Modeling Choreographies: BPMN 2.0 versus BPEL-based Approaches

Oliver Kopp, Frank Leymann, Sebastian Wagner
Institute of Architecture of Application Systems
University of Stuttgart, Universitätsstraße 38, 70569 Stuttgart, Germany
firstname.lastname@iaas.uni-stuttgart.de

Abstract: Choreographies capture the collaboration aspects between two or more processes. Explicit choreography notations have been included in the upcoming version 2.0 of the Business Process Model and Notation language (BPMN 2.0). This paper presents a first evaluation of the choreography modeling capabilities of BPMN 2.0 and presents a summary of the evaluation of BPEL-based approaches. The result is that BPMN 2.0 does not support reference passing and is tightly tied to technical configurations.

1 Introduction

The collaboration between two or more processes is captured by process choreographies [Pel03]. There are two paradigms of choreography languages available: interconnection models and interaction models [DKB08]. In both modeling styles, the participants interact with each other and activities are connected. On the one hand, an interconnection model connects communication activities belonging to two participants. Thus, each send/receive message exchange is expressed using a *connection* between participants. On the other hand, interaction models express each send/receive message exchange as atomic *interaction*. Generally said, the terms "interaction model" and "interconnection model" are derived from the way each modeling paradigm expresses a send/receive message exchange and not from the fact that activities are connected or general interactions between processes are presented. The current version of BPMN 2.0 [Obj10] supports both, interconnection models and interaction models.

We use the requirements by Decker et al. [DKLW09] to evaluate the choreography capabilities of BPMN 2.0 and provide a comparison with BPEL-based choreography approaches. In this work, we dropped "R10: Integration with service orchestration languages". BPMN 2.0 also offers modeling orchestration of Web services and thus a discussion of mapping concepts from BPMN 2.0 to BPEL is obsolete. In fact, it is possible that a workflow engine accepts both formats [Ley10].

A variant of the requirements by Decker et al. [DKLW09] is presented by Schönberger [Sch11]. In principle, "Decomposability" and "Standardization" have been added, whereas "Correlation" and "Message Formats" have been dropped. We did not adopt that requirements framework to keep the existing evaluations of BPEL^{light}, WSFL, WS-CDL, Let's Dance, BPMN 1.x, iBPMN, BPSS/UMM, and SCA comparable with our work. Our future

225

work is to consolidate these requirements with the work of Schönberger and re-evaluate all choreography languages using the consolidated requirements. Neither the requirements by Decker et al. nor by Schönberger take comprehensibility of the language itself into account. Such a quality assessment might be done by using the quality framework presented by Krogstie et al. [KSJ06].

2 Chorographies in BPMN 2.0

In BPMN 2.0 **interconnection models** are implemented by *collaboration* diagrams. These diagrams consist of two or more pools, where each of them represents a participant in the overall collaboration. Within a pool, the internal process behavior of a participant is modeled with BPMN common elements (e. g., activities, events, and sequence flows). Alternatively, pools can be depicted as black boxes, i. e., no internal process behavior is modeled. In the discussion below we focus on pools that contain internal processes as they provide more information about the collaboration.

A choreography language has to support multi-lateral interactions (requirement *R1*). Multi-lateral interactions are supported by the pools and message flows in BPMN collaboration diagrams.

It must be possible to specify the participant topology, i. e. the number and types of participants involved in a choreography (*R2*). The participant types can be expressed by pools and by using attributes, the minimum and maximum number of participants can be specified. Hence, *R2* is supported as well.

The choreography language has to provide also means to model an arbitrary number of participants of the same type ("Service sets", *R3*), where the concrete number is determined at runtime. Multiplicity can by indicated in BPMN collaborations by three black parallel lines in the bottom center of the pool, thus *R3* is supported.

Another important choreography feature is "reference passing" (*R4*), where participant *A* enables participant *C* to communicate with participant *B* by passing the reference of *B* to *C*. This is not explicitly covered by the BPMN specification. The only way to indicate that messages contain references are documentation elements.

Message formats (*R5*) cannot be expressed by the graphical notation of BPMN, i. e., there exist no graphical elements in the specification to define the format of a message. A message, however, may have an optional `itemRef` property (which is invisible in the diagrams). This property represents an *item definition*, which contains a reference to a concrete data schema (e. g., XML schema), thus *R5* is fully met.

To enable an easy exchange of choreography implementations (as concrete WSDL portTypes and WSDL operations [W3C01]) the technical configurations of a choreography have to be interchangeable and should be separated from choreography model (*R6*). In BPMN collaborations, participants can be associated to interfaces (which are no graphical elements). These abstract interfaces may contain references that point to a corresponding technical implementation, such as WSDL `portTypes`. This, in turn, leads to the fact, that a reference has to be changed if the name of the `portType` has been changed, thus

requirement *R6* is not supported.

The modeling of time constraints (*R7*) has to be supported by choreographies, too. In BPMN collaborations they can be modeled by timer intermediate events.

Another choreography requirement is "Exception handling" *R8*. This is also supported by BPMN collaborations. An exception can be caught and raised by error events.

As choreographies may be enacted multiple times messages have to be routed to the correct process instance that belongs to the choreography instance, thus a message correlation mechanism is required (*R9*). In BPMN correlation information cannot be specified graphically in a diagram, but BPMN collaborations provide means to define correlation information for the collaboration (i. e., *R9* is met). For each collaboration a property named `correlationKey` can be defined. This property consists of one or many correlation properties that provide expressions to extract certain data from the payload of a message in order to associate the message to a particular process instance.

Interaction models are realized by *choreography* diagrams in BPMN 2.0. These diagrams define a special kind of processes that reflect the interaction sequence, i. e. the message exchanges between two or more participants (which are represented by pools in the collaboration diagrams). The atomic building block in choreography diagrams is the *choreography task* (briefly called "task" in the following). "[An atomic task] represents an Interaction, which is one or two Message exchanges between two Participants." [Obj10].

Requirement *R1* is satisfied, because the purpose of choreography diagrams is the definition of interaction behavior between two or more participants. If one participant has to interact with multiple partners, a task for each interaction can be created.

Choreography diagrams have full support of requirement *R2*: One can specify the type of a service involved in a choreography diagram. It is not possible, however, to determine the exact number of participant instances involved in an interaction. Using the `participantMultiplicity` attribute, it is possible to specify minimum and maximum number of participants.

Three black parallel lines in the bottom center of the participant bands indicate that multiple instances of a particular participant interact with another participant. Thus, requirement *R3* is satisfied.

There is no explicit support for reference passing mechanisms (*R4*) in choreography diagrams. As explained in the context of interconnection models, the only way to indicate that a message carries service references are natural language descriptions in documentation elements, which lack formality.

Choreography diagrams support message formats (*R5*). Although messages cannot be graphically visualized in the choreography diagram, each atomic task contains one or more messages. As mentioned in the context of interconnection models, a message may have an optional `itemRef` property that contains a reference to a concrete data schema.

Participants in collaborations can be associated to interfaces. This is also true for choreographies but as with collaborations an interface references to a `portType` (or any other technical implementation) by using its qualified name. Consequently, the reference has to be changed after the `portType` name has been changed. Thus, requirement *R6* is violated

as the technical configuration cannot be interchanged without adapting the choreography accordingly.

Requirement *R7* (time constraints) is satisfied by BPMN choreography diagrams as timer intermediate events can be added to the choreography.

A requirement that is not satisfied by BPMN choreographies is exception handling (*R8*): Exceptions are only visible to the participant where the error occurred. That means that the other participants are not aware of any exception that was thrown within a particular participant. As a consequence, they cannot handle this exception. The only way for a participant to inform the other participants about an exception are ordinary messages, i. e., there are no special error message types.

Like in BPMN collaboration diagrams, correlation information (requirement *R9*) cannot be depicted in a choreography diagram either. BPMN 2.0 choreographies, however, provide a property named `correlationKey` to define correlation information, thus *R9* is met.

3 BPEL-based Choreography Approaches

BPEL supports modeling of single business processes. It does not directly support choreographies. Hence, BPEL4Chor [DKLW09] has been introduced to support interconnection models and BPEL^{gold} [KEvL$^+$11] has been introduced to support interaction models.

BPEL4Chor directly supports **interconnection models**. The behavior of each participant is modeled using BPEL. Each behavior description is called "participant behavior description". Communication activities must not have WSDL-specific attributes such as `partnerLink` and `operation`. The interconnection to other participants is established by "message links", which connect sending and receiving activities. A message link contains the attribute `participantRefs`, which lists the participant references transmitted on the message flowing on the message link. Message links themselves are listed in a "topology". Besides message links, the topology contains a list of participant types (referring to participant behavior descriptions) and a list of participants and participant sets which denote the participants taking part in one instance of the choreography. Decker et al. [DKLW09] already evaluated BPEL4Chor using the requirements. We summarize the findings in the following.

BPEL4Chor introduces a topology (*R*2) with sets of services (*R*3). BPEL4Chor allows an arbitrary number of participant types and thus supports multi-lateral interactions (*R*1). BPEL4Chor supports participant references by the attribute `participantRefs` on a message link (*R*4). Specification of message formats is fully supported by BPEL4Chor (*R*5). Technical configurations are stored in a separate grounding document, which enables interchangeability of technical WSDL information in different settings (*R*6). Time constraints may be specified using BPEL's control flow constructs (*R*7). BPEL's exception handling can be reused in BPEL4Chor (*R*8). Correlation may be specified on a name-basis or directly using message properties (*R*9).

BPEL^{gold} is a choreography language providing **interaction models** using the control flow semantics of BPEL. "gold" stands for *glo*ba*l* *d*efinition. The complete language and an

228

evaluation of the requirements is presented by Kopp et al. [KEvL+11]. Here, we provide a summary of the language and the evaluation according to the requirements.

BPEL^{gold} reuses the topology and the grounding of BPEL4Chor. The participant behavior descriptions are replaced by a single *process interaction description*. A message link of the topology relates a sending participant to a receiving participant and excludes concrete activities. The attribute `participantRefs` still exists.

The process interaction description is a BPEL process following the *Abstract Process Profile for Basic Interaction Models*. "Basic" denotes that only interactions between listed participants are allowed. BPEL^{gold} introduces the *Abstract Process Profile for Extended Interaction Models*, where a global observer may also interact with the choreography. This is used for runtime compliance checking of choreographies as described by Kopp et al. [KvLN08]. This profile is out of scope of this paper. In the case of the basic profile, BPEL's communication activities are disallowed in BPEL^{gold}. Interactions are solely described by an `interaction` activity. This activity presents an atomic message exchange between two participants: One participant sends a message and another participant receives that message.

As BPEL^{gold} builds on BPEL4Chor, it supports a topology (*R*2) with sets of services (*R*3). BPEL^{gold} also supports arbitrary numbers of participant types and thus multi-lateral interactions (*R*1). The attribute `participantRefs` is still present in message links and thus participant references can be passed over a message link (*R*4). Message formats can be specified using variables in the process model (*R*5). The concept of a grounding is still present (*R*6): In a grounding, an interaction activity is assigned a `partnerLink`/`operation` combination, which in turn is used in the services implementing the described participants. Time constraints can be modeled using the constructs provided by BPEL (*R*7). BPEL^{gold} supports the exception handling mechanisms of BPEL. Thus the exceptional path through the choreography model can be explicitly described (*R*8). BPEL's concept of correlation is reused in BPEL^{gold}: Each `interactionActivity` contains a `correlationSet` element referring to a correlation set used for correlation (*R*9). BPEL^{gold} can be mapped to BPEL4Chor: each `interactionActivity` is split into an `invoke`/`receive` pair. The name of each communication activity is generated. Thus, all information of a BPEL4Chor message link is available.

4 Conclusion and Outlook

We compared the choreography modeling capabilities of BPMN 2.0 using the evaluation criteria of Decker et al. [DKLW09]. For each requirement, we showed the constructs offered by BPMN 2.0 to implement each requirement. Section 3 summarized the evaluation of BPEL-based approaches. Table 1 summarizes the evaluation. On the one hand, it turns out, that BPEL4Chor and BPEL^{gold} provide a complete support of the requirements, whereas BPMN 2.0 lacks support: Reference passing and interchangeability of technical configurations are not fully supported. On the other hand, only BPMN offers a full graphical notation for both interaction models and interconnection models. Thus, a next step is a detailed investigation of the integration of BPMN 2.0 with BPEL4Chor and BPEL^{gold}.

	BPMN 2.0 Collaboration	BPMN 2.0 Choreography	BPEL4Chor	BPEL^{gold}
Modeling Style	interconnection	interaction	interconnection	interaction
R1. Multi-lateral interactions	+	+	+	+
R2. Service topology	+	+	+	+
R3. Service sets	+	+	+	+
R4. Reference passing	–	–	+	+
R5. Message formats	+	+	+	+
R6. Interchangeability of technical configurations	–	–	+	+
R7. Time constraints	+	+	+	+
R8. Exception handling	+	–	+	+
R9. Correlation	+	+	+	+

Table 1: Assessment of BPMN 2.0 and BPEL-based approaches

This may be an extension of BPMN to support all constructs of BPEL4Chor similar to the approach taken by Pfitzner et al. [PDKL07] in the case of BPMN 1.0 and BPEL4Chor.

Currently, there exists a mapping of BPMN 1.0 to BPEL4Chor [PDKL07] and no mapping of BPMN 2.0 to BPEL4Chor. In case such a mapping is available, BPMN 2.0 collaborations can be transformed to BPEL4Chor choreographies. Such a transformation would lead to a fulfillment of requirement R6: The technical configuration is not done in the BPMN 2.0 model, but using the mechanisms offered by BPEL4Chor.

References

[DKB08] G. Decker, O. Kopp, and A. Barros. An Introduction to Service Choreographies. *it - Information Technology, Special Issue on Service-oriented Architectures*, 50(2), 2008.

[DKLW09] G. Decker, O. Kopp, F. Leymann, and M. Weske. Interacting services: from specification to execution. *Data & Knowledge Engineering*, 68(10):946–972, April 2009.

[KEvL+11] O. Kopp, L. Engler, T. van Lessen, F. Leymann, and J. Nitzsche. Interaction Choreography Models in BPEL: Choreographies on the Enterprise Service Bus. In *S-BPM ONE 2010*, CCIS. Springer, 2011.

[KSJ06] J. Krogstie, G. Sindre, and H. Jørgensen. Process models representing knowledge for action: a revised quality framework. *European Journal of Information Systems*, 15(1):91–102, 2006.

[KvLN08] O. Kopp, T. van Lessen, and J. Nitzsche. The Need for a Choreography-aware Service Bus. In *YR-SOC*, 2008.

[Ley10] F. Leymann. BPEL vs. BPMN 2.0: Should You Care? In *2nd International Workshop on BPMN*, LNBIP. Springer, 2010.

[Obj10] Object Management Group. *Business Process Model and Notation (BPMN) Version 2.0*. OMG, 2010.

[PDKL07] K. Pfitzner, G. Decker, O. Kopp, and F. Leymann. Web Service Choreography Configurations for BPMN. In *WESOA*, volume 4907 of *LNCS*. Springer, 2007.

[Pel03] C. Peltz. Web Services Orchestration and Choreography. *IEEE Computer*, 36(10):46–52, 2003.

[Sch11] A. Schönberger. Do We Need a Refined Choreography Notion? In *ZEUS*. CEUR-WS.org, 2011.

[W3C01] W3C. *Web Services Description Language (WSDL) 1.1*, March 2001.