From an ebXML BPSS choreography to a BPEL-based implementation

Ja-Hee Kim and Christian Huemer
University of Vienna and Research Studios Austria

ebXML and Web Services are two well accepted technologies to implement business collaborations. In this paper we concentrate on the choreography of business collaborations. The corresponding standards of ebXML and Web Services are BPSS and BPEL, respectively. We present when and why we need to transform between these two standards. For the transformation we examine the difference between them. Furthermore, we suggest guidelines for a transformation from BPSS to BPEL.

Categories and Subject Descriptors: H.5.3 [**Group and Organization Interfaces**]: Web-based interaction

Additional Key Words and Phrases: ebXML, Web Services, business process, BPSS, BPEL

1. INTRODUCTION

In November 1999 the organizations UN/CEFACT and OASIS joined forces to develop an XML-based solution for companies conducting business over the Internet. This initiative became known as ebXML. It resulted in a set of modular specifications. These specifications define ways to (1) describe a business process, (2) assemble business documents from a set of core components with well defined business semantics, (3) describe a company's profile and agreements between companies, (4) register business processes and profiles, and (5) exchange business messages.

Shortly after the start of ebXML, the work on another set of standards, which became later known as Web Services, was initiated by the software industry. The core standards in the Web Services environment are WSDL, SOAP and UDDI. Additionally, new Web Services standards popped up whenever new requirements in the implementation of the Web Services stack were detected. Missing a central control - although most Web Services standards are managed by W3C or OASIS - there sometimes exist different Web Services standards for the same purpose.

It follows that both ebXML and Web Services are technologies for implementing business partnerships. It is quite common to consider these two families of standards as being competitors. However, this is not necessarily true. It seems that Web Services technology has been rapidly adopted by the software industry.

This work was partially supported by the Post-doctoral Fellowship Program of Korea Science & Engineering Foundation (KOSEF).

ebXML gained a lot of acceptance in different B2B communities specifying standard scenarios of business collaborations. Therefore, we believe that it is best to combine the strengths of both standard technologies.

In this paper we demonstrate that both standards can co-exist. We concentrate on standards to describe the sequencing of activities in an inter-organizational business process, which is further on called business collaboration. In the literature (c.f. [Peltz 2003]) both terms orchestration and choreography are related to the definition of an execution sequence. Orchestration refers to a process that is executed within the boundaries of an organisation and that interacts with internal and external services. The term choreography has a collaborative nature and tracks the business document exchanges among multiple parties and sources. The ebXML standard to describe the choreography of business collaborations is the Business Process Specification Schema (BPSS). In the Web Services environment there had been a lot activities related to business processes: WSFL, XLANG, WSCL, WSCI, and BPML. However, the winner seems to be the Business Process Execution Language for Web Services (BPEL). Although BPEL's primary goal is to describe the orchestration of an executable business process, abstract BPEL is also used to define the choreography for business protocols.

In Section 2 we motivate the co-existence of BPSS and BPEL by describing the life-cycle of a business process definition. In this scenario it becomes evident that a transformation from BPSS to BPEL is needed. In Section 3 we describe our approach towards an automatic transformation. The summary in Section 4 concludes the paper.

2. LIFE-CYCLE OF A BUSINESS COLLABORATION

In order to define a typical life-cycle of a business process we consider the scenario defined in the ebXML architecture ([UN/CEFACT OASIS 2001], page 8). In this scenario a small and medium enterprise *Company B* wants to collaborate with a large enterprise *Company A*. Instead of repeating the steps identified, we put the whole scenario in a business process-centric perspective (see figure 1). A business process definition for a business collaboration is usually created by a standard organization or industry consortia representing users of the respective industry (step 1). A methodology like UN/CEFACT's Modeling Methodology (UMM) [UN/CEFACT TMG 2003] might support the definition. In order to make this definition available to the public, it is uploaded to a business registry (step 2). This requires that the business process definition is described in a machine-processable format. Such a format that describes the business collaboration from a global and neutral perspective is BPSS.

A large company - in our example the seller - downloads the BPSS definition of a business collaboration and reviews it to check whether it is appropriate or not (step 3). If so, the seller implements the local process of the supported role, i.e. the seller role (step 4). A candidate technology for the implementation is Web Services. The workflow on the sellers side might be implemented by an engine that is based on BPEL. Hence, it is important to automatically derive a BPEL process from a BPSS process definition. Once the seller finishes the implementation, it registers its profile stating that it supports the seller role of the business collaboration in

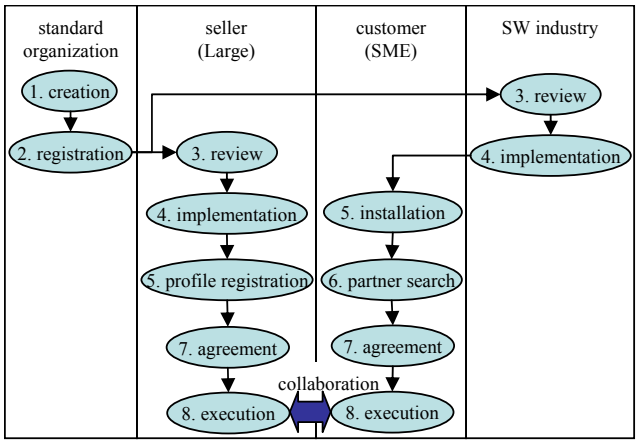


Fig. 1. Life-cycle of a business process.

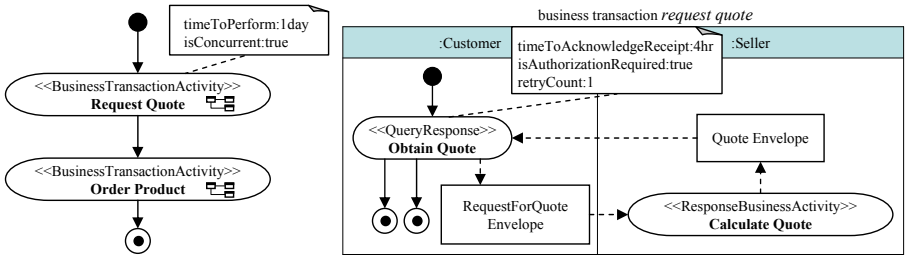
question (step 5). A candidate technology is ebXMLs collaboration protocol profile (CPP) referencing the respective role in the BPSS business process definition. In our example the customer is a small and medium enterprise that cannot afford to implement the collaboration by itself. It relies on the software industry. The implementers of commercial off-the-shelf software (COTS) are expected to integrate well-accepted business collaborations into their products. Therefore step 3 and 4 are not performed by the customer, but by his software provider. The customer buys the software package that supports its role in the business collaboration. By installing the software the customers profile is registered as well (step 5).

If the customer wants to perform the business collaboration, it searches for possible partners in the registry. This means it looks for CPPs that refer to the same BPSS business process definition, but to the complimentary role (step 6). Next an appropriate seller will be contacted to reach an agreement towards executing the business collaboration in question (step 7). The ebXML collaboration protocol agreement (CPA) supports this task. Finally, the customer and the seller are able to execute the business collaboration (step 8). According to the implementation in step 4 the run-time environment is based on Web Services and BPEL workflow engines.

Since BPEL always describes a scenario from the view of a specific business partner type, it is not suited to support the whole scenario. BPSS is able to describe the overall process from a neutral perspective. However, BPSS has not so much support by software providers. This requires a transformation from BPSS to BPEL in the transition from step 3 to step 4.

3. TRANSFORMATION

In this section we concentrate on the transformation from BPSS to BPEL. BPSS describes a business collaboration from a global and neutral view. A BPSS instance includes all the choreography information for the involved business service interfaces. BPSS does not care about the implementation of these business service interfaces. In our paper we assume that these business service interfaces are imple-



(a) binary collaboration. (b) business transaction.

Fig. 2. An example of UMM.

mented by Web Services. Each information exchange in the business collaboration is realized by a basic Web Services. In the Web Services technology, BPEL is the language of choice to build a complex choreography - like that of a business collaboration - from basic Web Services. Thus, the business collaboration originally defined in a BPSS instance must be transformed to BPEL.

In the first subsection we show how the BPSS hierarchy of business collaboration - business transaction - business action is mapped to BPEL. This is demonstrated by a very simple choreography. Real-world business collaborations involve a much more complex choreography. Thus, the second subsection suggests that a transformation of such a complex choreography should be based on well-known workflow patterns. Finally, the third subsection deals with the transformation of additional information assigned to BPSS activities.

3.1 Business Collaborations Refined

BPSS version 1.1 is limited to binary collaborations. This means that no more than two business partners collaborate. The activities within a binary collaboration are business collaboration activities and/or business transaction activities. Business collaboration activities are refined by a nested binary collaboration. A business transaction activity is refined by a business transaction.

In figure 2 we show a very simple example of a binary collaboration between a customer and a seller. This example uses the UMM notation [UN/CEFACT TMG 2003]. The equivalent BPSS code is given in code 1. This simple order management collaboration includes two business transaction activities *request quote* (line 104-106) and *order product* (line 107-108). The first activity is the *request quote* activity, since it is referenced by name in the *toBusinessState* attribute of the start element (line 103). The *order product* activity follows in sequence, which is indicated by the references in the attributes of the transition element in line 111. The elements *success* (line 109) and *failure* (line 110) lead to the end states of the business collaboration. Since both elements reference the *order product* activity, the collaboration ends after this activity. The collaboration ends successfully if the condition guard *ProtocolSuccess* is met. This is the case if the business transaction *OrderProductBT* refining the last business transaction activity does not lead to a protocol failure. Otherwise the condition guard for the failure element is met and the binary collaboration fails.

CODE 1. *BPSS code for the binary collaboration*

```
(101) <BinaryCollaboration name="OrderManagement">
(102)  <!- definition of Roles-->
(103)   <Start toBusinessState="RequestQuote"/>
(104)   <BusinessTransactionActivity name="RequestQuote"
(105)    businessTransaction="RequestQuoteBT"
(106)    timeToPerform="BT24H" isConcurrent="true"/>
(107)   <BusinessTransactionActivity name="OrderProduct"
(108)    businessTransaction="OrderProductBT"/>
(109)   <Success fromBusinessState="OrderProduct" conditionGuard="ProtocolSuccess"/>
(110)   <Failure fromBusinessState="OrderProduct" conditionGuard="AnyProtocolFailure"/>
(111)   <Transition fromBusinessState="RequestQuote" toBusinessState="OrderProduct"/>
(112) </BinaryCollaboration>
```

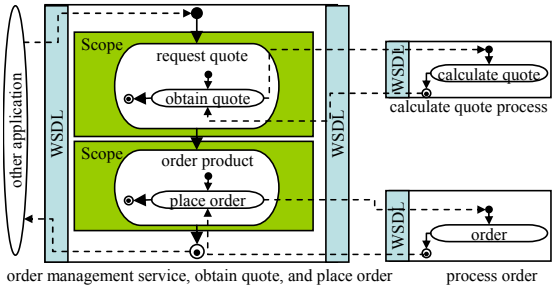
Our goal is to generate a semantically equivalent business collaboration in BPEL. The root element of BPEL is always a *process*. Therefore, the binary collaboration becomes a *process*. The first option is the following: Whatever is nested within the binary collaboration does not become a process of its own, but is specified within the flow of the binary collaboration. As a consequence the flow of the binary collaboration includes all activities performed by one partner type. Of course the resulting flow requires communication with services provided by the other partner type. This first approach depicted in 3a is suggested by [Hofreiter and Huemer 2004]. In our paper we suggest to solve out the business transactions. This means each business transaction becomes a BPEL process as well. This approach is shown in 3b. It results in a better modular architecture that enables the re-use of business transactions. Furthermore, the transformation process is clearer and the resulting BPEL code is easier to understand.

Code 2 defines the *process* for the binary collaboration *order management service*. The resulting process definition consists of a sequence of four activities. The first activity is by default the activation of the binary collaboration. The last activity returns by default the result of the binary collaboration. In between go the invocation of *obtain quote* (line 204), which starts the *request quote* process, and the invocation of *place order* (line 205), which starts the *order product* process.

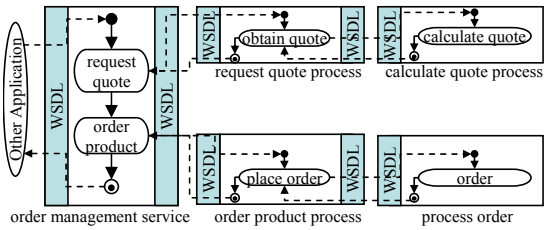
CODE 2. *XLANG-style BPEL code for the binary collaboration*

```
(201) <process name="orderManagementService">
(202)    <sequence>
(203)       <receive operation="orderManagement"/>
(204)       <invoke operation="obtainQuote"/>
(205)       <invoke operation="placeOrder"/>
(206)       <reply operation="orderManagement"/>
(207)    </sequence>
(208) </process>
```

BPSS defines a business transaction as a special type of a binary collaboration. It is an atomic unit that leads to a synchronized state in both information systems. It is always composed of two activities, each of them performed by one business partner. For example the *request quote* business transaction in figure 2b includes two activities, *obtain quote* and *calculate quote*. The customer starts by performing



(a) BT nested in a scope.



(b) BT as a another process.

Fig. 3. Two implementations of BPEL

the *obtain quote* activity, which creates a *request for quote envelope* and sends it to the seller. The receipt of this envelope initiates the seller's *calculate quote* activity. After calculating, the seller returns the *quote envelope* to the customer's *obtain quote* activity.

A business transaction always follows this simple pattern. Only the information flow for the response is optional. Thus, BPSS defines the following element structure. A business transaction element contains a *requesting business activity* element and a *responding business activity*. A document envelope created by one of the two activities is specified as child element of the corresponding activity. Code 3 represents the business transaction *request quote* depicted in figure 2b.

CODE 3. *BPSS code for the business transaction*

```
(301) <BusinessTransaction nameID="RequestQuoteBT">
(302)  <RequestingBusinessActivity nameID="ObtainQuote" isAuthorizationRequired="true"
(303)   retryCount="1" timeToAcknowledgeReceipt="BT4H">
(304)   <DocumentEnvelope nameID="RequestForQuote" businessDocument="RequestForQuote"/>
(305)  </RequestingBusinessActivity>
(306)  <RespondingBusinessActivity nameID="CalculateQuote">
(307)   <DocumentEnvelope nameID="QuoteEnvelope" businessDocument="QuoteEnvelope"/>
(308)  </RespondingBusinessActivity>
(309) </BusinessTransaction>
```

BPEL splits a business transaction into two processes - one for each participating partner. In our example, the business transaction is divided into the customer's process and the seller's process. Code 4 is the BPEL code for the customer.

This process is started by receiving a call of the *obtain quote* operation (line 413). This operation is called by the *order management service* (c.f. line 204) Since

request quote is an atomic process, it requires a failure handling mechanism in order to recover to the initial state if needed (line 402–406). This process sends a message *request for quote envelope* to the *seller* and waits for the response message *quote envelope* (line 420-421).

CODE 4. *BPEL code for the business transaction*

```
(401) <process name="requestQuoteProcess">
(402)  <faultHandlers>
(403)   <catch>
(404)    <!- operation for recover-->
(405)   </catch>
(406)  </faultHandlers>
(407)  <eventHandlers>
(408)   <onAlarm for="PT48H">
(409)    <!- conflict to time to perform -->
(410)  </onAlarm>
(411)  </eventHandlers>
(412)  <sequence>
(413)   <receive operation="obtainQuote" createInstance="yes"/>
(414)  <scope>
(415)   <eventHandlers>
(416)    <onAlarm for="PT4H">
(417)     <!- repeat from 420 to 421 -->
(418)    </onAlarm>
(419)   </eventHandlers>
(420)   <invoke operation="calculateQuote" inputVariable="requestForQuoteEnvelope"
(421)    outputVariable="quoteEnvelope"/>
(422)   </scope>
(423)   <reply operation="quoteRequest"/>
(424)  </sequence>
(425) </process>
```

The BPEL code for the collaborating partner - the seller - is a little bit simpler (See code 5). This process consists of recovery operations, an alarm handling, receive, reply activities. The receive and reply activities are complementary to the invoke activity of the customer process (line 420-421). In general, if the business transaction does not have any response message, the reply activity is removed.

CODE 5. *BPEL code for the responding part*

```
(501) <process name="calculateQuoteProcess">
(502)   <faultHandlers>
(503)     <catch>
(504)       <!- operation for recover-->
(505)     </catch>
(506)   </faultHandlers>
(507)    <sequence>
(508)       <receive operation="calculateQuote"
(509)          variable="requestForQuoteEnvelope" createInstance="yes"/>
(510)       <reply operation="calculateQuote" variable="quoteEnvelope"/>
(511)    </sequence>
(512) </process>
```

3.2 Complex Collaborations

The business transaction of our example in the previous subsection might be realistic. However, the binary collaboration - consisting of a sequence of two business transactions - is over-simplified. Real-world collaborations are built by much more complex workflow patterns than a sequence. It would go beyond the scope and page limit of this paper to elaborate all the possible workflow patterns in detail. In general, we suggest to base the transformation on the workflow patterns proposed by Aalst et al [Van der Aalst et al. 2003]. These patterns were originally developed to examine the expressive power of a workflow server. However, they have been used as well to evaluate languages/standards describing workflows. Amongst others, there exist evaluations of UML [Dumas and ter Hofstede 2001], BPEL [Wohed et al. 2003], and UMM/BPSS [Kim and Huemer 2004]. We summarize their results in table I, which shows the workflow patterns supported by each business process language. A '+' and a '-' in the cell mean direct support and no support, respectively. If the business process language partially supports the pattern, it is rated as '+/-'.

The basic idea of our approach is the following: If the flow of business transaction activities in the BPSS binary collaboration follows a certain pattern, the equivalent BPEL pattern is created. This task becomes easier, if not only the BPSS collaboration - there is no other choice - but also the BPEL collaboration follows a graph structure. In accordance to its predecessors, a BPEL process might be based on the pi-calculus (XLANG) or petri nets (WSFL). Code 2 is an example of XLANG style BPEL. It looks similar to structured program languages such as Pascal and C. The advantages of XLANG style are simple and easy to program structures. Since we prefer a graph-like notation, the WSFL-style is our choice. Consequently, we prefer the WSFL-style as given in code 6, which is equivalent to code 2.

	UMM	BPSS	BPEL
Sequence	+	+	+
Parallel Split	+	+	+
Synchronization	+	+	+
Exclusive Choice	+	+	+
Simple Merge	+	+	+
Multi Choice	+	+	+
Synchronizing Merge	+	-	+
Multi Merge	-	-	-
Discriminator	-	+	-
Arbitrary Cycles	+/-	+	-
Implicit Termination	+/-	+/-	+
MI without Synchronization	+	+	+
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	+	+	-
Cancel Activity	-	-	+
Cancel Case	+	+	+

Table I. Workflow patterns of UMM, BPSS, and BPEL.

CODE 6. *WSFL style BPEL code for the binary collaboration*

```
(601) <flow>
(602)      <links>
(603)         <link name="Start"/> <link name="RQ2OP"/> <link name="Final"/>
(604)     </links>
(605)     <receive operation="orderManagement">
(606)         <source linkName="Start"/>
(607)     </receive>
(608)     <invoke operation="quoteRequest" outputVariable="PO">
(609)         <target linkName="Start"/> <source linkName="RQ2OP"/>
(610)     </invoke>
(611)     <invoke operation="orderProduct" inputVariable="PO" outputVariable="">
(612)         <target linkName="RQ2OP"/> <source linkName="Final"/>
(613)     </invoke>
(614)     <reply operation="orderManagement">
(615)         <target linkName="Final"/>
(616)     </reply>
(617) </flow>
```

Each element of BPSS can be mapped to WSFL style BPEL nearly one by one. Start (line 103) and business transaction activity (line 104-106 and line 107-108) elements in code 1 can be mapped to receive (line 605-607) and invoke (line 608-610 and line 611-613) elements in code 6, respectively. BPSS has at least two final state, success (line 109) and failure (line 110). In contrary BPEL uses exactly one reply element (line 614-616) which must signal the state in the message of the corresponding Web Services. A pair of elements is connected by a transition in BPSS and a link in BPEL. A BPSS transition elements specifies the source and the target activity. In contrary, a BPEL activity includes the information for which links it is the source and/or target. The number of BPEL links corresponds to the number of BPSS transitions plus two, a link from the start activity and a link to the reply element.

A transformation from BPSS to BPEL should include all the patterns that are supported by BPSS. In other words we have to consider all the patterns marked '+' in the BPSS column of table I. Among these patterns a sequence, a parallel split, a synchronization, an exclusive choice, and a simple merge pattern are most elementary [Van der Aalst et al. 2003; WFMC 1999]. Both BPSS and BPEL support these elementary workflow patterns. Moreover, these languages also support a multi choice, a deferred choice, and a cancel case pattern.

Some workflow patterns such as a discriminator, an arbitrary cycle, and a milestone pattern are supported by BPSS but not by BPEL. However, we can transform them using work-around solution suggested by Aalst *et al.* [Van der Aalst et al. 2003]. For example, an arbitrary cycle implemented in BPSS can be transformed to a structured cycle, which can be modeled in BPEL. Therefore, BPEL can model the same choreography to the arbitrary cycle pattern after the transformation to the structured cycle.

3.3 Properties of Activities

UMM defines tagged values for activities that are specific for the purpose of modeling B2B business collaborations. Since BPSS is derived from the UMM meta-model, some of these tagged values are reflected as attributes to BPSS elements. A BPSS business transaction activity includes information about the maximum time to perform and a flag to signify whether the activity might be concurrent or not. The business transaction activity *request quote* in figure 2a shows two tagged values, *time to perform: 1 day* and *is concurrent: true*. This is equivalent to the attributes in line 106.

BPEL does not include this kind of attributes for activities. However, we are able to map the semantics of both attributes. If a business transaction is concurrent, this means that a new instance of an activity must be created for each concurrent execution. Hence, the *createInstance* attribute of the BPEL activity is set to *yes* (line 413). It is important to distinguish different instantiations of the same activity. Usually, messages carry business tokens as unique identifiers in the header or in the business documents. The values for the same token must differ for different instances. If ebXML messaging is used, header elements for such tokens are provided. BPEL uses so-called correlation sets to declaratively express such tokens in the process description. In this case the XPath value of a BPEL correlation set should reference the corresponding ebXML messaging header elements.

The implementation of *time to perform* attribute in BPEL is more complex. The tagged value *time to perform: 1 day* of business transaction activity *quote request* means that the corresponding business transaction should raise an alarm if the activity takes more than one day. Therefore, we attach the alarm event handler to *quote request process* (line 407-411).

The requesting and responding activities in a business transaction also carry tagged values. In figure 2b, the action *obtain quote* has three tagged values: *time to acknowledge receipt:4hr, is authorization required: true, retry count: 1*. These tagged values are implemented in lines 302-303. In order to transform these tagged values, we should understand their semantics. Tagged value *time to acknowledge receipt* means that the seller signals within 4 hours that the received document passed a syntax and a sequence check . Therefore, it is also implemented as an alarm event (line 415-418). This alarm relates only to the invoke activity *calculate quote*(line 420-421). The tagged value *retry count* is closely related. It defines the number of retrials in case of an alarm. Hence, the number of alarms must correspond to the number of retrials.

BPSS includes some features that do not directly match to any BPEL concept. For example, the activities in a business transaction have special security requirements. These security requirements are assigned as attributes to the *requesting/responding business activity* elements. Consider the *is authorization required* attribute of *obtain quote* in figure 2. This means that the output of this activity, which is the *request for quote envelope* must be signed. Since BPEL deals only with choreography/orchestration of Web Services, this security feature is not supported. Thus, this concept must be handled by other standards of the Web Services stack. The appropriate standard in this case is WS-Security [OASIS 2004]. WS-Security extends SOAP by specific header elements ensuring secure SOAP mes-

sage exchanges. These header elements are used to sign a document. Alternatively, the ebXML messaging extending the SOAP header in a similar way might be used.

Accordingly, there are three ways to transform the attributes of a business transaction activity and actions in business transaction. They depend on the nature of the attribute. The first group uses a mapping to a corresponding BPEL attribute. The second group is reflected in the process structure. Finally, the third group does not have a corresponding BPEL concept. They must be handled by other standards of the Web Services stack.

4. CONCLUSION

Recently, ebXML and Web Services became popular technologies to implement business collaborations. The standards used to describe the choreography of business collaborations are BPSS for ebXML and BPEL for Web Services, respectively. In this paper, we demonstrate that these two standards do not exclude each other in a business environment. We show a scenario that requires both BPSS and BPEL to co-exist. BPSS is used to describe a global choreography. The implementation of a workflow in the participants' systems is based on BPEL. As a result, a transformation from BPSS to BPEL is required. In this paper we suggest guidelines for this transformation. We propose a general guideline in order to reflect the hierarchical structure of BPSS business collaborations in BPEL. Furthermore, we suggest to base the transformation of complex workflows on well-accepted workflow patterns. In our future work we will develop formal rules for the transformation of each pattern.

REFERENCES

Dumas, M. and ter Hofstede, A. H. 2001. UML activity diagrams as a workflow specification language. *Lecture notes in computer science 2185*, 76 – 90.

Hofreiter, B. and Huemer, C. 2004. Transforming UMM business collaboration models to BPEL. In *Workshop on Modeling Inter-Organizational Systems (MIOS 04)*. Springer LNCS 3292, Cyprus, 507 – 519.

Kim, J.-H. and Huemer, C. 2004. Analysis, transformation and improvements of ebXML choreographies based on workflow patterns. In *International Conference on Cooperative Information Systems (CoopIS 04)*. Springer LNCS 3290, Cyprus, 66 – 84.

Peltz, C. 2003. Web services orchestration and choreography. *IEEE Computer 36,* 10 (October), 46 – 52.

OASIS. 2004. Web services security: SOAP message security 1.0. Tech. Rep. OASIS Standard 200401, OASIS, http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0. March.

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

UN/CEFACT TMG. 2003. UN/CEFACT modeling methodology, revision 12. http://www.untmg.org.

WFMC. 1999. Workflow management coalition terminology & glossary. Tech. Rep. WFMC-TC-1011, WfMC. February.

Van der Aalst, A., Hofsteded, A. T., Kiepuszewski, B., and Barros, A. 2003. Workflow patterns. *Distributed and Parallel Databases 14*, 5 – 51.

Wohed, P., van der Aalst, W. M., Dumas, M., and ter Hofstede, A. H. 2003. Analysis of web services composition languages: The case of BPEL4WS. In *International Conference on Conceptual Modeling (ER 2003)*. Springer LNCS 2813, Berlin, 200–215.