



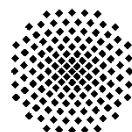
The Need for a Choreography-aware Service Bus

Oliver Kopp, Tammo van Lessen, Jörg Nitzsche

Institute of Architecture of Application Systems, University of Stuttgart, Germany
{kopp,vanlessen,nitzsche}@iaas.uni-stuttgart.de

BIB_TE_X:

```
@inproceedings{ChorAwareESB,  
  author    = {Kopp, Oliver and van Lessen, Tammo and Nitzsche, J\"{o}rg},  
  title     = {The Need for a Choreography-aware Service Bus},  
  booktitle = {The 3rd European Young Researchers Workshop on Service Oriented  
Computing (YRSOC 2008)},  
  year      = {2008},  
  pages     = {30-36}  
}
```



The Need for a Choreography-aware Service Bus^{*}

Oliver Kopp, Tammo van Lessen, and Jörg Nitzsche

Institute of Architecture of Application Systems
University of Stuttgart
Universitätsstraße 38, 70569 Stuttgart, Germany
{oliver.kopp,tammo.van.lessen,joerg.nitzsche}
@iaas.uni-stuttgart.de
<http://www.iaas.uni-stuttgart.de>

Abstract Choreographies offer means to describe the long-running collaboration of business partners. Such descriptions can be used to create new participant processes which comply to the overall choreography or to check whether participating processes conform to the protocol. In addition, choreography descriptions allow for asserting whether a completed cross-organizational conversation has been compliant to the planned choreography. However, choreography descriptions have so far not been used during execution but only during design time. Therefore, it is not yet possible to immediately detect protocol violations and to instantly handle such violations. In this paper we motivate the need of a Choreography-aware Service Bus which is capable of tracking the soundness of cross-organizational conversations while they are running. This fosters a novel notion of exception handling in the context of choreographies.

Key words: Compliance, Choreography, Service Bus

1 Introduction

Choreographies offer means to describe the collaboration of business partners. They can be modeled by (i) interconnection models or (ii) interaction models. An *interconnection model* captures the observable behavior of each participant in a choreography. It defines a “network of bilateral interactions”, i.e. it defines for each participant in which order it has to send and receive messages to and from its partner services. BPMN [1] and BPEL4Chor [2] are languages to express choreographies by interconnection models using sending and receiving activities. An *interaction model* defines an ordering of process interactions from

^{*} The work published in this article is funded by the SUPER project under the EU 6th Framework Programme Information Society Technologies Objective (contract no. FP6-026850, <http://www.ip-super.org/>) and the Tools4BPEL project, which in turn is funded by the German Federal Ministry of Education and Research (project no. 01ISE08).

a global point of view, i.e. it defines a process “in the middle” that captures the interactions of all participants. Current languages providing interaction models are Let’s Dance [3] and WS-CDL [4].

In case an interaction model was used to model the choreography, the behavioral description of the participants can be generated. When using an interconnection model, they are already available. These descriptions can be used to either find appropriate participants via conformance checking, or to implement new ones. When a choreography is executed, the choreography itself does not get executed but the participant behaviors described in the choreography are executed. That means, the choreography as an artifact is not used during execution. Since there are constraints that cannot be transformed into local behavior, not all constraints of a choreography can be enforced [5]. An approach is to check the audit log on whether the participants complied with the choreography specification [6]. However, if choreography execution lasts for years and an early message exchange was wrong, it is not desirable to check this choreography violation after the last message. It has to be possible to immediately react on faulty messages to prevent aftereffects.

A Service Bus [7] is middleware to connect services with each other. Services are implementations of participants. A Service Bus provides virtualization of services and thus contributes to a loosely coupled environment. However, besides the lack of a built-in choreography conformance checking, the bus so far does not offer a possibility for a service to register as participant of a choreography and participant virtualization. Such a registration allows for an automatic binding of participants to a choreography. Therefore we propose to make a Service Bus choreography-aware, i.e. to extend it with choreography specific features. For instance, choreography based discovery and conformance checking, which enables the bus to react to violations of the choreography specification in “realtime”.

The remainder of the paper is structured as follows: Section 2 shows the state of the art in choreography implementation and describes the limitations of traditional Service Bus when employed for execution. The idea of a Choreography-aware Service Bus is presented in section 3. Section 4 presents related work and section 5 concludes the paper.

2 Limitations of a Traditional Service Bus

Figure 1 presents a sample choreography for investments. A client talks to a financial adviser. The adviser recommends an investment and hands out the requested information material. The government requires to give the customer at least 24 hours to decide on the investment. After 24 hours have passed, the customer may sign the contract. It is not allowed to receive a signature from the customer beforehand.

One approach to implementing a choreography is to transform the choreography to orchestrations [8]. In contrast to a choreography which describes the interaction of all participants from a global view, an orchestration implements the behaviour of a single participant. The Business Process Execution Language

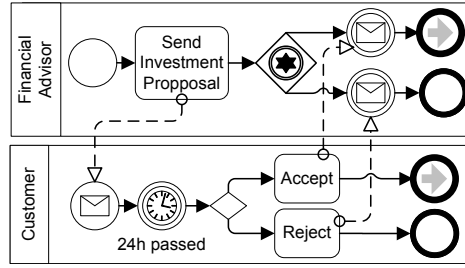


Figure 1. Process of investment offers

(BPEL, [9]) is the de facto standard to describe Web Service orchestrations. Figure 2 presents the steps required to derive executable BPEL processes from a choreography descriptions. These steps are independent from the choreography language chosen. First, the behavior of each participant in the choreography is mapped to an abstract BPEL process. For each participant, the BPEL process is enriched with technical details, such as message types, variables and internal behavior to get an executable BPEL process. These processes then need to be deployed to an infrastructure.

Figure 3 illustrates the scenario for the case in which a traditional Service Bus is used. The Service Bus routes the messages to the corresponding process instance. Each workflow engine generates audit logs. These logs can then be used to check whether the sent messages conform to the choreography. However, a wrong message cannot be held back. If the customer accepts after 12 hours, the bank may continue with the investment. The bank could automatically debit the current account of the customer, transfer the invested money to a third party, etc. If an error is detected by the monitoring application, it could send a message to the bank and exception handling begins. For instance, money transfer has to be compensated.

A crucial point in choreography design is the “local enforceability”. A *locally enforceable* choreography is defined as follows: “The global model can be mapped into local ones in such a way that the resulting local models satisfy the following two conditions: (i) they contain only interactions described in the global model;

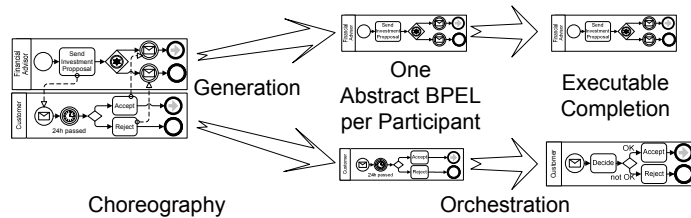


Figure 2. From Choreography to Orchestration

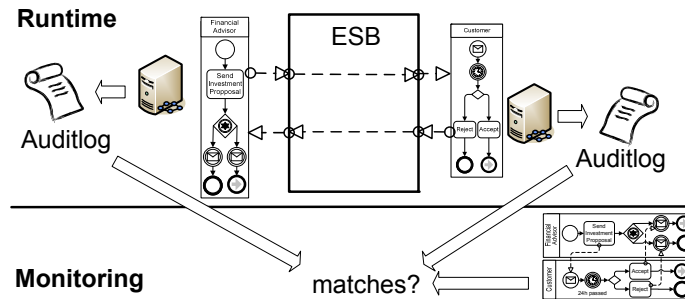


Figure 3. Infrastructure realized with a traditional Service Bus

and (ii) they are able to collectively enforce all the constraints expressed in the global model.” [5]. A locally unenforceable choreography is presented in Figure 4: Sender A sends a message to B. Afterwards C sends a message to D. The interaction from C to D must happen after the interaction from A to B. It is not possible to globally enforce a choreography using a traditional Service Bus. The choreography has to be modified to be locally enforceable.



Figure 4. Locally unenforceable choreography [5]

3 Choreography-aware Service Bus

In contrast to the traditional approach, a Choreography-aware Service Bus checks each message to route whether it conforms to the choreography. Figure 5 presents the infrastructure when using a Choreography-aware Service Bus. The choreography-conformance checking is built in: If a message conforms to the choreography, it is routed as in a traditional Service Bus. If a message does not conform to the choreography, the Service Bus does not route the message to the recipient. That means, the wrong message does not trigger any further processing. The bank does not automatically debit the giro account, does not transfer the invested money to a third party, etc.

In general, there are several options to treat a wrong message:

1. Move the message to a dead letter queue
2. Notify the sender that it violated the choreography
3. Trigger default exception handling
4. Trigger a pre-defined exception handling
5. Enforce the choreography

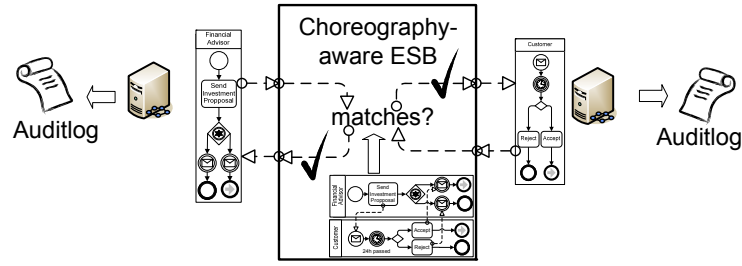


Figure 5. Infrastructure realized with a Choreography-aware Service Bus

Exception handling can include stopping the whole choreography. Another exception handling approach is to only disable messages, which are related to the wrong message. The last option (5) is not applicable in all cases: If the customer agrees too early, the Service Bus can technically hold the message back and resend the message later. However, this is not the intended behavior. In the case of the unenforceable choreography presented in Figure 4, the message sent from C to D can be hold back until the message from A to B has been sent.

4 Related Work

Usually the term “choreography” is used for models describing the global behaviour of multiple participants. However, sometimes the term “choreography” is used to describe the behavioural interface of a single service, like for instance used in the Web Service Modelling Ontology (WSMO, [10, 11]). The Web Service Modelling eXecution environment (WSMX, [12]) is the reference implementation for WSMO and contains a so called “choreography engine”. This choreography engine, however, only supports bilateral message exchanges, i.e. choreographies involving multiple participants are not taken into account.

An overview of languages that enable modelling choreographies involving multiple participants is given in [8]. The service interaction patterns can be used to determine the kinds of service interactions a choreography language supports. While BPEL4Chor and Let’s Dance have full support for a priori unknown sets of participants, the possible participants have to be known in advance in WS-CDL [13]. For example, a request-for-bid scenario with a priori unknown bidders cannot be modeled using WS-CDL.

Verification techniques are available for BPEL4Chor [14], BPMN [15, 16], Let’s Dance [17] and WS-CDL [18]. The verifications ensure that the choreographies are deadlock free or satisfy given properties. The given properties may originate from compliance requirements. Thus, the approaches ensure that the choreography is compliant but do not deal with the realization of a choreography. If a choreography is compliant, a Choreography-aware Service Bus ensures that the participants in that choreography behave according to the choreography and thus are compliant, too.

There are various methods to check conformance between a choreography and orchestrations at design time [19–23]. The drawback of these methods is that at least the behavior description, e.g. expressed in abstract BPEL, of the used services has to be available. If that is not possible, e.g. for services of other companies, the approaches cannot prove whether the opaque service implementation adheres to the choreography specification.

[5, 24] present methods for mapping choreographies to participants such that the participants conform to the choreography. While [24] proposes to ignore unenforceable constraints, [5] requests to reject a choreography which is not enforceable. [25] presents an approach to check whether a choreography is locally enforceable.

Since it is not possible to ensure at runtime that a service implementation conforms to the choreography, runtime monitoring is desirable. [6] presents an approach, where the monitoring data is used to determine whether an obligation has been carried out successfully. The determination is carried out after the execution of the choreography. We presented an approach to ensure conformance at runtime and to prevent false message exchanges.

5 Conclusion and Outlook

We presented how choreography conformance can be realized using a traditional Service Bus. We showed that messages not following the choreography requirements are routed to their destination. A solution is to check the audit logs and to start exception handling if a wrong message was sent.

To overcome the shortcomings of a traditional Service Bus, we presented the concept of a Choreography-aware Service Bus. Using a Choreography-aware Service Bus, messages not compliant to the choreography are not routed to the recipient, but handled in other ways.

Our ongoing work is to evaluate whether existing choreography languages are suitable to be used in a Choreography-aware Service Bus and if there is a need to extend them to specify exceptional behavior.

References

1. OMG: Business Process Modeling Notation, V1.1. OMG Available Specification, Object Management Group (2008)
2. Decker, G., Kopp, O., Leymann, F., Weske, M.: BPEL4Chor: Extending BPEL for Modeling Choreographies. In: ICWS 2007
3. Zaha, J.M., Barros, A., Dumas, M., ter Hofstede, A.: Let’s Dance: A Language for Service Behavior Modeling. In: CooPIS 2006. LNCS
4. Kavantzias, N., Burdett, D., Ritzinger, G., Lafon, Y.: Web Services Choreography Description Language Version 1.0. W3C Candidate Recommendation, W3C (2005)
5. Zaha, J.M., Dumas, M., ter Hofstede, A.H., Barros, A., Decker, G.: Service Interaction Modelling: Bridging Global and Local Views. EDOC 2006
6. Sailer, M., Morciniec, M.: Monitoring and Execution for Contract Compliance. Technical Report HPL-2001-261, Hewlett Packard Laboratories (2001)

7. Leymann, F.: The (Service) Bus: Services Penetrate Everyday Life. In: ICSOC 2005. LNCS
8. Decker, G., Kopp, O., Barros, A.: An introduction to service choreographies. *Information Technology* **50**(2/2008) (2008)
9. OASIS WS-BPEL TC: Web services business process execution language version 2.0. Technical report, OASIS (2007)
10. Lausen, H., Polleres, A., Roman, D.: Web Service Modeling Ontology (WSMO). W3C Member Submission **3** (2005)
11. Roman, D., Scicluna, J., Nitzsche, J.: D14 V 1.0: Ontology-based Choreography (2007) <http://www.wsmo.org/TR/d14/v1.0/>.
12. Haller, A., Cimpian, E., Mocan, A., Oren, E., Bussler, C.: WSMX - A semantic service-oriented architecture. ICWS 2005
13. Decker, G., Overdick, H., Zaha, J.M.: On the Suitability of WS-CDL for Choreography Modeling. In: EMISA 2006. LNI
14. Lohmann, N., Kopp, O., Leymann, F., Reisig, W.: Analyzing BPEL4Chor: Verification and Participant Synthesis. In: WS-FM 2007. LNCS
15. Puhmann, F., Weske, M.: Investigations on Soundness Regarding Lazy Activities. In: BPM 2006. LNCS
16. Ghose, A., Koliadis, G.: Auditing business process compliance. In: ICSOC 2007
17. Decker, G., Zaha, J.M., Dumas, M.: Execution Semantics for Service Choreographies. In: WS-FM 2006. LNCS
18. Flavio Corredini, Francesco De Angelis, A.P.: Verification of WS-CDL choreographies. In: YR-SOC 2007
19. Busi, N., Gorrieri, R., Guidi, C., Lucchi, R., Zavattaro, G.: Choreography and Orchestration Conformance for System Design. In: COORDINATION 2006. LNCS
20. Li, J., Zhu, H., Pu, G.: Conformance Validation between Choreography and Orchestration. In: TASE 2007
21. Hongli, Y., Xiangpeng, Z., Chao, C., Zongyan, Q.: Exploring the Connection of Choreography and Orchestration with Exception Handling and Finalization/Compensation. In: FORTE 2007. LNCS
22. M.Bravetti, Zavattaro, G.: Towards a Unifying Theory for Choreography Conformance and Contract Compliance. In: 6th International Symposium on Software Composition (SC'07). LNCS
23. Aalst, W., Dumas, M., Ouyang, C., Rozinat, A., Verbeek, H.M.W.: Choreography conformance checking: An approach based on BPEL and petri nets (extended version). BPM Center Report BPM-05-25, BPMcenter.org (2005)
24. Mendling, J., Hafner, M.: From Inter-organizational Workflows to Process Execution: Generating BPEL from WS-CDL. In: MIOS 2005
25. Decker, G., Weske, M.: Local Enforceability in Interaction Petri Nets. In: BPM 2007. LNCS

All links were last followed on May 8, 2008.