. In UMM and in BPSS the realization is identical to the synchronization pattern, because it is assumed that the merge pseudo state knows which branches have started and does not wait for others. **Multi merge** pattern merges two threads into one continuing activity. It is executed whenever a precedence thread reaches the multi merge pattern. In UMM it is not supported, since forks and joins must be well-nested. BPSS does not support the multi merge pattern either. **Discriminator** pattern (Figure 9e) is similar to the synchronization pattern since multiple threads converge to one tread and the following thread is executed only once. However, the continuing activity starts after the first preceding thread finishes. Again, this pattern is not supported by UMM, because there is no semantically equivalent pseudo state. In BPSS such a pseudo state exists. The pattern is realized by a *join* element, whose *wait for all*'s value is *false*. To depict this pattern in Figure 9e we extend the UML notation by a pseudo state called discriminator. **Arbitrary cycles** pattern (Figure 9f) means that one or more activities are repeated without any structural restrict, i.e. a cycle might have multiple entries and exists. Some arbitrary cycles are constructed by the combination of multiple *decisions*, *xor-typed forks* and *transitions*. In this case UMM and BPSS are able to realize the arbitrary cycle. However, arbitrary cycles might involve forks and joins as well. Since each fork has a corresponding join, transitions can not cross the boundary of the fork-join-block, UMM does not fully support the arbitrary cycle pattern. Since BPSS does not include a similar well-formedness rule it fully supports the arbitrary cycle. **Implicit termination** pattern (Figure 9g) means that no activity is performed anymore, although no deadlock exists and no end state is reached. In UMM a business collaboration protocol usually results in a final state. However, UMM supports a special kind of implicit termination. A business collaboration protocol has a tagged value *time to perform*. This means the business collaboration protocol is terminated by this time even if no final state is reached. BPSS uses a *time to perform* attribute for binary collaborations as well.

Patterns involving multiple instances are **multiple instances without synchronization**, **multiple instances with a priori design time knowledge**, **multiple instances with a priori run time knowledge**, and **multiple instances without a priori run time knowledge**. *Multiple instances with a priori design time knowledge* and *multiple instances with a priori run time knowledge* restrict the number of instances at design time and run time, respectively. In contrary, *multiple instances without synchronization* and *multiple instances without a priori run time knowledge* have no limitation on the number of instances. *Multiple instances without a priori run time knowledge* can manage the relationship among instances such as synchronization differently from *multiple instances without synchronization*. BPSS supports only the *multiple instances without synchronization* by assigning *true* to a *business transaction activity*'s attribute *is concurrent*. Since the activity diagram of UMM does not directly support this pattern, *is concurrent* is expressed as a tag value of an activity.

If the execution of one activity depends on the state of another activity, the pattern is categorized into the class of state-based patterns. State-based patterns include *deferred choice, interleaved parallel routing* and *milestone.* **Deferred choice** pattern (Figure 9h) selects only one continuing activity from several candidates like exclusive choice, but the decision is implicit. In UMM this pattern is not directly supported. However, it is possible to specify XOR-dependencies among the transitions to the alternatives. In BPSS, a corresponding pseudo state element for the deferred choice exists. BPSS realizes this pattern using the element *fork* whose *type* attribute is *XOR.* **Interleaved parallel routing** pattern defines the execution of a set of activities in an arbitrary order. Each activity of the set is executed once. At a given point in time only one activity is executed. The execution order is fixed at run time. Neither UMM nor BPSS support the interleaved parallel routing pattern. In the **Milestone** pattern the start of an activity depends on the state of other activities. In UMM activities usually result in certain post-conditions, i.e. business objects change to a certain state. Furthermore, activities might have pre-conditions that require business objects to be in a certain state. Thus, the milestone pattern is realized in UMM by a combination of the tagged values for *pre-* and *post-condition.* BPSS also uses attributes for *pre-* and *post-condition.*

A **Cancel activity** pattern cancels an enabled activity. UML supports through transition with triggers. However, UMM does not use this feature and BPSS does not support this pattern directly, either. A **cancel case** pattern terminates a *binary collaboration.* In UMM (BPSS), as soon as a *final* state (a *success* or a *failure* element) is reached, the *binary collaboration* is terminated. Even if other *business transaction activities* remain, they do not open any more. In this case, a timeout exception can be generated. Therefore, although a *binary collaboration* has several *final* states, they should be mutually exclusive.

5 Summary

This paper presented an introduction into the choreography of ebXML business collaborations. ebXML has delivered a modular set of specifications for conducting e-business between companies. Inasmuch ebXML looks after business processes in a B2B environment. It does not consider the internal processes of the business partners. It concentrates on the business processes crossing the borders of the involved business partners. This type of collaborative business process is called business collaboration. ebXML business collaborations should be modeled by the UN/CEFACT' modeling methodology (UMM) and are expressed as instances of the ebXML business process specification schema (BPSS). Thus, an analysis of the expressive power of these two standards builds the core of this paper.

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 the UMM 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 models.

We did not go into all the details of UMM. Rather we concentrated on those UMM artifacts that BPSS is based upon: business transactions and their choreography into business collaborations. We presented the underlying UMM meta model and the resulting BPSS schema definition for both artifacts. An example of a over-simplified purchase order management collaboration was used to express the same business semantics in UMM and BPSS. It became obvious that UMM and BPSS usually 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.

Another goal of this paper was to identify all the business semantics needed to describe the choreography of a B2B process. Both UMM and BPSS have been specifically developed for this purpose and cover best practice approaches from industry. Both standards might not be the best choice for describing private business processes, and, thus, BPSS might not serve as a general business process interchange format. In contrary, any alternative business process interchange format used to describe B2B collaborations must support the UMM and BPSS concepts introduced in this paper.

References

Booch G, Jacobson I, Rumbaugh J (1998) The Unified Modeling Language User Guide. Addison Wesley Object Technolgy Series

Dumas M, Ter Hofsteded AHM (2001) Uml activity diagrams as a workflow specification language. In: Proceedings of the 4th International Conference on the Unified Modeling Language (UML): Modeling Languages, Concepts, and Tools. Springer LNCS 2185. pp. 76 – 90

Hammer M, Champy J (1993) Reengineering the Corporation: Manifesto for Business Revolution. Harper Business

Hill NC, Ferguson DM (1989) Electronic data interchange: A definition and perspectivie. EDI Forum: The Journal of Electronic Data Interchange 1(1): 5 – 12

Hofreiter B, Huemer C (2003) Modeling business collaborations in context. In: On The Move to Meaningful Internet Systems 2003: OTM 2003 Workshops. Springer

Hofreiter B, Huemer C (2004) Transforming umm business collaboration models to bpel.. In: OTM Workshops. Springer LNCS 3292. Agia Napa, Cyprus. pp. 507–519

Hofreiter B, Huemer C, Klas W (2002) ebXML: Status, research issues and obstacles. In: Proc. of 12th Int. Workshop on Research Issues on Data Engineering (RIDE02). San Jose

Kindler E (2004) Using the Petri Net Markup Language for Exchanging Business Processes? Potential and Limitations. In: Proceedings of the 1st GI Workshop XML4BPM – XML Interchange Formats for Business Process Management at 7th GI Conference Modellierung 2004, Marburg Germany, March 2004. pp. 43–60

Kotok A (2000) Even more extensible. http://webservices.xml.com/pub/a/ws/2000/08/02/ebiz/extensible.html

Leymann F, Roller D (2004) Modeling Business Processes with BPEL4WS. In: Proceedings of the 1st GI Workshop XML4BPM – XML Interchange Formats for Business Process Management at 7th GI Conference Modellierung 2004, Marburg Germany, March 2004. pp. 7–24

Li H (2000) XML and industrial standards for electronic commerce. Knowledge and Information Systems 2(4): 487 – 497

Mendling J, Nüttgens M (2004) Exchanging EPC Business Process Models with EPML. In: Proceedings of the 1st GI Workshop XML4BPM – XML Interchange Formats for Business Process Management at 7th GI Conference Modellierung 2004, Marburg Germany, March 2004. pp. 61–80

Schatz W (1988) EDI: putting the muscle in commerce and industry. Datamation 34(6): 56 – 64

ISO (1997) Open-edi Reference Model. ISO/IEC JTC 1/SC30 ISO Standard 14662. ISO

OASIS (2003) Business process execution language for web services. ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf

UN/CEFACT TMG (2003 a) Core components technical specification v2.01. http://www.untmg.org/downloads/General/approved/CEFACT-CCTS-Version-2pt01.zip

UN/CEFACT TMG (2003 b) ebXML business process specification version 1.10. http://www.untmg.org

UN/CEFACT TMG (2003 c) UN/CEFACT modeling methodology meta model, revision 12. http://www.untmg.org/downloads/General/approved/UMM-MM-V20030117.zip

UN/CEFACT TMG (2003 d) UN/CEFACT modeling methodology user guide, revision 12. http://www.untmg.org/downloads/General/approved/UMM-UG-V20030922.zip

UN/CEFACT, OASIS (2001) ebXML techinical architecture specification, version 1.0.4. http://www.ebxml.org/specs/ebTA.pdf

W3C (2002 a) Web service choreography interface 1.0. http://www.w3.org/TR/wsci/

W3C (2002 b) Web services conversation language 1.0. http://www.w3.org/TR/wscl10/

W3C (2004) Web services choreography description language 1.0. http://www.w3.org/TR/ws-cdl-10/

WfMC (1999) Workflow management coalition terminology & glossary. Technical Report WFMC-TC-1011. WfMC

Van der Aalst AMP, Ter Hofsteded AHM, Kiepuszewski B, Barros AP (2003) Workflow patterns. Distributed and Parallel Databases 14: 5 – 51

Winter A, Simon C (2004) Exchanging Business Process Models with GXL. In: Proceedings of the 1st GI Workshop XML4BPM – XML Interchange Formats for Business Process Management at 7th GI Conference Modellierung 2004, Marburg Germany, March 2004. pp. 103–122