Modeling Business Entity State Centric Choreographies

Christian Huemer, Marco Zapletal
Institute of Software Technology
Vienna University of Technology
Favoritenstr. 9-11/188, 1040 Vienna, Austria
huemer@big.tuwien.ac.at, marco@ec.tuwien.ac.at

Philipp Liegl, Rainer Schuster
Austrian Research Centers GmbH - ARC
Research Studios Austria
Thurngasse 8/20, 1090 Vienna, Austria
{pliegl, rschuster}@researchstudio.at

Abstract

In a B2B environment business partners interact with each other by exchanging electronic business documents. The agreements and commitments between the partner require a certain order in the flow of business documents. This flow - commonly known as choreography - often depends on the actual content of a business document. E.g. the next step depends on whether a price was stated in a quote document or not in the step before. These characteristics usually affect the states of a business entity - whether a quote is in state provided or in state refused. The states of a business entity usually define the next steps in the choreography of a collaborative business process. Thus, it is important that the actual business document content and resulting business entity states are unambiguously defined in a global choreography. In this paper we show how the modeling of business entity state centric choreographies may be incorporated into UN/CEFACT's modeling methodology (UMM) - one of the best known approaches to model global choreographies.

1. Introduction

Exchanging business documents between business partners is not particularly new. It has been implemented for a long time by the concepts of Electronic Data Interchange (EDI) [4]. EDI standards like UN/EDIFACT and ANSI X12 standardized the business document types. Different document types are exchanged in an agreed order to implement a business case electronically. Usually, business partners sign a trading partner agreement in order to establish an electronic partnership. This trading partner agreement covers the definition of a subset of the EDI document standards and the flow of business documents in addition to the legal terms of the partnership. In absence of a formal description for the document flow in EDI standards, the agreed order of the document types is described in plain text.

Later on XML-based business document standards provided an alternative to traditional EDI standards. An overview of XML business document standards is provided by Li [13]. These standards are also limited to the format of the business documents and do not consider the document flow. However, XML also led to more advanced technologies like Web Services and ebXML that may be used to exchange business documents of a certain (XML-based) business document standard. Both, Web Services and ebXML include specifications for describing the flow of business documents, e.g. BPEL [1], WS-CDL [22], ebXML BPSS [20].

The flow of documents may be described from the perspective of a participating partner or from a neutral perspective. In the first case we talk about a local choreography and in the second case of a global choreography. BPEL may be used to describe a local choreography, whereas WS-CDL and ebXML BPSS capture a global choreography. A global choreography is usually used to define a contractual agreement on the document flow between the business partners. In contrary, a local choreography specifies the flow of documents as expected at the interface of the a business partner's information system. This means that if a buyer and a seller want to collaborate they agree on a single global choreography and each one derives its own local choreography as an interface to its own information system.

In this paper we concentrate on global choreographies. We prefer a graphical modeling language to specify a global choreography. A visual representation of the agreed document flow enables a better understanding and verification by the business partners. Furthermore, modeling languages are able to abstract from the IT-platform - such as Web Services, ebXML and what ever comes up in the future - in order to survive technological changes. Modeling of choreographies is based on three main concepts according to Dubray [2]: event, activity and state. Furthermore, he analyzes that most approaches have focused on events (pi-calculus) or activities (petri nets, activity diagrams), but hardly any have focused on states, in particular business entity states.

In this paper we demonstrate the importance of business

The 9th IEEE International Conference on E-Commerce
Technology and The 4th IEEE International Conference
on Enterprise Computing, E-Commerce and E-Services(CEC-EEE 2007)
0-7695-2913-5/07 $25.00 © 2007 IEEE

entity states in modeling public choreographies. We build upon UN/CEFACT's modeling methodology (UMM) [21], which we have co-authored. UMM is based on UML and uses activity graphs to model a global choreography. After a basic introduction into UMM in section 2, we present our approach to incorporate business entity states into UMM in section 3. In this section we first illustrate that a UMM choreography may be well interpreted by a human, but fails to give an unambiguous machine-processable definition to further derive software artifacts. Next, we show how to overcome these limitations by considering business entity states. We detail how business entity states are set within transactions by the characteristics of the business document instances and how the resulting business entity states guard the long-running business transactions. In section 4 we discuss related work and we conclude with a short summary in section 5.

2. Global Choreographies in UMM

In this section we give a brief introduction into the UMM concepts to model a business choreography. UMM is defined as a UML profile for the special purpose of modeling business collaborations [21] [8]. This means it defines stereotypes, their tagged values and constraints on top of the UML meta model in order to define inter-organizational business processes. It is independent of any protocols used to implement the business collaborations. Nevertheless, it is envisioned that UMM business collaboration models are transformed to certain Web Services protocols [5] or ebXML protocols [6].

UMM is a modeling methodology capturing the commitments and agreements between business partners in a business collaboration. Thereby, the methodology follows a number of different steps that are organized in three main views each consisting of subviews. The business domain view (BDV) is used to gather a basic understanding of the business domain. The business requirements view (BRV) elaborates on existing and envisioned business processes and derives the requirements of a business collaboration. Finally, the business transaction view leads to an agreed choreography of business document exchanges that follows the commitments and agreements discovered in the previous steps. In this section we concentrate on the artifacts of the business transaction view. The reader interested in all the UMM steps is referred to our paper [8]. We explain the concepts of the business transaction view by the means of an over-simplified example of an `order from quote` that still allows the introduction of the relevant concepts.

2.1. Business Transactions

In UMM the overall choreography of a business collaboration is built by a composition of multiple interactions between business partners. These interactions are called *business transactions*. They serve as basic building blocks for modeling a business choreography. The aim of a *business transaction* is aligning the business information systems of the collaborating business partners. This means that the *business transaction* is responsible for keeping all relevant business entities in the same state in each information system.

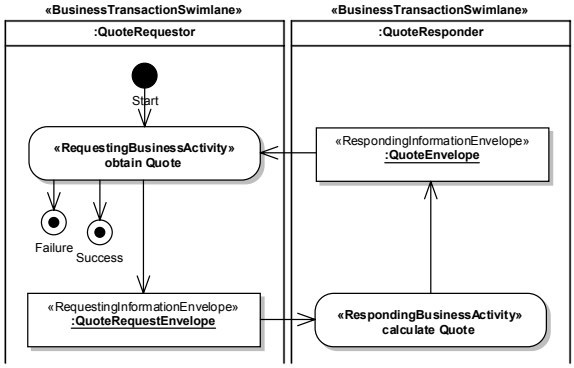

Figure 1. Business transaction

A *business transaction* is an atomic unit responsible for the synchronization between exactly two business partners' information systems. This synchronization process happens either in a uni-directional way (one-way transaction) or bi-directional (two-way transaction). The first one describes an information flow from the initiating to the reacting partner provoking an irreversible state change of the initiator, which the responder has to accept. In the second type the initiating business partner sets the business entity into an interim state, till the response of the reacting business partner triggers a change to the final state.

A *business transaction* is represented by a UML activity graph. This graph always follows the same pattern. The activity graph of a *business transaction* consists of exactly two *business transaction swimlanes* each representing an *authorized role*. Each *authorized role* performs exactly one activity. While the object flow from the *requesting* to the *responding business activity* is mandatory, the object flow vice versa is optional. The exchanged information is represented by a *requesting* or by a *responding information envelope*. The static structure of these envelopes and business documents included therein is described in the next subsection in more detail.

Furthermore, UMM distinguishes between two one-way (information distribution, notification) and four two-way (query/response, request/response, request/confirm, com-

The 9th IEEE International Conference on E-Commerce
Technology and The 4th IEEE International Conference
on Enterprise Computing, E-Commerce and E-Services(CEC-EEE 2007)
0-7695-2913-5/07 $25.00 © 2007 IEEE

mercial transaction) types of *business transactions*. Each type differs in the default values of the tagged values characterizing a *requesting/responding business activity*: *is authorization required*, *is non-repudiation required*, *time to perform*, *time to acknowledge receipt*, *time to acknowledge acceptance*, *is non-repudiation of receipt required* and *retry count*. These tagged values are considered as self-explanatory. For further clarification please refer to the specification [21]. These business transaction types and their tagged values have proven to be useful in RosettaNet [18], because they cover all known legally binding interactions between two decision making applications as defined in the Open-edi reference model [10].

Figure 1 shows an example of a two-way *business transaction* called `request for quote`. In this example the initiator requests a `quote`, that either may be provided by the reacting role or not. The `quote requestor` takes on the role of the initiator and the `quote responder` performs the reacting role. The `quote requestor` starts an `obtain quote` activity and creates a `quote request envelope` that triggers the `calculate quote` activity of the `quote responder`. Having reached this point of the *business transaction*, the `obtain quote` activity is still alive, because it is waiting for a response of the `quote responder`. Afterwards the `calculate quote` activity generates a `quote envelope` in order to return it to the `obtain quote` activity. In the last step the `obtain quote` activity processes the response and sets its final state either to `failure` or to `success`.

2.2. Business Documents

As outlined in the previous subsections, a mandatory *requesting information envelope* and an optional *responding information envelope* are exchanged in the course of a *business transaction*. The modeling of these envelopes and the business documents included therein follows the so-called core components standard [19]. This standard provides core components as basic building blocks that may be re-used for modeling business documents. The core components standard defines its own - non-UML based - meta model. For the purpose of integrating the core components methodology into UMM, the UML profile for core components [9] maps the concepts of the core components standard to the UML meta model.

Core components are split up into three distinctive groups: *basic core components*, *aggregate core components* and *association core components*. *Basic core components* represent atomic values such as postal code or name and are consolidated in *aggregate core components*. An *aggregate core component* contains *basic core components* and could for instance be an address. An *association core component* can be used to model the dependencies between different

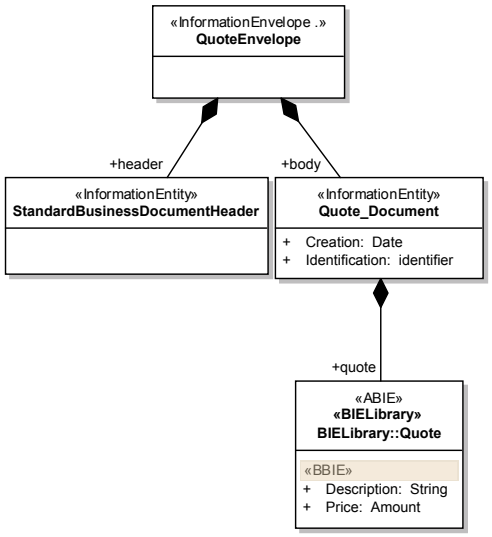

Figure 2. Business documents

aggregate core components i.e. between a person and an address. If core components are used in a certain business context, they are called *business information entities*. Core components and business information entities are stored in libraries facilitating their reuse in different application scenarios. We have already given a thorough insight into the core components standard in [9].

Figure 2 shows a `quote` document type, which is exchanged during the *business transaction* `request for quote`. In order to emphasize the relevant concepts of our approach, the business document model is kept rather simple. We assume that the quote request and the resulting quote are made for exactly one product. In addition, we limit the information to what is really necessary to accomplish a business entity state change. This conforms to the idea of a stateful process modeling approach.

The *information envelope* `quote envelope` consists of two *information entities* `standard business document header` and `quote document`. The *header* contains auxiliary information about the business document and the *body* holds the actual business information. Both stereotypes - *information entity* and *information envelope* - are part of the UMM standard.

The *business information entity* `quote`, being part of the core components library, consists of a `description` and a `price`. The `price` is of type `amount` which is not shown in the figure. `Amount` is a complex type holding additional information such as currency details whereas `description` is a simple string. The package name of the element `quote` (separated by two colons) already indicates, that `quote` is coming from a different package - in this case from a *business information entity library* called `BIELibrary`. Busi-

The 9th IEEE International Conference on E-Commerce
Technology and The 4th IEEE International Conference
on Enterprise Computing, E-Commerce and E-Services(CEC-EEE 2007)
0-7695-2913-5/07 $25.00 © 2007 IEEE

ness information entities are created in specific libraries from which they can be retrieved for usage in specific application scenarios.

2.3. Business Collaboration Protocols

According to the definition of Hammer and Champy [3] a business process is an organized group of related activities that together create customer value. A business collaboration is a special kind of a business process involving multiple partners, i.e. an inter-organizational business process. As we learned in the previous subsection, *business transactions* are the basic building blocks for a business collaboration. A choreography of a business collaboration defines the execution order between business transactions and/or nested business collaborations.

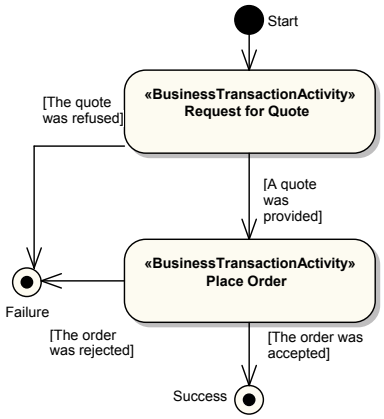

Figure 3. Business collaboration protocol

In UMM the choreography of a business collaboration is modeled by the means of an activity graph called *business collaboration protocol*. This activity graph models a long-running business transaction between the business partners. All activities in the *business collaboration protocol* are either stereotyped as *business transaction activities* or *business collaboration activities*. A *business transaction activity* is refined by the activity graph of a *business transaction* (c.f. subsection 2.1). A *business collaboration activity* is further detailed by the activity graph of a nested *business collaboration protocol*.

A *business collaboration protocol* specifies the choreography by means of transitions between the *business transaction activities* and/or *business collaboration activities*. A transition may be guarded according to the results reached in the business collaboration so far. The *business collaboration protocol* uses the the following pseudo states as defined in UML 1.4. in order to specify complex flows: *decision* and *merge*, *fork*, and *join*.

Figure 3 depicts the *business collaboration protocol* of our `order from quote` example. The first *business transaction activity* is `request for quote`. It is refined by the homonymous *business transaction* we illustrated in subsection 2.1. If a `quote` was refused during this *business transaction*, the business collaboration immediately ends with a `failure`. Conversely, if a `quote` was provided, the choreography continues with a `place order` *business transaction activity*. This activity is refined by the `place order` *business transaction*, which we do not detail in this paper. It should be noted, that it follows the strict pattern of any *business transaction* as described in subsection 2.1. If an `order` is accepted during the `place order` *business transaction*, the `order from quote` *business collaboration protocol* ends with a success. Otherwise - if the `order` is rejected - it ends with a failure.

3. A Business Entity State Centric Choreography

3.1 Ambiguities in the current UMM

According to the UMM introduction in the previous section we recognize interdependencies between *business collaboration protocols*, *business transactions* and *business documents*. The content of a *business document* leads either to a success or a failure of the *business transaction*. The *business collaboration protocol* uses guards on its transitions that depend on a success or failure of *business transactions*.

In the current UMM approach we see the following three problems:

- The same business document type is used for different business intentions. The actual business document content, which is not considered by UMM, is needed to reveal the business intention.
- The business intention leads to a success or failure of a *business transaction*. This means the actual business document content decides on the end state of a *business transaction*.
- The *business collaboration protocol* is guarded by the outcome of *business transactions*. However, there is no explicit reference between the transition guards and the business transaction end states.

We further illustrate these problems by means of our `order from quote` example. The `request for quote` *business transaction* always returns a `quote envelope` (see figure 1. The format of the `quote envelope` depicted in figure 2 is always the same regardless of the business intention. The business intention may be either to provide a quote by instantiating the attribute `price` of class `quote`

or to refuse a quote by not instantiating this attribute. This means, that the actual content of the business document - whether the attribute `price` is stated or not - is needed to perceive the business intention. Depending on this intention the *business transaction* in figure 1 results in a success or a failure. It follows, that the current UMM standard does not unambiguously define how to reach a certain end state of a *business transaction*. This can only be perceived by humans who interpret the business intention. This is due to the fact that the current UMM standard provides no means to formally specify which type of information must be stated in a business document in order to lead to a successful *business transaction*.

Similarly, the result of the `request for quote` *business transaction* influences the control flow of the `order from quote` *business collaboration protocol* shown in figure 3. A success of the *business transaction* leads to the `place order` *business transaction activity*, whereas a failure of the *business transaction* leads also to a failure in the *business collaboration protocol*. Although UMM uses guards on the transitions, it does not mandate any formalism on these guard conditions. Guards like `'A quote was provided'` in figure 3 do not have any traceability to the end states of a *business transaction*. Accordingly, such guards may only be interpreted by humans.

Since the UMM choreography requires interpretation by humans, UMM only provides a graphical notation supporting the human negotiations of agreements and commitments on the business document flows. Currently, it is not suited for deriving unambiguous protocols - such as BPEL - used to configure supporting software. In the following subsections we present a solution to overcome the shortcomings outlined above.

3.2. Implicit State Changes by Business Documents

The first requirement to overcome the ambiguities in UMM is to derive the business transaction end states from the exchanged business documents. A first option is to use different business document types for different business intentions. With respect to our example, this means to use a different type for a positive response and for a negative response. However, this would result in a proliferation of business document types, especially when more complex business documents are considered. Furthermore, this would not correspond to the current practice in business document standards, which e.g. use only one `quote` document type regardless of the business intention. The second option is to include explicit flags within a business document type signaling the business intention. Again, this is in contradiction to current business document standards that do not include these kind of flags. The third option - which is our preferred one - is to put OCL constraints on the business document types which guard the results of a *business transaction*.

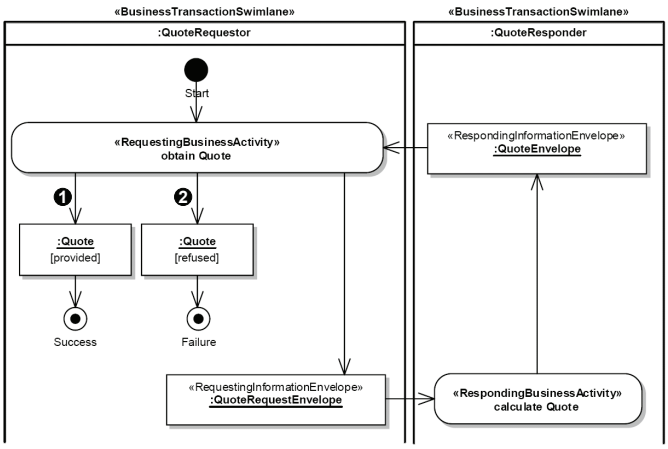

Figure 4. Business transaction with business entity states

We demonstrate the incorporation of OCL constraints by the means of the `request for quote` *business transaction* and the `quote envelope`. If the `price` attribute in figure 2 has a value, `request for quote` in figure 4 is successful - if not, it fails. We use OCL constraints to formally represent the logic needed to determine, whether the request was successful or not. Accordingly, we define an invariant for a positive `quote` in listing 1 that checks that the `price` attribute in class `quote` is not empty. Similarly, we define an invariant for a negative `quote` in listing 2, which checks a null value of the `price` attribute.

Listing 1. OCL constraint 1

```
context QuoteEnvelope
inv quoteProvided:
   self.body.quote.Price->notEmpty()
```

Listing 2. OCL constraint 2

```
context QuoteEnvelope
inv quoteRefused:
    self.body.quote.Price->isEmpty()
```

These OCL constraints are used to guard the transitions to final states in the `request for quote` *business transaction* presented in figure 4. The transition denoted by bullet 1 in figure 4 is guarded by the OCL expression in listing 1 and the transition denoted by bullet 2 is guarded by the OCL constraint in listing 2. The *requesting business activity* `obtain quote` receives in return the `quote envelope`. The `obtain quote` activity checks the `quote envelope` to determine which of the mutually exclusive invariants holds. If a `price` is present, the first invariant is

valid and the *business transaction* ends with a success. Otherwise, the second invariant is true leading to a failure of the *business transaction*.

It would also be possible to specify only a constraint for the transition leading to the final state success and use the OCL expression `else` for the other transition or vice versa. In regard to extensibility, however, we suggest to use a separate OCL constraint for each transition.

We have shown, that OCL constraints are an appropriate mean to evaluate the business document content and direct the control flow according to the business intention. The mandatory use of mutually exclusive OCL guards on the business document content leads to unambiguous flows that are understood by humans and machines. This facilitates the automatic processing of a UMM model by artifact generators (e.g. UMM-BPEL transformer as shown in our UMM tool [7]).

3.3. Changing Business Entity States in Business Transactions

In order to reflect the results of *business transactions* in the choreography of *business collaboration protocols*, we first incorporate the concept of *business entities* and their states into UMM *business transactions*. A *business entity* is defined as a real-world thing having business significance that might be shared among two or more business partners in a collaborative business process [21].

In the current UMM, the concept of a *business entity* already exists, but just as a mechanism for requirements elicitation. During the requirements phase, the business analyst gathers *business entities* in the domain of consideration. Capturing *business entities* together with their life cycles fosters the understanding of the business domain as well as the business processes therein. Moreover, in a collaborative process, a state change of a *business entity* indicates a step in the process that requires communication between business partners. In UMM, communication between participants is realized by the means of a *business transaction*. Aligning states in a collaborative environment ensures that all partners have a shared understanding of the progress of the business collaboration. This assures that all participants know exactly how they have to act in each step of the business collaboration.

Figure 5 shows the life cycle for a *business entity* `quote`. In the current UMM, this life cycle was created in the business requirements view, but is not explicitly referenced by any artifacts of the business transaction view. The business entity life cycle defines that a `quote` is at first in state `requested`. Later on it is set either to the state `provided` or to the state `refused`. This corresponds to the intention of the *business transaction* `request for quote`. By sending the `quote request envelope`, the *business entity* `quote` is set into the interim state `requested`. When the quote envelope is returned, the *business entity* `quote` is either set to `provided` or to `refused`. Although this intention seems to be obvious for humans, it is not traceable in the current UMM. Thus, we include the resulting final *business entity states* in the concept of a *business transaction*.

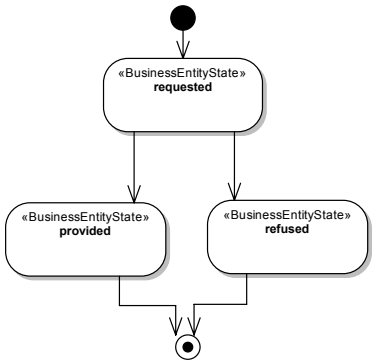

Figure 5. Business entity life cycle

In the previous subsection we introduced an approach that allows the *requesting business activity* to decide whether the *business transaction* ends with a success or with a failure. In fact, the *requesting business activity* sets the corresponding *business entity state* before it reaches the final state. Consequently, we include an *object flow state* that sets the corresponding object state before reaching the final state.

Figure 4 exemplifies this concept by means of the `request for quote` *business transaction*. As already outlined before, the `obtain quote` activity checks the existence of the `price` attribute. If this attribute is instantiated, it sets the *business entity* `quote` to the state `provided` before ending with a success. Otherwise, it sets the state to `refused` before ending with a failure.

3.4. A Business Collaboration Protocol guarded by Business Entity States

In the previous subsection we bound the result of *business transactions* to *business entity states*. In this subsection we use these *business entity states* to govern the flow of a *business collaboration protocol*. This is realized by guarding the transitions between *business transaction activities* by the actual *business entity states*. In the current UMM there is no formal notation mandated to specify guards. For example, the transition between `request for quote` and `place order` in figure 3 is guarded by the plain text `'The quote was provided'`. This certainly reflects the result of the `request for quote` *business transaction*, or better the *business entity state* that was

The 9th IEEE International Conference on E-Commerce
Technology and The 4th IEEE International Conference
on Enterprise Computing, E-Commerce and E-Services(CEC-EEE 2007)
0-7695-2913-5/07 $25.00 © 2007 IEEE

set by this *business transaction*. Since *business entity states* are explicitly modeled in our approach of *business transactions*, we are able to access the actual state. In order to check the state of a *business entity*, OCL provides the function `<Object>.oclInState(theState)`. This function returns true if the object currently is in the specified state. Accordingly, we mandate this function on the guards of the *business collaboration protocol*.

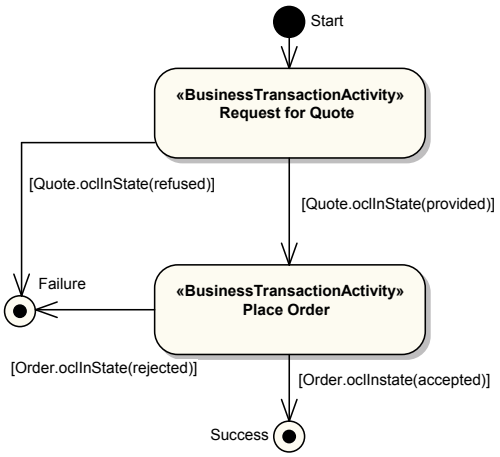

Figure 6. Business collaboration protocol with OCL guards

This leads to the *business collaboration protocol* `order from quote` as depicted in figure 6. The first activity in this choreography is `request for quote`. As outlined in the subsection before, the `request for quote` *business transaction* sets the *business entity* `quote` either into state `provided` or `refused`. The two outgoing transitions from the `request for quote` activity carry mutually exclusive OCL functions checking whether `quote` has been set to one or the other state. The *business collaboration protocol* ends if `quote` is in state `refused`, and continues with *place order* if `quote` is in state `provided`. Similarly, the two outgoing transitions from `place order` are guarded by mutually exclusive OCL functions verifying the state of the *business entity* `order` leading either to an overall success or failure.

4. Related Work

In the field of business process choreographies several research efforts have been undertaken and methodologies have been developed. Some of these methodologies are based on special notations, often maintained by standardization bodies. Others are based on UML by tailoring the UML meta model to the specific needs using UML profiles. Furthermore we identify a third group which can be regarded as a hybrid of the former and latter.

The Business Process Modeling Notation (BPMN) is a standardized modeling notation developed by the Object Management Group (OMG) [16]. The methodology incorporates the benefits of already advanced modeling notations (e.g. UML activity diagrams, IDEF [15], ebXML BPSS [20], RosettaNet [18], etc.). Furthermore OMG's Business Process Management Initiative (BPMI) considers the transition from the business process design to the business process implementation. Thus the mapping from BPMN to an XML based executable business process (e.g. BPEL is part of the standard.

Apart from using standards defined by standardization bodies for modeling business processes, there are some approaches extending the UML meta model for the purpose of process modeling. In particular UML activity diagrams are suitable for depicting the flow of business objects. List and Korherr propose a UML 2 profile for modeling business processes considering business process goals and performance measures [12]. In order to integrate measurable values into a business model, they extend the UML activity diagram with a set of stereotypes and tagged values. These artifacts are then proposed for a mapping to BPEL, in order to execute the business process model and to monitor these values. However, this approach - as well as [14] and [17] - are restricted to the intra-organizational view of an organization.

The concept of defining a global choreography and then deriving local choreographies from the global one has also been examined in a paper by Xiaoquiang and Jun [23]. They propose to establish a process mediation model based on data dependencies which helps to facilitate the adaptation between the local business process and the global choreography. However, their model is missing a methodology to unambiguously define an inter-organizational business process and to capture the requirements for the specific business process.

Another approach for managing B2B business processes in an inter-organizational manner has been proposed by Kim [11]. Kim uses UML 1.x activity diagrams for modeling collaborative processes as a flow of transactions in order to create an ebXML compliant business process specification. Similar to business transactions in UMM, a transaction in this approach is a message exchange between two business partners. These transactions are represented as activities in the activity graph and the flow of data exchanged is then captured in sequence diagrams.

5. Summary

In this paper we presented an approach to modeling global choreographies based on business entity states. Our approach extends UN/CEFACT's modeling methodology

that uses UML to design global choreographies. The UMM is used to specify a contractual flow of business document exchanges that business partners will agree on. However, the flow as specified by the current UMM requires human interpretation. This is due to the fact that there exist ambiguous interdependencies between the business documents, the business transactions and the business collaboration protocol. Business documents are exchanged leading to a business success or a business failure of a business transaction. However, the same business document type is used to signal a positive and a negative response. It is up to human interpretation how the business document content looks like for a positive response and how it looks like for a negative response. Furthermore, the flow among business transactions - which is specified by a business collaboration protocol - depends on the result of these business transactions. Unfortunately, the transition guards between the business transactions do not require any formalism allowing traceability to the transaction results.

In order to overcome these limitations we extend UMM by three concepts. Firstly, we introduce OCL invariants for a positive response as well as for a negative response. An incoming response document is checked against the OCL invariants to determine whether a business transaction succeeds or fails. Secondly, we incorporate the concept of business entity states into business transactions. This means if the positive invariant applies for a response document, the business entity manipulated my the business transaction is set to an explicitly named business entity state before reaching the successful end state. Similarly, accordance with the negative invariant results in another explicitly named business entity state before ending with a failure. Thirdly, these business entity states are checked in the guard conditions of the control flow of a business collaboration protocol by using the corresponding OCL function.

Our UMM extensions lead to an unambiguous choreography that is understood by humans, but is also automatically processable by machines. This is a prerequisite to derive unambiguous platform specific protocols, such as a local choreography for each business partner in BPEL. The transformation of our UMM extension to BPEL is part of our future work.

References

[1] BEA, IBM, Microsoft, SAP AG and Siebel Systems. *Business Process Execution Language for Web Services*, May 2003. Version 1.1.

[2] J.-J. Dubray. Microsoft's Next Frontier. *BPTrends*, 2004. http://www.bptrends.com/publications.cfm.

[3] M. Hammer. *Beyond Reengineering: How the Process-Centered Organization is Changing Our Work and Our Lives*. Collins, 1997.

[4] N. C. Hill and D. M. Ferguson. Electronic Data Interchange: A Definition and Perspective. *EDI Forum: The Journal of Electronic Data Interchange*, 1(1):5–12, 1989.

[5] B. Hofreiter and C. Huemer. Transforming UMM Business Collaboration Models to BPEL. In *Proceedings of OTM Workshops 2004*, volume 3292, pages 507–519. Springer LNCS, 2004.

[6] B. Hofreiter, C. Huemer, and J.-H. Kim. Choreography of ebXML business collaborations. *Information Systems and e-Business Management (ISeB)*, June 2006.

[7] B. Hofreiter, C. Huemer, P. Liegl, R. Schuster, and M. Zapletal. UMM Add-In, Mar. 2006. http://www.ifs.univie.ac.at/ummaddin/.

[8] B. Hofreiter, C. Huemer, P. Liegl, R. Schuster, and M. Zapletal. UN/CEFACT'S Modeling Methodology (UMM): A UML Profile for B2B e-Commerce. In *Advances in Conceptual Modeling - Theory and Practice, ER 2006 Workshops*, pages 19–31. Springer LNCS 4231, 2006.

[9] C. Huemer and P. Liegl. A UML Profile for Core Components and their Transformation to XSD. In *Second International Workshop on Services Engineering, IEEE ICDE Workshops 2007*. IEEE Computer Society, Apr. 2007.

[10] ISO. *Open-edi Reference Model*, 1995. ISO/IEC JTC 1/SC30 ISO Standard 14662.

[11] H. Kim. Conceptual Modeling and Specification Generation for B2B Business Processes based on ebXML. In *SIGMOD Rec., vol. 31, no. 1, pp. 37-42*, 2002.

[12] B. Korherr and B. List. Extending the UML 2 Activity Diagram with Business Process Goals and Performance Measures and the Mapping to BPEL. In *Proceedings of the ER-Conference on Conceptual Modeling (Workshop BP-UML'06)*. Springer, Nov. 2006.

[13] H. Li. XML and Industrial Standards for Electronic Commerce. *Knowl. Inf. Syst.*, 2(4):487–497, 2000.

[14] B. List and B. Korherr. A UML 2 Profile for Business Process Modelling. In *ER 2005 Workshop Proceedings*, 2005.

[15] R. J. Mayer, P. C. Benjamin, B. E. Caraway, and M. K. Painter. A Framework and a Suite of Methods for Business Process Reengineering. *Business Process Reengineering: A Managerial Perspective*, pages 245–290, 1995.

[16] OMG - Object Management Group. *Business Process Modeling Notation Specification 1.0*, 2006.

[17] M. Penker and H.-E. Eriksson. *Business Modeling With UML: Business Patterns at Work*. Wiley, 2000.

[18] RosettaNet. *RosettaNet Implementation Framework: Core Specification*, Dec. 2002. V02.00.01.

[19] UN/CEFACT. *Core Components Technical Specification - Part 8 of the ebXML Framework*, Nov. 2003. Version 2.01.

[20] UN/CEFACT. *UN/CEFACT - ebXML Business Process Specification Schema*, Nov. 2003. Version 1.11.

[21] UN/CEFACT. *UN/CEFACT's Modeling Methodology (UMM), UMM Meta Model - Foundation Module*, Mar. 2006. Technical Specification V1.0, http://www.unece.org/cefact/umm/UMM_Foundation_Module.pdf.

[22] World Wide Web Consortium (W3C). *Web Services Choreography Description Language*, Nov. 2005. Version 1.0.

[23] Q. Xiaoqiang and W. Jun. A Decentralized Services Choreography Approach for Business Collaboration. In *IEEE International Conference on Services Computing, Proceedings* , pages 190–197. IEEE, Sept. 2006.

The 9th IEEE International Conference on E-Commerce
Technology and The 4th IEEE International Conference
on Enterprise Computing, E-Commerce and E-Services(CEC-EEE 2007)
0-7695-2913-5/07 $25.00 © 2007 IEEE