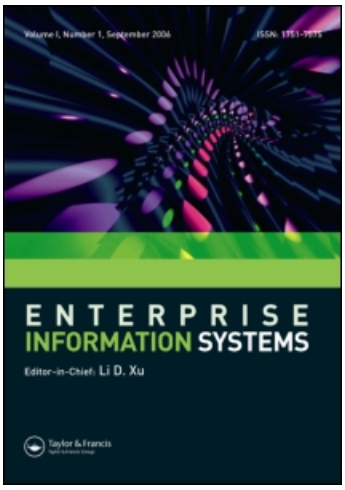
This article was downloaded by: *[Rosenberg, Florian]*
On: *24 November 2008*
Access details: *Access Details: [subscription number 904758859]*
Publisher *Taylor & Francis*
Informa Ltd Registered in England and Wales Registered Number: 1072954 Registered office: Mortimer House, 37-41 Mortimer Street, London W1T 3JH, UK



Enterprise Information Systems
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t748254467

Top-down business process development and execution using quality of service aspects

Florian Rosenberg [a]; Anton Michlmayr [a]; Schahram Dustdar [a]
[a] Distributed Systems Group, Technical University of Vienna, Vienna, Austria

Online Publication Date: 01 November 2008

To cite this Article Rosenberg, Florian, Michlmayr, Anton and Dustdar, Schahram(2008)'Top-down business process development and execution using quality of service aspects',Enterprise Information Systems,2:4,459 — 475
To link to this Article: DOI: 10.1080/17517570802395626
URL: http://dx.doi.org/10.1080/17517570802395626

PLEASE SCROLL DOWN FOR ARTICLE

Full terms and conditions of use: http://www.informaworld.com/terms-and-conditions-of-access.pdf

This article may be used for research, teaching and private study purposes. Any substantial or systematic reproduction, re-distribution, re-selling, loan or sub-licensing, systematic supply or distribution in any form to anyone is expressly forbidden.

The publisher does not give any warranty express or implied or make any representation that the contents will be complete or accurate or up to date. The accuracy of any instructions, formulae and drug doses should be independently verified with primary sources. The publisher shall not be liable for any loss, actions, claims, proceedings, demand or costs or damages whatsoever or howsoever caused arising directly or indirectly in connection with or arising out of the use of this material.

Enterprise Information Systems
Vol. 2, No. 4, November 2008, 459–475

Taylor & Francis
Taylor & Francis Group

Top-down business process development and execution using quality of service aspects

Florian Rosenberg*, Anton Michlmayr and Schahram Dustdar

Distributed Systems Group, Technical University of Vienna, Vienna, Austria

(*Received 27 January 2008; final version received 7 August 2008*)

Developing cross-organisational business processes is a challenging task. The partners have to agree on a common data format and meaning as well as on the quality of service (QoS) requirements each partner has to fulfil. The QoS requirements are typically described using service level agreements (SLAs) among the partners. In this paper, a top-down modelling approach for Web service based business processes is proposed to capture the functional and non-functional aspects using a choreography language (WS-CDL) which describes the message interactions among the participants. The choreography is annotated with SLAs for the different partners. For each partner in the process, an orchestration (in WS-BPEL) and the necessary Web service templates are automatically generated. The service level objectives (SLOs) from the partner SLAs are automatically translated into policies that can then be enforced by a BPEL engine during execution. The deployment of the WSDL files, the monitoring of QoS attributes and the execution of the BPEL process itself are then handled by the VRESCo SOA runtime.

Keywords: quality of service; top-down modelling; business process development; service level agreements; WS-CDL; WS-BPEL

Downloaded By: [Rosenberg, Florian] At: 10:28 24 November 2008

1. Introduction

Service-oriented architecture (SOA) represents an emerging paradigm to develop flexible and large-scale software systems using the Internet as the main infrastructure. Web services are one realisation of this paradigm by using well-established standards to describe and interact with other services.[1]

Many organisations are building their cross-organisational business processes based on Web services because of their platform-agnostic nature and the ease of integration. Currently available technologies such as composition engines using the Web Service Business Process Execution Language (WS-BPEL, or BPEL for short; see Alves *et al.* 2007) can be used to orchestrate business processes within an organisation.

An engineering method for Web service based business processes involving multiple partners requires agreement on the data exchanged that cannot be achieved using BPEL. For this purpose, the Web Service Choreography Description Language (WS-CDL; see Kavantzas *et al.* 2008) provides an XML-based language to describe the cross-organisational message exchanges from a global viewpoint. The different views (local *versus* global) are described by the terms *choreography* and *orchestration*. Choreography

*Corresponding author. Email: florian@infosys.tuwien.ac.at

ISSN 1751-7575 print/ISSN 1751-7583 online
© 2008 Taylor & Francis
DOI: 10.1080/17517570802395626
http://www.informaworld.com

can be defined as 'processes involving multiple services where the interactions between these services are seen from a global perspective' (Kavantzas *et al*. 2008). A choreography does not describe any internal actions that occur within a participating service, such as internal computation or data transformation, but rather focuses on the observable public exchange of messages. In contrast to that, Peltz (2003) defines *orchestration* as an 'executable business process that can interact with both internal and external Web services. The interactions occur at the message level. They include business logic and task execution order, and they can span applications and organisations to define a long-lived, transactional, multistep process model'.

These two concepts imply that such a choreographic description can be used to generate the orchestration behaviour (e.g. in the form of BPEL stubs) and the necessary WSDL templates automatically. Nowadays, service level guarantees and obligations among the service providers are becoming increasingly important as a means to capture runtime quality requirements and guarantees for a partner's service (such as response time, availability and security). Such quality of service (QoS) requirements are generally specified in a service level agreement (SLA) and need to be fulfilled during the service execution among the partners. Using existing Web service standards and proposals there is currently no integrated modelling method available to build such cross-organisational business processes considering SLA requirements as first-class citizens from the starting point of the development process. Additionally, the orchestration parts and the WSDL files for each partner in the choreography should be automatically generated.

In this paper, we propose a top-down modelling approach to build such QoS-aware, Web service based business processes using currently available technologies such as WS-CDL and BPEL. It leverages a design approach for efficient development of cross-organisational business processes similar to the idea of model-driven software development (Völter and Stahl 2006). The novelty of our approach is twofold: (1) a relevance mapping from WS-CDL to BPEL and (2) the consideration of SLA requirements from the beginning of the choreography development process. These SLA requirements are then automatically transformed and mapped to a WS-QoS policy and attached to the BPEL process of the affected partner allowing policy-aware middleware to check and enforce the SLA.

This paper is organised as follows. Section 2 describes the case study we implemented for evaluating the concepts of this work. Section 3 describes our main approach for realising QoS-aware business process development. The implementation of this approach is sketched in Section 4 followed by an evaluation in Section 5. Section 6 positions our approach among existing work and, finally, Section 7 concludes the paper.

2. Case study

In this case study we developed a *build-to-order* (BTO) scenario in the B2B area. The use case consists of a customer, a manufacturer, and suppliers for CPUs, main boards and hard disks. The manufacturer offers assembled IT hardware equipment to its customers. For this purpose the manufacturer has implemented a BTO business model. It holds a certain part of the individual hardware components in stock and orders missing components if necessary. In the implemented BTO scenario, the customer sends a quote request with details about the required hardware equipment to the manufacturer. The latter sends a quote response back to the customer. As long as customer and manufacturer do not agree on the quote, this process will be repeated. If a mutual agreement is achieved, the customer sends a purchase order to the manufacturer. Depending on its hardware stock the manufacturer has to order the required hardware components from its suppliers.

Downloaded By: [Rosenberg, Florian] At: 10:28 24 November 2008

If the manufacturer needs to obtain hardware components to fulfil the purchase order, an appropriate hardware order is sent to the respective supplier. In turn the supplier sends a hardware order response to the manufacturer. Finally, the manufacturer sends a purchase order response back to the customer.

The interactions of the participants in our BTO scenario[2] are illustrated in the collaboration sequence diagram shown in Figure 1. The BTO scenario consists of six different Web service invocations which correspond to the following SOAP operations: `requestForQuote`, `updateQuote`, `sendPurchaseOrder`, `orderCPU`, `orderMB`, `orderHD`. Each SOAP operation depicts a synchronous message request-reply scenario which will be illustrated as an exemplar for the `requestForQuote` operation. The customer invokes the operation `requestForQuote` on the service interface of the manufacturer sending the `QuoteRequest` message. The manufacturer receives the message request and replies to the service invocation by returning the `QuoteResponse` message. Contrary to this, an asynchronous message scenario would require additional callback operations on the service requestor side. In this case the manufacturer invokes an operation `requestForQuoteCallback` on the service interface of the customer to send back the `QuoteResponse`.

The definition of SLA and QoS plays a crucial rule in cross-organisational business processes. Each participant offers services to other partners over the Internet which the latter need to run their businesses. Therefore, a certain degree of reliability concerning response time, throughput, uptime, etc. is desired and has to be specified and explicitly expressed from the beginning of the modelling phase. In our scenario we distinguish four different relationships between the choreography participants. The customer interacts with

Downloaded By: [Rosenberg, Florian] At: 10:28 24 November 2008

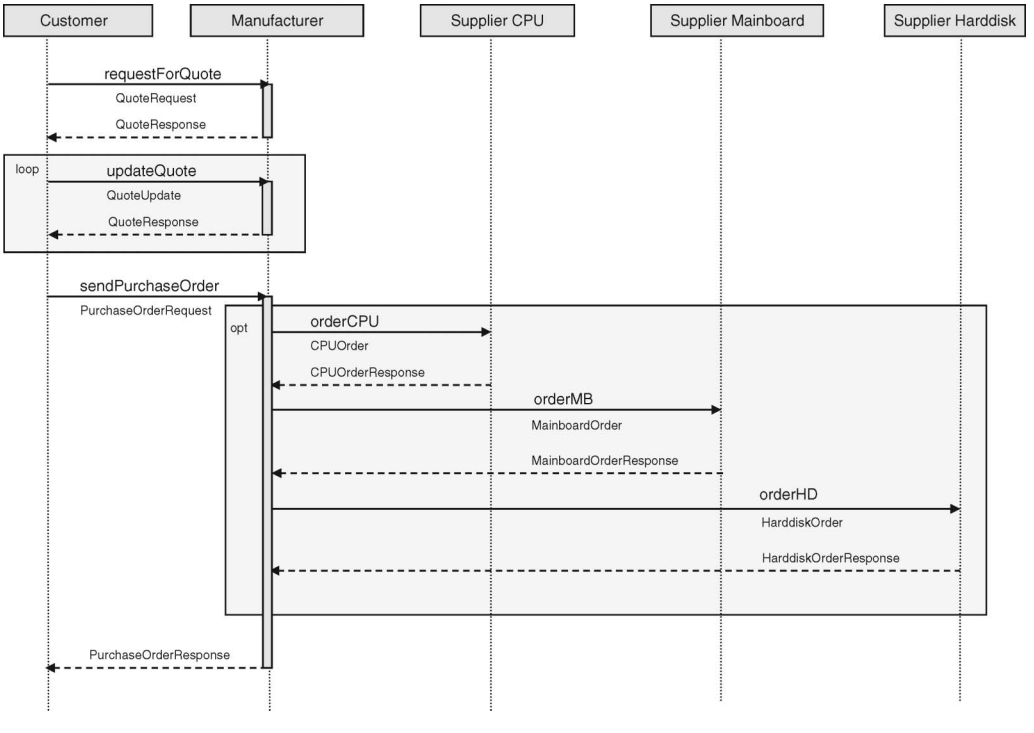


Figure 1. BTO case study.

the manufacturer; the manufacturer interacts with different suppliers. For each relationship an SLA is defined between the partners to regulate this degree the partners need for their business.

3. Top-down modelling approach

In this section, we describe our top-down modelling approach and the QoS integration. Due to space restrictions, we cannot describe WS-BPEL (Alves *et al.* 2007), WS-CDL (Kavantzas *et al.* 2008) and WS-Policy (Vedamuthu *et al.* 2008a) and therefore assume basic familiarity with the concepts and their syntax. For a short overview of these technologies, see Rosenberg *et al.* (2007).

3.1. Overview

As shown in Figure 2, the language constructs of WS-CDL can be mapped to BPEL allowing a choreographic description to be transformed into separate BPEL processes, one for each partner in the choreography, including corresponding WSDL descriptions.

Our approach consists of two main steps which are highlighted in this figure. In the first step, a number of models are defined at the highest level of abstraction (the choreography layer). On the one hand, this contains the WS-CDL document specifying the choreography in detail (*Step 1a*). The choreography is used to describe the partners in the process and the message exchanges. On the other hand, the SLAs between the participants are defined (*Step 1b*) which describe obligations and guarantees among the participants. The choreography is annotated with the SLA references to allow a pairwise agreement between two partners on a specific SLA. Furthermore, SLAs may also include QoS attributes as described in more detail below.

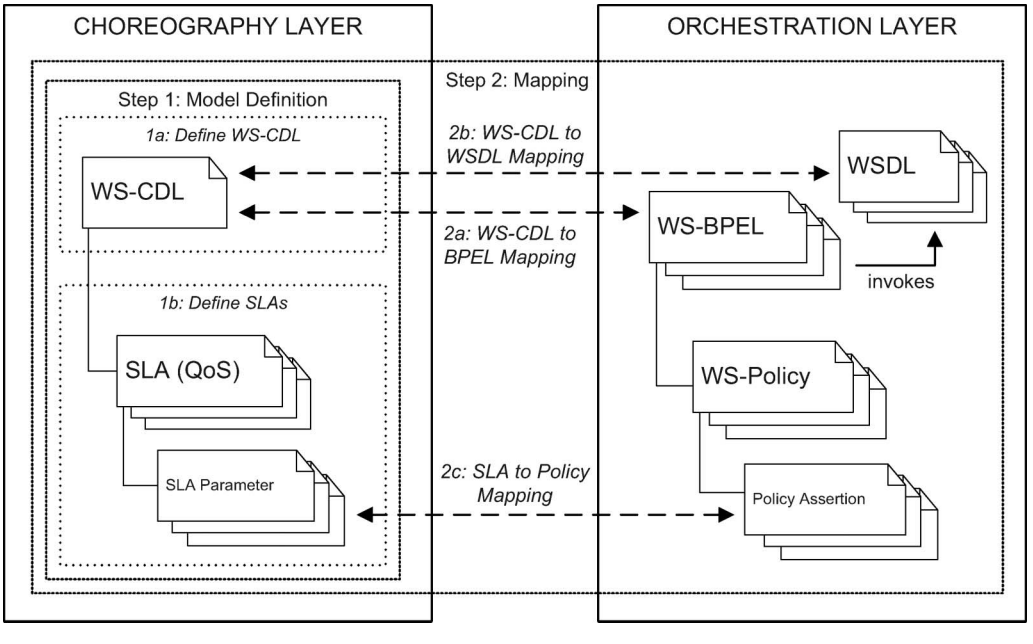


Figure 2. Modelling approach.

Downloaded By: [Rosenberg, Florian] At: 10:28 24 November 2008

In the second step, the documents defined in the first step are transformed from the choreography layer into executable code in the orchestration layer. First, the WS-CDL choreography is mapped into a number of BPEL processes depending on the number of participants (*Step 2a*). Secondly, the WS-CDL document is used to generate WSDL descriptions of those Web services, which each participant has to implement and provide to its business partners (*Step 2b*). The details of mapping WS-CDL to BPEL and WSDL are described in Section 3.2.

The importance of QoS in cross-organisational business processes makes it necessary to consider these aspects from the beginning of the development process. Therefore, the SLAs are transformed to WS-QoSPolicy statements, which is our extension to WS-Policy, that are directly attached to the corresponding partner links in BPEL (*Step 2c*). The details of this transformation step, the integration of QoS and WS-QoSPolicy are given in Section 3.3.

Downloaded By: [Rosenberg, Florian] At: 10:28 24 November 2008

3.2. *Mapping WS-CDL to BPEL and WSDL*

The main goal of transforming WS-CDL to BPEL is to allow the participants a rapid modelling and development process and generate relevant BPEL and WSDL documents which can then be used as a basis to implement the private (non-visible) business logic. The projection of such a global description to endpoint processes whose interactions precisely realise the global description is called *endpoint projection* (Carbone *et al*. 2007).

Mendling and Hafner (2007) define basic mapping rules from WS-CDL to BPEL. They use a recursive XSLT-based approach to generate the BPEL processes by iterating through each role type to check the relevance of the node. The authors consider a node as relevant if it contains activities with the attribute `toRoleTypeRef` and `fromRoleTypeRef`. However, this approach does not correspond with the endpoint projection definition given above, because more structured BPEL elements are generated than necessary. This is due to the fact that all parent nodes are considered during the mapping process even if they are not directly relevant (it can be considered as a simple 1:1 mapping). Listing 1 depicts an

```
1  <package>
2    <choreography>
3       <sequence>
4        <!-- ... -->
5        <sequence>
6          <!-- ... -->
7          <sequence>
8            <interaction operation="sendPO" ...>
9              <participate fromRoleTypeRef="Customer"
10                          toRoleTypeRef="Manufacturer" .../>
11             <exchange action="request" ...>
12               <!-- ... -->
13             </exchange>
14           </interaction>
15           <interaction operation="sendPO" ...>
16             <participate fromRoleTypeRef="Customer"
17                          toRoleTypeRef="Manufacturer" .../>
18             <exchange action="respond" ...>
19               <!-- ... -->
20             </exchange>
21           </interaction>
22         </sequence>
23       </sequence>
24     </sequence>
25   </choreography>
26 </package>
```

Listing 1. Choreography example.

example of this problem by using three nested `sequence` elements. In this example, their approach generates three nested BPEL `sequence` activities, although only the inner sequence (Lines 7–22) is relevant for the `Customer` and `Manufacturer` as they are referenced in the `fromRoleTypeRef` and `toRoleTypeRef`, respectively.

Therefore, we have adapted the rules from Mendling and Hafner (2007) and propose an extended endpoint projection mechanism based on a so-called *relevance mapping*. The basic idea is to map only those WS-CDL elements which are relevant in the BPEL process. To map the different ordering structures we need to distinguish between *child* and *descendant* relevance. The former describes that a relevant basic activity occurs as an immediate child of the respective ordering structure in the tree, whereas the latter describes that a relevant basic activity is nested at an arbitrary level. The relevance of a WS-CDL basic activity is determined by the occurrence of `interaction`, `assign` or `silentAction` where the `roleType` attribute is matching the `roleType` of the corresponding participant in the choreography.

If a node represents a relevant activity as described above, it is mapped to a BPEL activity according to Table 1, otherwise no mapping is generated. The basic algorithm for the relevance mapping is depicted in Listing 2. This algorithm takes a valid WS-CDL document as input and produces a number of BPEL documents, depending on the number of role types. For each role type at least one BPEL document is generated. We assume that a WS-CDL file is correct when it fulfils a number of properties, e.g. deadlock-free and livelock-free (Lohmann *et al*. 2007).

From Lines 2 to 7, we generate a BPEL process for each `roleType` found in the choreography. The algorithm inspects the `choreography` tag of the WS-CDL document by iterating each activity. If the activity type is an ordering structure or workunit a relevance mapping is performed (Lines 11–22). If the currently inspected activity is *descendant relevant* we must consider all child nodes of this activity (Line 12). If an activity is *child relevant* (Line 16), we generate the corresponding BPEL mapping according to Table 1 (Line 17). Otherwise, we recursively visit all child nodes (Line 19).

For our BPEL mapping we implemented an additional optimisation concerning the ordering structures. If a `parallel` or `sequence` ordering structure contains only one basic child activity, this ordering structure is ignored in the BPEL mapping (Lines 13–14). For instance, considering the example from Listing 1, only one BPEL `sequence` activity will be generated.

In Table 1 we have depicted a detailed overview of the WS-CDL to BPEL mapping rules. These rules are based on the mappings proposed by Mendling and Hafner (2007) and adapted where necessary. The adaption mainly includes `interaction` and `choice`. For the `interaction` activity and `choice` ordering structure, we also have to consider the role types to determine the sending and receiving party. Additionally, we address the synchronous and asynchronous message exchange patterns properly in the `interaction` activity.

Besides generating the BPEL artefacts, we also need to generate the WSDL artefacts from the WS-CDL description. The latter defines the static structure which can be extracted without analysing the choreography flow in detail. We generate a new WSDL document for each `roleType` of the choreography if the service interface is invoked somewhere in the choreography flow. The main idea is to check if the `roleType` is referenced within a `channelType` and a `variable` for this `channelType` exists that is used in an `interaction` with another partner. If this is the case, the `roleType` is in use and a WSDL needs to be generated. The WS-CDL to WSDL mapping is summarised in Table 2.

Downloaded By: [Rosenberg, Florian] At: 10:28 24 November 2008

Table 1. WS-CDL to BPEL mapping.

WS-CDL	BPEL	Semantics
	Activities	
workunit (nested in choice)	case	Repeat and block attributes always false
workunit (block = true)	(receive)	Concept of blocking condition not defined in BPEL (Barros *et al*. 2005)
workunit (all other cases)	while	repeat = true and block = false
sequence	sequence	Sequential execution of activity units
parallel	flow	Parallel execution of activities
choice	switch	If inspected `roleType` is referenced in the guard condition of the inner workunit
	onMessage (nested in) pick	If inspected `roleType` is not referenced in the guard condition of the inner workunit but referenced in an `interaction` activity
interaction		
action = request	invoke	`fromRoleType` attribute corresponds to inspected role type
action = request	receive	`toRoleType` attribute corresponds to inspected role type. If `interaction` inside `workunit` which is defined inside a choice generate a BPEL `onMessage`
action = response	reply	`toRoleType` attribute corresponds to inspected role type
action = response	receive	`receive` only in the asynchronous case. For synchronous interaction append `outputVariable` to corresponding BPEL `invoke` which is defined in case 1
perform	*no mapping*	Perform separately defined choreography
assign	assign (for party in role type)	Variable assignment
silentAction	sequence with nested empty	To be refined in the BPEL process
noAction	empty (for party in role type)	Do nothing
finalise	compensationHandler	Finalising activities after completion

Downloaded By: [Rosenberg, Florian] At: 10:28 24 November 2008

For a detailed description of the WS-CDL to WSDL mapping algorithm, refer to Rosenberg *et al*. (2007).

3.3. SLA/QoS integration

The integration of QoS attributes in business process development raises the need for appropriate techniques to consider QoS at the choreography and orchestration layer. Considering QoS at the choreography layer can be achieved by using SLAs which focus on performance and dependability aspects of the underlying QoS model. In contrast, the integration of QoS at the orchestration layer can be attained by the use of Web service policies. This section describes how WS-CDL and BPEL can be extended to support QoS attributes.

```
1  void transform(WscdlDoc doc) {
2    List<RoleType> roles = doc.getRoleTypes();
3    for (RoleType role : roles) {
4      for (activity : n.getActivites()) {
5        if (isOrderingStructure(activity) || isWorkUnit(activity))
6          relevanceMapping(activity, role);
7      }
8    }
9  }
10
11 void relevanceMapping(Node n, RoleType role) {
12   if (isDescendanceRelevant(n)) {
13     if (n.getRelevantChildCount() > 1)
14       createBPELMapping(n);
15     for (child : n.getChildNodes()) {
16       if (isChildRelevant(child))
17         createBPELMapping(child);
18       else
19         relevanceMapping(child, role);
20     }
21   }
22 }
```

Listing 2. Relevance mapping.

Downloaded By: [Rosenberg, Florian] At: 10:28 24 November 2008

Table 2. WS-CDL to WSDL mapping.

WSDL		WS-CDL	
Element	*Attribute*	*Element*	*Attribute*
definitions	xmlns:tns targetNS name	package behaviour	xmlns:tns targetNS name
message	name	exchange	informationType

portType
  operation
  [input | | output]
	name
	name
	name
	message
		behaviour
		interaction
		exchange
			interface
			operation
			action
			informationType

binding
 	
  operation
  soap:operation
 	
  input soap:body
  output soap:body
	name
	type
	name
	soapAction
	
	namespace
	namespace
		behaviour
		
		interaction
		behaviour
		interaction
		behaviour
		behaviour
			name + 'Binding'
			'tns:' + interface+'Binding'
			operation
			interface namespace + operation
			
			interface namespace
			interface namespace

service port	name name binding	behaviour behaviour	interface + 'Service' interface + 'Port' 'tns:' + name + 'Binding'

3.3.1. *SLA integration*

As mentioned above, we use SLAs to integrate QoS at the choreography layer. For the definition of the SLAs we decided to use WSLA (Ludwig *et al.* 2008). For the actual integration, we extended WS-CDL with a construct which holds SLA references. We therefore leverage semantic annotations in WS-CDL constructs using the `description` element as shown in Listing 3 (Lines 3–7).

3.3.2. WS-QoS policy

In order to bring QoS aspects from the choreography to the orchestration layer, SLAs have to be mapped to the corresponding Web service policies to allow monitoring and enforcement of QoS attributes at the execution layer. However, the current WS-Policy specification focuses on security (WS-SecurityPolicy) and reliable messaging (WS-RMPolicy), whereas performance and dependability are not addressed. The WS-Policy framework provides a grammar for the definition of domain-specific policies. Hence, we use this feature to extend the WS-Policy framework by defining a WS-QoSPolicy. In our approach, the mapping from SLA to WS-QoSPolicy and its attachment to BPEL is done automatically, whereas other policy types (such as security and reliable messaging) are currently not supported by the mapping. Nevertheless, these can be manually attached to the BPEL documents as described in Charfi *et al*. (2007).

Before we go into the details of our mapping between SLAs and policies, we briefly sketch the underlying QoS model. Rosenberg *et al*. (2006) have defined a QoS model for Web services by identifying different QoS attributes. Since some attributes are either dependent on external factors or derived from empirical values, not all attributes are determinable in advance. In this work, we consider *Processing Time, Execution Time, Throughput* and *Availability* as possible QoS attributes in the WS-QoSPolicy. However, guarantees on the *Execution Time* will usually be defined in SLAs instead of *Processing Time* because the first also includes the network latency, which partly depends on the client's network connection.

The WS-QoSPolicy defines assertions for these QoS attributes. The normative outline of the assertions is shown in Listing 4. It defines `type`, `unit`, `predicate` and `value` of the assertion.

A concrete example for two such policy assertions is illustrated in Listing 5. Lines 3–4 define an assertion on the execution time, which has to be less than five seconds. Lines 5–6 state that the throughput has to be more than 100 requests per second.

3.3.3. SLA/QoS mapping

Our extension of the WSLA schema restricts the SLA parameters to the pre-defined QoS attributes introduced in the previous section. Therefore, the SLA can be directly mapped

```
1 <roleType name="ManRoleType">
2   <behaviour interface="b2o:manInterface" name="ManBehaviour"/>
3     <description type="semantics">
4       <qosp:slaReference name="SLA1" serviceconsumer="CustRoleType"
5         uri="http:// www.vitalab.tuwien.ac.at/slas/ManufacturerCustomerSLA.xml"
6       </qosp:slaReference>
7     </description>
8   </behaviour>
9 </roleType>
```

Listing 3. SLA integration in WS-CDL.

```
1 <qosp:[QoS]Assertion unit="xs:string"
2       predicate="tns:PredicateType"
3       value="xs:integer | xs:flow"/>
```

Listing 4. WS-QoSPolicy assertions.

Downloaded By: [Rosenberg, Florian] At: 10:28 24 November 2008

to the WS-QoSPolicy, which consists of the following two steps: first, each SLA is mapped to a policy and, secondly, each SLA parameter is mapped to a policy assertion.

As each SLA may consist of one or more SLOs, we identified three patterns:

(1) One SLO is defined for each SLA parameter.
(2) One SLO consists of multiple SLA parameters.
(3) SLA parameters are defined in multiple SLOs.

Each of these patterns can be successfully mapped to an equivalent policy. In the first case, one `All` operator is used to contain all policy assertions. For each SLO, exactly one policy assertion will be generated. For example, an SLO `SLOServiceExecutionTime` defines an SLA parameter which corresponds to the type `ExecutionTime`. This parameter will be mapped to the corresponding policy assertion according to the WS-QoSPolicy.

In the second case, SLA parameters are grouped by an SLO using the logical operators `And, Or, Not, Implies`. Table 3 shows how these constructs can be mapped to equivalent WS-QoSPolicy operators. For example, such groupings of SLA parameters can be used to define an SLO `SLOServicePerformance` by combining `Throughput` and `ExecutionTime` in various ways using the provided logical operators.

In the third case, for each SLO a time period has to be specified. Therefore, it is possible to define multiple SLOs for different time periods. For instance, during peak-hours the execution time of a service has to be less than a specific value.

3.3.4. *WS-QoS policy integration*

The definition of a QoS policy and QoS/SLA mapping rules are the fundamental concepts for considering QoS in Web service based business process development. Yet the question remains of how to integrate the generated QoS policies in the orchestration layer. Regarding the top-down modelling approach of Web services, two integration approaches can be differentiated: policies can either be attached to service descriptions (WSDL) or be integrated in BPEL processes.

```
1 <wsp:Policy>
2   <wsp:All>
3     <qosp:ExecutionTimeAssertion unit="seconds"
4                                  predicate="Less" value="5"/>
5     <qosp:ThroughputAssertion unit="requests"
6                               predicate="GreaterEqual" value="100"/>
7   </wsp:All>
8 </wsp:Policy>
```

Listing 5. Assertion example.

Table 3. SLA operator mapping.

SLA operator		WS-QoSPolicy operator
`And`	→	`All`
`Or`	→	`ExactlyOne`
`Not`	→	Reverse predicate
`Implies`	→	`ExactlyOne` and reverse predicate

Downloaded By: [Rosenberg, Florian] At: 10:28 24 November 2008

Attaching policies to WSDL descriptions following the WS-PolicyAttachment (Vedamuthu *et al.* 2008b) specification has two main drawbacks. First, service invocations are always subject to a policy, even if the service consumer has no corresponding SLA. Secondly, the service provider cannot differ between multiple policies for the same service since policies do not contain information about the participating parties. Therefore, following the second approach, the policies should be integrated in BPEL processes.

Extensibility in BPEL is achieved by allowing elements from other namespaces. The BPEL `partnerLink` element is the place to integrate the policy. For this integration, both synchronous (request-reply) and asynchronous (callback) message exchange patterns have to be considered. In contrast to the asynchronous case, in the synchronous case the service provider has no additional information about the service consumer, because the `partnerLink` has no service consumer-specific details. Therefore, the policy has to be integrated at the service consumer side as illustrated in Listing 6 (Lines 5–8).

Downloaded By: [Rosenberg, Florian] At: 10:28 24 November 2008

```
1  <process>
2    <partnerLinks>
3      <partnerLink name="POService"
4                   partnerLinkType="ns1:POServiceLT" partnerRole="POServiceRole">
5        <wsp:Policy xmlns:qosp="..." xmlns:wsu="..."
6                    wsu:Id="xs:QName" qosp:operation="...">
7          <!-- ... -->
8        </wsp:Policy>
9      </partnerLink>
10     <!-- ... -->
11   </partnerLinks>
12   <!-- ... -->
13 </process>
```

Listing 6. Policy integration in BPEL.

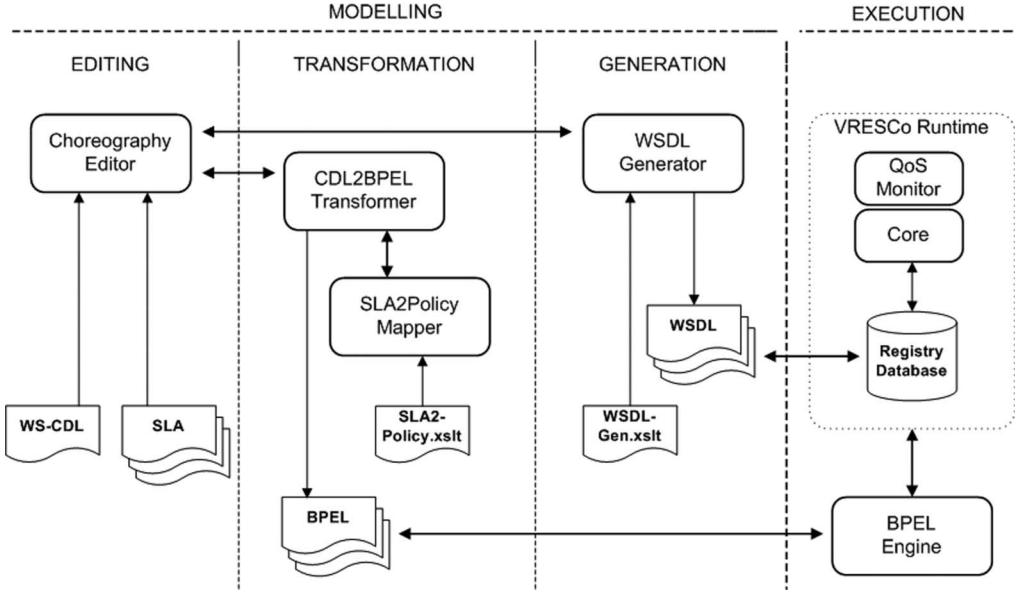


Figure 3. System architecture.

4. Architecture and implementation

The concepts and algorithms described in Section 3 have been implemented in Java using a simple Swing-based graphical user interface. The architecture of this system consists of four parts, which are depicted in Figure 3. We distinguish between the modelling and the execution phases. The modelling phase consists of editing choreographies and SLAs, transforming WS-CDL into BPEL and SLAs into policy assertions, and finally generating the WSDL artefacts. After implementing the private business logic in the BPEL artefacts and the corresponding services, they can be deployed in the VRESCo runtime as part of the execution phase.

4.1. Modelling phase

Editing choreographies in our approach is done in two steps: first, the choreography is modelled using the Pi4soa Eclipse plugin (pi4 Technologies Foundation 2008). This plugin provides a graphical tooling for creating complex choreographies. Secondly, our simple Swing-based SLA annotation tool is used to add SLA references to specific role types. Furthermore, the editor initiates the generation of WSDL and BPEL artefacts as described above.

The *Transformation* component implements the algorithms for transforming WS-CDL to BPEL, and SLA to policies. The WS-CDL to BPEL transformation is implemented using the DOM4J API whereas the SLA transformation is implemented using XSLT. During the transformation step one BPEL document is generated for each partner including the policy references which conform to the SLAs in the choreography layer.

The *Generation* component is responsible for generating the WSDL files from a choreography according to the algorithm described in Section 3. This component is also implemented using XSLT stylesheets.

4.2. Execution phase

The VRESCo runtime introduced in Michlmayr *et al*. (2007) aims at addressing some of the current challenges in service-oriented computing research and practice (Papazoglou *et al*. 2007). Among others, this includes topics related to service discovery and metadata, dynamic binding and invocation, service monitoring, QoS-aware service composition, service management and versioning (Leitner *et al*. 2008), and service notifications (Michlmayr *et al*. 2008). Besides this, another goal of the VRESCo project is to facilitate the engineering of service-oriented applications by reconciling some of these topics and abstracting from protocol related issues.

The VRESCo core services illustrated on the right-hand side of Figure 3 are responsible for publishing and querying services and associated metadata in the registry database. After generating the services in the modelling phase, the VRESCo publishing service is used to make these services available to the runtime.

The QoS monitor introduced in Rosenberg *et al*. (2006) is responsible for measuring the performance and dependability attributes of Web services. This monitor is integrated as part of the VRESCo runtime to measure the QoS attributes of the services published in the registry database. These QoS attributes are then used to monitor compliance with the SLA defined for the choreographies.

The BPEL processes generated in the modelling phase are finally executed using the Windows Workflow Foundation (WWF) that runs on top of the Microsoft .NET platform

Downloaded By: [Rosenberg, Florian] At: 10:28 24 November 2008

and provides support for executing BPEL compliant processes. WWF is integrated into VRESCo as part of its composition support.

5. Validation and discussion

In this section, we validate our work using the case study described in Section 2 to demonstrate the feasibility of our approach. Additionally, we discuss the lessons learned during the validation and describe the limitations of our approach.

5.1. Validation

For modelling the choreography we have used the Pi4soa Eclipse Plugin. During the modelling phase, the main part is to identify the partners in the process and the messages that are exchanged among the partners. Most parts of the BTO scenario are implemented in the choreography itself. However, some non-observable implementation-specific details cannot be considered from a choreography point of view but have to be implemented internally by the choreography participants.

The choreography itself is then used to generate the BPEL and WSDL templates for each partner in the choreography. The SLAs are modelled pairwise and independently among the partners. The partners agree on a set of runtime constraints that need to hold during the message interactions. In general, the SLAs are independent of the choreography itself; nevertheless, the integration of an SLA in the development process can be achieved by adding an SLA reference to a specific `roleType` in the WS-CDL (compare Listing 3 for an integration example).

In our case study we have identified four different SLAs: one between the customer and the manufacturer, and one for every manufacturer–supplier pair. For example, an SLA between the manufacturer and the CPU supplier specifies the expected response time, throughput and execution time of a service including the periods where these obligations are valid. The transformation of SLAs to policy assertions does not generate new files. Instead, after the transformation of the WS-CDL to BPEL processes, the generated BPEL process contains a `partnerLink` annotated with WS-Policy statements to express the SLOs as enforceable policies, as can be seen in Listing 4 for execution time and throughput.

After the transformation steps, the generated BPEL and WSDL files are taken as a starting point for implementing the private business logic. This mainly deals with aspects that cannot be modelled from a global viewpoint in the choreography. These internal implementations are referred to as silent actions implemented during refinement of the BPEL code.

Finally, the services and BPEL processes are deployed to an orchestration engine such as ActiveBPEL that is responsible for executing the BPEL processes. Besides this, the services are published into the VRESCo runtime, which has a number of advantages. For instance, all services and associated metadata are published within the registry database and can be queried using the VRESCo core service. Furthermore, the QoS monitor integrated into the VRESCo runtime continuously measures the QoS attributes of the services within the runtime to monitor compliance with the SLAs defined in the choreography.

5.2. Discussion and limitations

An important point of discussion is the use of WS-CDL. Some may scrutinise why we use choreographies instead of following a bottom-up approach that builds on orchestration

Downloaded By: [Rosenberg, Florian] At: 10:28 24 November 2008

languages such as BPEL. In fact, both modelling approaches are feasible and have their strengths and weaknesses. However, BPEL is intended for modelling business processes without knowledge of the global viewpoint. In contrast to this, we decided to stay close to the vision of cross-organisational choreography descriptions by using WS-CDL.

During the implementation of our case study we encountered several aspects which have to be considered when using such a top-down modelling approach. Some of these issues seem inherent to the domain of model-driven development in general. On the one hand, our approach is based on choreographies representing a global viewpoint of the business processes which raises the need for precise modelling of the global behaviour. To be more concrete, the business partners have to agree precisely on the message format used for their interaction. On the other hand, after the choreography has been initially defined, the underlying business model may evolve and lead to significant changes. Such changes clearly affect the partner processes, which causes the generation of new BPEL processes and corresponding WSDL files.

A similar problem of the top-down modelling approach is the compliance with existing choreographies. In such cases, an automated generation of the BPEL processes for each partner is not reasonable, as parts of the overall system already exist. Nevertheless, our approach can be used to generate the local BPEL code and WSDL files from an existing choreography. The QoS-specific policies then have to be attached manually because no global SLA annotations are available.

Enforcing SLA is an important aspect when considering cross-organisational business processes. So far we have focused on the modelling and integration of SLAs and QoS attributes in choreographies and orchestrations. As part of the VRESCo project, we are currently working on SLA enforcement. The goal is to monitor the QoS attributes from the orchestration layer, as described in Section 4, and trigger actions if these attributes do not comply with values defined in the SLA. Such actions may range from notifying the service providers to assigning penalties according to some penalty model.

Downloaded By: [Rosenberg, Florian] At: 10:28 24 November 2008

6. Related work

Integrating QoS in Web service based business process development has not yet received much attention, whereas the modelling of choreographies is the subject of various research activities (e.g. Decker *et al*. 2007, Zaha *et al*. 2006). In this section, we discuss existing choreography modelling and transformation approaches and the integration of policies in BPEL.

Mendling and Hafner (2007) define mapping rules for the derivation of BPEL processes from a WS-CDL choreography description. For each WS-CDL ordering structure and activity the corresponding BPEL construct respective activity is determined. These mapping rules define the basis for the mapping rules used throughout the top-down modelling process in this work. Whereas the mapping of WS-CDL to BPEL is referenced in detail, the generation of WSDL interfaces used in the BPEL processes is not addressed explicitly. In contrast to our work, the authors do not define explicit endpoint projection rules to determine which ordering structures are relevant for the participants of the choreography description. Additionally, we define mapping rules for the generation of WSDL descriptions which correspond to the service interface descriptions of the derived BPEL processes.

Diaz *et al*. (2006) use an intermediary model for the generation of BPEL processes from a WS-CDL choreography description concentrating on Web services where time constraints play a critical role. A choreography description is first transformed into a

timed automata model which is verified and validated for correctness using formal model checking techniques. This model is then further used to generate BPEL processes. In contrast to our work the focus is laid on the generation and verification of the timed automata model. Detailed mapping rules for the derivation of BPEL processes out of this model are not specified. We argue that in the context of top-down modelling it seems more appropriate to perform a direct mapping between WS-CDL and BPEL instead of using an intermediary model.

Decker *et al.* (2007) propose an new extension to BPEL called BPEL4Chor that allows modelling of choreographies within BPEL by leveraging an interconnected interface behaviour model, whereas WS-CDL represents an interaction model. As stated in their work, it has not been investigated yet which of these two approaches is more appropriate for human modellers. While we follow a top-down approach by transforming WS-CDL into BPEL, the authors propose a bottom-up approach by introducing a new choreography layer on top of BPEL. However, in contrast to our work, the integration of QoS into Web service choreographies is not addressed.

Pi4soa (pi4 Technologies Foundation 2008) is a toolset from π4 Technologies representing one of the first WS-CDL implementations. It provides a designer tool as an Eclipse plugin, which we used for modelling our choreographies, and a possibility to generate Java services from a WS-CDL. The support for generating BPEL processes is currently in progress. In contrast to pi4soa, our work considers QoS from the beginning of the development. However, it might be interesting to include the SLA/QoS related aspects into the pi4soa Eclipse plugin.

Downloaded By: [Rosenberg, Florian] At: 10:28 24 November 2008

7. Conclusions

There have been some considerable debates as to the relationship between choreography and orchestration. Some people argue that there is no need for choreography and all business interactions can, and in fact should, be modelled in BPEL. Others advocate the use of modelling by using WS-CDL but then lament the lack of execution abilities. The prime motivation for the contribution of this paper is today's lack of modelling support for QoS-aware business processes. In particular, the need for QoS-aware processes is apparent in inter-organisational business processes.

The novelty of our approach lies within the fact that we consider SLAs as first-class entities while modelling service choreographies. Our approach allows for automatic generation of executable BPEL orchestrations and WSDL files for each partner in the choreography. A novel contribution is the mapping of QoS information specified in SLAs to WS-QoS policies which are attached to the BPEL process. As a consequence, a policy-aware middleware can verify, and possibly enforce, SLAs. The approach has been implemented and the feasibility is demonstrated using a simplified version of a built-to-order case study.

As future work, we plan to apply this approach to larger case studies in order to estimate the scalability of our approach. Moreover, we are currently adding SLA enforcement capabilities to the VRESCo runtime. Finally, we envision the implementation of our tool support within the Pi4soa Eclipse plugin to allow a better integration with existing modelling tools.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under Grant Agreement 215483 (S-Cube).

We also want to thank Christian Enzi for a first prototype implementation of these mappings and the tool support, and the anonymous reviewers for their helpful comments to improve the paper.

Notes

1. In this paper we use the term Web service and service interchangeably.
2. The case study can be downloaded from http://www.vitalab.tuwien.ac.at/~florian/qosintegrator/.

References

Alves, A., *et al*., 2007. *Web service business process execution language 2.0*. Billerica, MA, USA: OASIS. Available from: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel [Accessed 1 October 2008].

Barros, A., Dumas, M., and Oaks, P., 2005. A critical overview of the web services choreography description language (WS-CDL). *BPTrends Newsletter*, 3 (3).

Carbone, M., *et al*., 2007. Structured communication-centred programming for web serices. *In*: R. De Nicola, ed. *Proceedings of the 16th european symposium on programming (ESOP'07)*, 24 March–1 April 2007, Braga, Portugal. Berlin, Germany: Springer Verlag, 2–17.

Charfi, A., Khalaf, R., and Mukhi, N., 2007. QoS-aware web service compositions using non-intrusive policy attachment to BPEL. *In*: B.J. Krämer, K.J. Lin, and P. Narasimhan, eds. *Proceedings of the 5th international conference on service-oriented computing (ICSOC'07)*, 17–20 September 2007, Vienna, Austria. Berlin, Germany: Springer Verlag, 582–593.

Decker, G., *et al*., 2007. BPEL4chor: extending BPEL for modeling choreographies. *In*: L.J. Zhang, K.P. Birman, and J. Zhang, eds. *Proceedings of the IEEE international conference on web services (ICWS'07)*, 9–13 July 2007, Salt Lake City, UT, USA. Los Alamitos, CA, USA: IEEE Computer Society, 296–303.

Diaz, G., *et al*., 2006. Automatic generation of correct web services choreographies and orchestrations with model checking techniques. *In*: IEEE, ed. *Proceedings of the international conference on internet and web applications and services (ICIW'06)*, 19–25 February 2006, Guadeloupe, French Caribbean. Los Alamitos, CA, USA: IEEE Computer Society.

Kavantzas, N., *et al*., 2005. *Web services choreography description language (WS-CDL)*. Cambridge, MA, USA: W3C. Available from: http://www.w3.org/TR/ws-cdl-10/ [Accessed 1 October 2008].

Leitner, P., *et al*., 2008. End-to-end versioning support for web services. *In*: IEEE, ed. *Proceedings of the international conference on services computing (SCC 2008)*, 8–11 July 2008, Honolulu, HI, USA. Los Alamitos, CA, USA: IEEE Computer Society.

Lohmann, N., *et al*., 2007. Analyzing BPEL4Chor: verification and participant synthesis. *In*: M. Dumas and R. Heckel, eds. *Proceedings of the 4th international workshop on web services and formal methods (WS-FM'07)*, 28–29 September 2007, Brisbane, Australia. Berlin, Germany: Springer Verlag, 46–60.

Ludwig, H., *et al*., 2003. *Web service level agreement (WSLA) language specification*. Hawthorne, NY, USA: IBM Corporation. Available from: http://www.research.ibm.com/wsla/ [Accessed 1 October 2008].

Mendling, J. and Hafner, M., 2008. From WS-CDL choreography to BPEL process orchestration. *Journal of Enterprise Information Management*, 21 (5), 525–542.

Michlmayr, A., *et al*., 2008. Advanced event processing and notifications in service runtime environments. *In*: A.P. Buchmann and S.T. Piergiovanni, eds. *Proceedings of the 2nd international conference on distributed event-based systems (DEBS'08)*, 1–4 July 2008, Rome, Italy. New York, NY, USA: ACM Press.

Michlmayr, A., *et al*., 2007. Towards recovering the broken SOA triangle – a software engineering perspective. *In*: E.D. Nitto, A. Polini, and A. Zisman, eds. *Proceedings of the second international workshop on service oriented software engineering (IW-SOSWE'07)*, 3 September 2007, Dubrovnik, Croatia. New York, NY, USA: ACM Press, 22–28.

Papazoglou, M.P., *et al*., 2007. Service-oriented computing: state of the art and research challenges. *IEEE Computer*, 40 (11), 38–45.

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

pi4 Technologies Foundation, 2006. *pi4soa*. Mountain View, CA, USA: SourceForge. Available from: http://sourceforge.net/projects/pi4soa [Accessed 25 January 2008].

Downloaded By: [Rosenberg, Florian] At: 10:28 24 November 2008

Rosenberg, F., *et al*., 2007. Integrating quality of service aspects in top-down business process development using WS-CDL and WS-BPEL. *In*: IEEE, ed. *Proceedings of the 11th enterprise computing conference (EDOC'07)*, 15–19 October 2007, Annapolis, MD, USA. Los Alamitos, CA, USA: IEEE Computer Society, 15–26.

Rosenberg, F., Platzer, C., and Dustdar, S., 2006. Bootstrapping performance and dependability attributes of web services. *In*: IEEE, ed. *Proceedings of the IEEE international conference on web services (ICWS'06)*, 18–22 September 2006, Chicago, IL, USA. Los Alamitos, CA, USA: IEEE Computer Society.

Vedamuthu, A.S., *et al*., 2007a. *Web services policy 1.5 – attachment*. Cambridge, MA, USA: W3C. Available from: http://www.w3.org/TR/ws-policy-attach/ [Accessed 25 January 2008].

Vedamuthu, A.S., *et al*., 2007b. *Web services policy 1.5 – framework*. Cambridge, MA, USA: W3C. Available from: http://www.w3.org/TR/ws-policy/ [Accessed 25 January 2008].

Völter, M. and Stahl, T., 2006. *Model-driven software development: technology, engineering, management*. Hoboken, NJ, USA: John Wiley & Sons.

Zaha, J.M., *et al*., 2006. Let's dance: a language for service behavior modeling. *In*: R. Meersman and Z. Tari, eds. *Proceedings of the 14th international conference on cooperative information systems (CoopIS'06)*, 29 October–3 November 2006, Montpellier, France. Berlin, Germany: Springer Verlag.

Downloaded By: [Rosenberg, Florian] At: 10:28 24 November 2008