Service Referrals in BPEL-based Choreographies

Gero Decker[1], Oliver Kopp[2], Frank Puhlmann[1]

[1] Hasso Plattner Institut, University of Potsdam, Germany
{gero.decker,frank.puhlmann}@hpi.uni-potsdam.de
[2] Institute of Architecture of Application Systems, University of Stuttgart, Germany
oliver.kopp@iaas.uni-stuttgart.de

Abstract. Choreographies describe the interactions between two or more services from a global perspective and specify allowed service conversations. Choreographies typically do not rely on static binding, i.e. the participating services are not selected at design-time of the choreography. Some services might only be selected at runtime and this selection has to be propagated in the case of multi-lateral conversations. Hence, the notion of service referrals (also called link passing mobility) is recurrent in choreographies. In past work, we have proposed BPEL extensions for describing service choreographies, namely BPEL4Chor. This paper closely investigates the link passing mobility capabilities of BPEL4Chor and illustrates their semantics using π-calculus.

1 Introduction

Service-oriented architecture (SOA) is an architectural style for information systems that relies on message exchanges between loosely coupled services [4]. Web services are typically used for implementing an SOA. The first standards in the field of Web services such as XML, WSDL, and SOAP put simple request/response interactions between services into the center of attention. Further standards like BPEL [10] enable the implementation of services that engage in more complex interaction scenarios with its environment. This second generation of Web services supports long-running conversations in bilateral and multi-lateral settings.

As BPEL only considers conversations from the perspective of an individual service, a new viewpoint was proposed to capture conversations from a global point of view. These *choreographies* define allowed conversations and therefore serve as interaction agreement between different parties. In some choreographies it is already defined which concrete services are to participate in the conversations. Imagine e.g. a collaboration between two companies who defined their respective interaction behavior in the choreography. In other choreographies, a notion of roles or participant types can be found, leaving it open to select participating services just before starting a conversation or even after the conversation has already started. As typically more than two services participate in such conversations, it is important to pass on the reference to the concrete service during the conversation. The Service Interaction Patterns [1], a catalog of common patterns in interaction scenarios, highlight such *link passing mobility* as recurrent phenomenon under the

name of *Request with Referral*. We therefore conclude that support for link passing mobility is an essential feature of choreography description languages. As an alternative to existing languages and in order to enable more direct integration of service orchestrations and choreographies, we have introduced BPEL extensions for choreography modeling (BPEL4Chor) in [5]. In this paper we are going to closely investigate how link passing mobility is realized in BPEL4Chor. In order to provide unambiguous semantics we use π-calculus, a modern process algebra that inherently supports link passing mobility. An extended discussion on the advantages can be found in [12].

The remainder of this paper will be organized as follows. The next section discusses related work, before section 3 gives a short overview of BPEL4Chor. The main contribution will be found in section 4 where link passing mobility in BPEL4Chor is discussed. Section 5 concludes and points to future work.

2 Related Work

Since the formal semantics of BPEL4Chor is based on π-calculus, we refer to earlier work on the formal representation of process and interaction patterns [13,7]. Dynamic binding in π-calculus is introduced in [11]. In a nutshell, the π-calculus is based on a set of agent identifiers (denoted with uppercase letters) and another set of names (denoted with lowercase letters). Names are a unification of concepts known as pointers, links, channels, etc. The agents of the π-calculus can interact by sending names via names used as channels, denoted as $\overline{a}\langle b\rangle$, and receiving names via names used as channels, denoted as $a(x)$. The ordering of the send and receive operations can be sequential, denoted as $a(b).\overline{b}\langle x\rangle.\mathbf{0}$, parallel, denoted as $a(b).\mathbf{0} \mid \overline{x}\langle y\rangle.\mathbf{0}$, or exclusive, denoted as $a(b).\mathbf{0} + \overline{x}\langle y\rangle.\mathbf{0}$. Each execution path is terminated with $\mathbf{0}$. Furthermore, agents can create new, unique names during their execution, denoted as νx, where x is the new name. Due to space limitations, we refer to [9] for an extended introduction. Existing approaches for formalizing BPEL do not support dynamic binding and are hence improper for an extension to choreographies [8,3].

A strong competitor for BPEL4Chor is given by WS-CDL as a choreography language. While WS-CDL is able to support most of the service interaction patterns, it also introduces different realizations for the workflow patterns. Notable, these are difficult to map to BPEL [6]. Since BPEL is the state-of-the-art orchestration language for business processes, a mismatch between choreography and orchestration languages should be avoided. This paper focuses on an extension of BPEL to overcome these limitations. Other competitors are given by BPML [2] and BPSS [14]. However, both are outdated nowadays.

3 BPEL4Chor Overview

In contrast to other choreography languages such as BPSS and WS-CDL, BPEL4Chor does not have interactions as basic building blocks but rather communication actions, i.e. send and receive actions. Therefore, control flow

Listing 1 Participant behavior description for a migration service

```
<process name="migrationservice" targetNamespace="urn:visa:ms"
  abstractProcessProfile="urn:HPI_IAAS:choreography:profile:2006/12">
  <sequence>
    <receive wsu:id="ReceiveEmployeeDetails" createInstance="yes" />
    <opaqueActivity name="PrepareVisaApplication" />
    <invoke wsu:id="SubmitVisaApplication" />
    <receive wsu:id="ReceiveConfirmation" />
  </sequence>
</process>
```

dependencies are not defined between interactions but locally between communication actions. BPEL4Chor uses *participant behavior descriptions* (PBDs) for this purpose. For each participant type a PBD has to be provided. PBDs are a special kind of abstract BPEL processes. This enables to define control and data flow in choreographies as it is the case in BPEL.

In contrast to classic BPEL, where send and receive actions include information about who the respective interaction partner is (through the `partnerLink` and `operation` attributes), PBDs have to be glued together in a separate artifact, the *participant topology*. This document captures the structural aspects of the choreography and defines which two communication actions from the PBDs are connected through a message link. While the PBDs and the topology are free of web-service-specific configuration, *participant groundings* are introduced to provide the mapping of elements in the topology to WSDL specifications.

Listings 1 and 2 show two BPEL4Chor artifacts of a choreography description, where a visa is to be organized for a new employee. As the employing company has outsourced all migration related activities, it sends the employee's details to a migration service. This service prepares and submits a visa application to the government's immigration office. The immigration office sends a nomination approval to the employer which is needed for picking up the visa from the embassy. In addition, a confirmation is sent to the migration service.

4 Link Passing Mobility in BPEL4Chor

The example from the previous section illustrates the main concepts in BPEL4Chor. While merely control and data flow is defined in the participant behavior descriptions, the main structural setting can be found in the topology. Here, participant types and participant references are defined. It is possible that several references or even reference sets are used for one participant type. This indicates that different participants of the same type are involved in one conversation. Imagine e.g. a logistics scenarios where several shippers transport goods from a production site to a warehouse or imagine a bidding scenario where different bidders take part in one auction.

Listing 2 Participant topology for the visa application scenario

```
<topology name="visa" targetNamespace="urn:visa" xmlns:ms="urn:visa:ms">
  <participantTypes>
    <participantType name="MigrationService"
     participantBehaviorDescription="ms:migrationservice" />
    <participantType name="Employer" ... />
    <participantType name="ImmigrationOffice" ... />
  </participantTypes>
  <participants>
    <participant name="e" type="Employer" selects="ms" />
    <participant name="ms" type="MigrationService" />
    <participant name="io" type="ImmigrationOffice" />
  </participants>
  <messageLinks>
    <messageLink name="employeeDetailsLink" sender="e"
     sendActivity="SubmitEmployeeDetails" receiver="ms"
     receiveActivity="ReceiveEmployeeDetails"
     messageName="employeeDetails" />
    <messageLink name="visaApplicationLink" sender="ms"
     sendActivity="SubmitVisaApplication" receiver="io"
     receiveActivity="ReceiveVisaApplication"
     messageName="visaApplication" participantRefs="e" />
    <!-- ... -->
  </messageLinks>
</topology>
```

Although participant references are defined on a global level, not all participants necessarily know about all other participants involved. Through the receipt of messages or through explicit link passing the knowledge about participants is propagated. The immigration office knows which migration service is involved in the conversation as it receives a message from it. On the other hand, the office gets to know the employing company through the mechanism of link passing mobility. The migration service passes the reference to this company on to the immigration office as part of the visa application.

The notion of participant references cannot be directly found in π-calculus. On the other hand, send and receive activities are mapped to input and output actions on a π-channel, leading to the fact that message links from BPEL4Chor are represented by one or several π-channels. Several channels are needed in the case of several participants of the same type taking part in the conversation. We therefore introduce the term *message link instance* for corresponding to the actual connection between two participants in a conversation. The example from the previous section could be formalized as follows:

$$E \stackrel{def}{=} (\nu details, na)\ \overline{ed}\langle details, na\rangle.na(approval).\mathbf{0}$$

$$MS \stackrel{def}{=} (\nu c, application)\ ed(details, na).\overline{va}\langle application, c, na\rangle.c(conf).\mathbf{0}$$

$$IO \stackrel{def}{=} (\nu approval, conf)\ va(application, c, na).(\overline{na}\langle approval\rangle.\mathbf{0} \mid \overline{c}\langle conf\rangle.\mathbf{0})$$
$$SYS \stackrel{def}{=} (E \mid MS \mid IO)\ .$$

The message link `employeeDetailsLink` is represented by channel ed and `visaApplicationLink` by va. We see that ed and va are free names. This indicates that there is a static binding between the employer E and the migration service MS as well as between MS and the immigration office IO. In order to explicitly represent dynamic selection of the migration service by the employer (which is indicated by the `selects` attribute in the participant topology) a broker B could be introduced into the formalization:

$$SYS' \stackrel{def}{=} (E' \mid MS \mid IO \mid B) \text{ with } E' \stackrel{def}{=} lookup(ed).E \text{ and } B \stackrel{def}{=} \overline{lookup}\langle ed\rangle.B\ .$$

The propagation of knowledge about participants can be found in the formalization. We have mentioned that this propagation either takes place (i) through the receipt of a message or (ii) through passing on participant references. (i) can be found where MS sends c as attachment to the visa application. MS therefore passes a callback channel to IO for the confirmation. Hence, this is an example for indirectly representing participant references through message link instances. (ii) can be also found where MS sends the application to IO: na is the channel where the approval has to be sent to, again indirectly representing the participant reference for the employer. This formalizes the attribute `participantRefs` of the message link `visaApplicationLink` set to `e`. In both cases we see that the propagation of knowledge about participants corresponds to the notion of scope extrusion in π-calculus.

We can summarize that the information given in the PBDs is mainly encoded in control flow constructs in π-calculus, i.e. choice, parallelism and sequence. [13] shows how more complex control flow constructs are represented in π-calculus. The information given in the participant topology specifies the π-names used and defines which names have to be passed in interactions between the different π-processes. In addition to the attribute `participantRefs` indicating link passing mobility, message links can also have the attribute `copyParticipantRefsTo` set. As an effect the bound name in the receiving π-process is simply renamed.

All BPEL4Chor constructs can be translated to BPEL constructs. E.g. `participantRefs` indicates that a `copy from partnerLink` action takes place prior to a send activity and a `copy to partnerLink` after a receive activity. In the case of `copyParticipantRefsTo` set, the target partnerLink at the receiving side has a different name than the source partnerLink on the sending side.

5 Conclusion

This paper has investigated the link passing mobility capabilities of BPEL4Chor. For illustrating this, a sample choreography was partially given in BPEL4Chor and formally given in π-calculus. It was briefly discussed how constructs from BPEL4Chor map to those from π-calculus. As link passing mobility plays an

essential role in choreographies, any useful formalization of BPEL4Chor has to include this concept. Therefore, a complete mapping from BPEL4Chor to π-calculus is desirable. Since BPEL4Chor is heavily based on BPEL and since there is no complete π-formalization of BPEL so far, such a complete mapping goes beyond the scope of this paper and is therefore left to future work.

References

1. A. Barros, M. Dumas, and A. ter Hofstede. Service interaction patterns. In *Business Process Management, volume 3649 of LNCS*, Nancy, France, September 2005. Springer.
2. BPMI.org. *Business Process Modeling Language*, 2002.
3. A. Brogi and R. Popescu. From BPEL Processes to YAWL Workflows. In *Web Services and Formal Methods, volume 4184 of LNCS*, pages 107–122, Berlin, 2006. Springer Verlag.
4. 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.
5. G. Decker, O. Kopp, F. Leymann, and M. Weske. BPEL4Chor: Extending BPEL for Modeling Choreographies. *ICWS 2007*.
6. G. Decker, H. Overdick, and J. M. Zaha. On the Suitability of WS-CDL for Choregraphy Modeling. In *EMISA 2006, volume P-95 of LNI*, pages 7–19, Bonn, 2006. Gesellschaft für Informatik.
7. G. Decker, F. Puhlmann, and M. Weske. Formalizing Service Interactions. In *Business Process Management, volume 4102 of LNCS*, pages 414–419, Berlin, 2006. Springer Verlag.
8. S. Hinz, K. Schmidt, and C. Stahl. Transforming BPEL to Petri nets. In *Business Process Management, volume 3649 of LNCS*, pages 220–235, Berlin, 2005. Springer Verlag.
9. R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes, Part I/II. *Information and Computation*, 100:1–77, September 1992.
10. Web Services Business Process Execution Language Version 2.0 – OASIS Standard. Technical report, Organization for the Advancement of Structured Information Standards (OASIS), Mar 2007.
11. H. Overdick, F. Puhlmann, and M. Weske. Towards a Formal Model for Agile Service Discovery and Integration. In *Proceedings of the International Workshop on Dynamic Web Processes (DWP 2005)*, IBM technical report RC23822, Amsterdam, December 2005.
12. F. Puhlmann. On the Suitability of the Pi-Calculus for Business Process Management. In *Technologies for Business Information Systems*, pages 51–62. Springer Verlag, 2007.
13. F. Puhlmann and M. Weske. Using the π-Calculus for Formalizing Workflow Patterns. In *Business Process Management, volume 3649 of LNCS*, Nancy, France, September 2005. Springer.
14. UN/CEFACT and OASIS. ebXML Business Process Specification Schema (Version 1.01). `http://www.ebxml.org/specs/ebBPSS.pdf`, 2001.