Choreography Design Using WS-BPEL

Oliver Kopp, Frank Leymann
Institute of Architecture of Application Systems, University of Stuttgart
{kopp,leymann}@iaas.uni-stuttgart.de

Abstract

Web Services are the state-of-the-art realization of a service-oriented architecture. While there is an agreed standard to describe the interface of services (WSDL) as well as an agreed standard to describe the behavior of a single process (WS-BPEL), there is no agreed standard to describe choreographies. In this paper, we give an overview about existing approaches to model choreographies and present one approach based on WS-BPEL in detail.

1 Introduction

The service-oriented architecture (SOA) is an architectural style based on the services paradigm. The most popular realization of the SOA paradigm are Web Services [1]: each service is offered as Web Service. Web Services can be combined to form a business process using the Web Services Business Process Execution Language (WS-BPEL, BPEL for short). A BPEL process is in turn offered as Web Service, which enables recursive composition. Forming business processes out of services is called "orchestration". When multiple processes interact with each other, orchestrations describe the point of view of a single process only. In contrast to orchestrations, choreographies describe the interplay between processes from a global perspective. While orchestrations are well understood, choreographies are an open research field. In this paper, we give an overview about the state-of-the-art in choreography modeling and provide detail insight on a choreography language proposal based on BPEL.

Choreographies are used to capture collaborations between multiple business partners from a global perspective. While most of the published scenarios originate from a top-down approach, another use-case for choreographies is a bottom-up approach: for example, if a company acquires another company, the business processes of both have to be adapted to be able to work together and thus to make use of the synergy effects. Important reasons to design choreographies are acquisitions and merges between companies and the formation of virtual enterprises.

In the following, we use a RosettaNet Partner Interface Process (PIP) to illustrate choreography design. RosettaNet is an industry consortium defining "high-value process scenarios that deliver manufacturing quality data, end-to-end supply chain visibility, and legislative compliance" [12]. The process scenarios are described using interconnection models. In an interconnection model, the behavior of each participant and the messages exchanged are shown. A typical PIP is the PIP 3A1 "Request Quote" defined in RosettaNet Cluster 3 "Order Management". There, a buyer decides whether he needs to place an order. If yes, he specifies his quote request

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

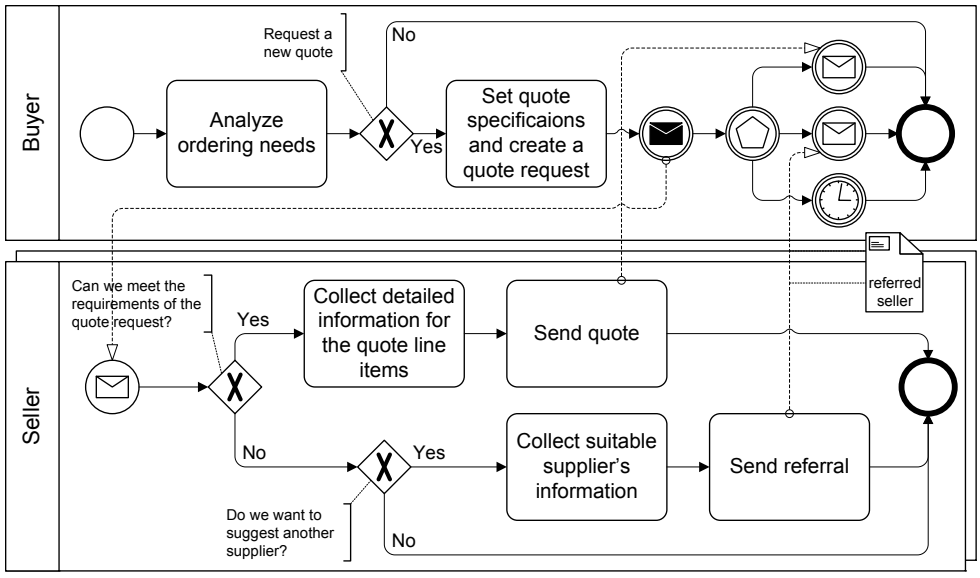

Figure 1: PIP 3A1: Request Quote [13]. Modeled using BPMN with choreography extensions

and sends the quote request to a seller. The seller in turn decides whether he meets the requirements of the quote request. If so, he replies with a quote. If the seller does not meet the requirements, he decides whether he can suggest another supplier. If yes, he sends the suggestion back. If not, he does nothing. Figure 1 presents the BPMN representation of the PIP. We use BPMN V1.1 and the choreography extensions presented in [4]. The shaded pool denotes that there are multiple sellers involved in the choreography. The referenced passed on the message flow is explicitly modeled and associated with the last message flow between the seller and the buyer. In the graphical representation, we assume that each pool is realized by one process. To ensure proper termination of the buyer, we had to include a timeout to handle the case that the supplier does not send any quote and does not send any referral.

In general, in the field of choreography design, there are three issues to tackle: (i) modeling of a choreography, (ii) verification of the choreography and finally (iii) mapping of the choreography to the runtime. In case of choreography modeling, the language to express the choreography has to have well-defined semantics and needs to be suitable to capture choreographies. When a choreography is modeled, the model itself has to be checked for modeling errors: the model has to be consistent within itself (e.g., not contain any deadlock and always reach an end state) and has to fulfill certain constraints (e.g., given by logical formulas). When it comes to execution, the semantics of the choreography has to be captured by local processes, which have to be capable to enact the constraints defined by the choreography.

Currently, there are three main approaches to model choreographies: interaction models, interconnection models and declarative models. Interaction models use the interaction a basic building block. In contrast to interaction models, the main idea of interconnection models is to be close to the execution and to re-use the idea of abstract processes: activities of the local abstract processes are interconnected. An abstract process itself leaves out process internal details, which are not needed to describe the interaction with the partners. While interaction and interconnection models describe all possible interaction schemes, declarative models define constraints on the execution. Thus, declarative models specify the "borders" of possible execution, but do not enumerate explicitly all possible executions [10].

Current languages to specify interaction models are for example the Web Service Choreography Description Language (WS-CDL, [5]) and extensions to BPMN for interaction modeling (iBPMN, [2]). While these languages are suited to capture the interactions between services on a higher level, the runtime-support of them is an open

field. The current solution is to map parts of the choreography specification to abstract BPEL process models, which are then refined and executed. However, not all constraints can be directly mapped to BPEL. For example, there is currently no solution to map the blocking wait of WS-CDL to BPEL. In the case of declarative process models, the mapping to BPEL is a complete open research field.

Orchestrations of Web services are mainly defined in BPEL. BPEL has native support for concurrency, backward and forward recovery. To enable modularity and composability, a choreography language should use the same control-flow semantics as an orchestration language to close the gap between choreography specification and runtime. While there is a mapping from BPMN to BPEL available [9], BPMN does not have the expressiveness to specify all the behavior which can be expressed by BPEL constructs. For example, event handlers and termination handlers cannot be modeled using BPMN. Furthermore, a BPEL process can be used to specify the behavior of one participant only. Therefore, we proposed extensions to BPEL to lift BPEL from an orchestration language to a full choreography language (BPEL4Chor [3]). In addition, we added constructs to BPMN to enable modeling choreographies using BPMN including a BPMN representations of BPEL constructs (`http://www.bpel4chor.org/editor`, [11]).

2 BPEL4Chor

BPEL4Chor itself consists of three artifacts: (i) participant behavior descriptions, (ii) a topology description and (iii) a participant grounding.

The *participant behavior descriptions* are abstract BPEL processes describing the behavior of each participant. "Abstract BPEL" denotes that the BPEL processes have to be refined to be fully deployable and to be executed on a BPEL engine. The steps going from an abstract BPEL processes to an executable BPEL process are called "executable completion" and are mainly manual work. It is important to note, that WSDL port types and WSDL operations are not used in the participant behavior descriptions. This allows to specify the behavior of a participant without the fixed connection to concrete realizations. The concrete WSDL information is brought in during the participant grounding.

The BPEL4Chor *topology* provides a global view on the choreography: it defines the participants and the message links. A message link connects communicating activities and corresponds to a message flow in BPMN. The concept of a message link allows to wire existing orchestrations to provide a global view on the interaction.

We see BPEL as orchestration standard and WSDL as standard to describe interfaces. Therefore, the *grounding* brings in the necessary WSDL information to enact the choreography. This information can then be used to generate abstract BPEL containing partner links, port types and operations. These BPEL processes can then serve as basis for the executable completion. However, it is not necessary to implement a participant using BPEL. A participant can also be realized by one or more Web services implemented in any language as long as the behavior of these Web services corresponds to the given participant behavior description.

A BPEL4Chor choreography can be verified using an approached based on Petri nets presented in [8]. There, the choreography's participants are translated into Petri nets. These nets are then connected according to the BPEL4Chor choreography. If there are multiple participants involved, the respective net is copied accordingly to reflect the multiple instances. The resulting Petri net can be checked for deadlocks or any other desired property using model checking tools. Experiments showed that choreographies with up to thousand instances can be verified [8]. In case a deadlock is found in the choreography, the faulty participant can be fixed automatically [7]. All results of the verification (e.g., deadlock traces) can be mapped back to the original BPEL processes. This allows for a seamless integration of choreography verification into the process of choreography modeling.

If an executable BPEL process was modeled based on a participant behavior description, it has to be checked, whether the executable process conforms to the participant behavior description. A general approach to check conformance of BPEL processes is presented in [6].

3 Summary

We presented an overview of choreography design and BPEL4Chor. We showed how existing technologies can be re-used to describe a choreography: BPEL is used to define the participant behavior descriptions and WSDL is brought in at the grounding to enable the message exchange via an Enterprise Service Bus. The BPEL4Chor topology is the first proposal enabling interconnection of BPEL activities.

BPEL4Chor is part of the Tools4BPEL project and is funded by German Federal Ministry of Education and Research (project number 01ISE08). The other partners involved are the Humboldt-Universität zu Berlin and the MEGA International GmbH. In the project, our task is to investigate the modeling of sub-processes, choreographies, cross-partner fault handling, cross-partner transactions and sub-processes using BPEL. The part of the Humboldt-Universität zu Berlin is to provide verification mechanisms and tools for BPEL as well as for our extensions of BPEL. Finally, MEGA delivers challenging examples guiding and driving our research.

References

[1] F. Curbera, F. Leymann, T. Storey, D. Ferguson, and S. Weerawarana. *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More*. Prentice Hall PTR, 2005.

[2] G. Decker and A. P. Barros. Interaction Modeling Using BPMN. In *1st International Workshop on Collaborative Business Processes (CBP)*. Springer, 2007.

[3] G. Decker, O. Kopp, F. Leymann, and M. Weske. BPEL4Chor: Extending BPEL for Modeling Choreographies. In *ICWS*. IEEE Computer Society, 2007.

[4] G. Decker and F. Puhlmann. Extending BPMN for Modeling Complex Choreographies. In *CoopIS*. Springer, 2007.

[5] N. Kavantzas, D. Burdett, G. Ritzinger, and Y. Lafon. Web Services Choreography Description Language Version 1.0. W3C Candidate Recommendation, W3C, November 2005.

[6] D. König, N. Lohmann, S. Moser, C. Stahl, and K. Wolf. Extending the Compatibility Notion for Abstract WS-BPEL Processes. In *International Conference on World Wide Web*. ACM, 2008.

[7] N. Lohmann. Correcting Deadlocking Service Choreographies Using a Simulation-Based Graph Edit Distance. In *BPM*. Springer, 2008.

[8] N. Lohmann, O. Kopp, F. Leymann, and W. Reisig. Analyzing BPEL4Chor: Verification and Participant Synthesis. In *WS-FM*. Springer, 2007.

[9] C. Ouyang, M. Dumas, S. Breutel, and A. H. M. ter Hofstede. Translating Standard Process Models to BPEL. In *CAiSE*. Springer, 2006.

[10] M. Pesic, M. H. Schonenberg, N. Sidorova, and W. M. P. van der Aalst. Constraint-based workflow models: Change made easy. In *CoopIS*, 2007.

[11] K. Pfitzner, G. Decker, O. Kopp, and F. Leymann. Web Service Choreography Configurations for BPMN. In *WESOA*. Springer, 2007.

[12] RosettaNet. Home page. `http://www.rosettanet.org/`.

[13] RosettaNet. *Overview: Clusters, Segments, and PIPs*. Version 02.04.00.