BPEL for Semantic Web Services (BPEL4SWS)

Jörg Nitzsche, Tammo van Lessen, Dimka Karastoyanova, and Frank Leymann
Institute of Architecture of Application Systems
University of Stuttgart,
Universitätsstraße 38, 70569 Stuttgart, Germany
http://www.iaas.uni-stuttgart.de

in: On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops.
See also BIBTEX entry below.

BIBTEX:

```
@inproceedings{NitzscheLKL07,
  author    = {J{\"o}rg Nitzsche and
               Tammo van Lessen and
               Dimka Karastoyanova and
               Frank Leymann},
  title     = {BPEL for Semantic Web Services (BPEL4SWS)},
  booktitle = {On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops},
  year      = {2007},
  pages     = {179-188},
  ee        = {http://dx.doi.org/10.1007/978-3-540-76888-3_37},
  publisher = {Springer}
}
```

BPEL for Semantic Web Services (BPEL4SWS)

Jörg Nitzsche, Tammo van Lessen, Dimka Karastoyanova, and Frank Leymann

Institute of Architecture of Application Systems
University of Stuttgart
Universitaetsstrasse 38, 70569 Stuttgart, Germany
{joerg.nitzsche,tammo.van.lessen,dimka.karastoyanova,
frank.leymann}@iaas.uni-stuttgart.de
http://www.iaas.uni-stuttgart.de

Abstract. In this paper we present BPEL for Semantic Web Services (BPEL4SWS) - a language that facilitates the orchestration of Semantic Web Services using a process based approach. It is based on the idea of WSDL-less BPEL and enables describing activity implementations semantically which increases the flexibility of business processes. Following an approach that uses a set of composable standards and specifications, BPEL4SWS is independent of any Semantic Web Service framework. It can be used to compose Semantic Web Services, traditional Web Services and a mix of them.

1 Introduction

Web Service (WS) [1] technology is one implementation of a service oriented architecture (SOA) [2,3]. It aims at integrating applications and has gained broad acceptance in research and industry. Service composition is currently enabled mainly by a process-based approach [4] embodied by the de facto standard BPEL (Business Process Execution Language) [5]. BPEL is using WSDL [6] descriptions to identify partner services, i.e. services are identified by port types and operations in the process models. As a result only services that implement a concrete interface can be used which is a major deficiency of BPEL.

One approach to addressing the rigidity of WSs has evolved from the Semantic Web - the Semantic Web Service (SWS) technology. The most prominent SWS frameworks are the Web Ontology Language for Services (OWL-S) [7] and the Web Service Modeling Ontology (WSMO) [8]. SWS technology introduces an additional level of abstraction and can be considered as an integration layer on top of Web Services. Instead of a syntactic description of a WS a declarative description of the service functionality is given.

In this paper we present BPEL4SWS, a language which enables describing activity implementations in a machine processable manner using Semantic Web technologies, as an alternative to specifying WS interfaces, i.e. it enables the use of Semantic Web Services as well as traditional Web Services. The BPEL4SWS framework exhibits and maintains the composability characteristics of the WS technology. In this way, BPEL4SWS processes are able to use both semantic

WSs and conventional WSs intermixed within a single process, and independent of any SWS framework. Additionally, BPEL4SWS processes can be exposed both as Semantic Web Services and conventional Web Services. To enable this, BPEL4SWS provides a grounding mechanism that contributes to maintaining BPEL4SWS processes compliant to standard WS-based communication.

The paper is organized as follows. In section 2 the necessary background information about BPEL is provided. BPEL4SWS is presented in section 3. We focus on the main aspects of BPEL4SWS and explain how they make up the functionality the language provides. These aspects are: (i) a WSDL-less interaction model (BPEL$^{\text{light}}$) for describing the process logic, (ii) the annotation of SWS descriptions (such as WSMO and OWL-S) to support semantic discovery, (iii) the usage of WS-* technology for the invocation of services and (iv) the usage of SA-WSDL [9] to provide a seamless mapping between XML data and ontological data. In section 4, an engine-prototype that implements BPEL4SWS is presented. Finally related work is examined in section 5 and a conclusion is given in section 6.

2 BPEL

BPEL is the de facto standard for specifying business processes in a WS world. It enables both, the composition of WSs [1] and rendering the composition itself as WSs. Thus, BPEL provides a recursive aggregation model for WSs. The composition of WSs can be specified as a flow between WS operations. Therefore BPEL provides several so called *structured activities* that prescribe the control flow between the *interaction activities* that model interactions with other WSs. BPEL does not support explicit data flow; instead, data is stored in shared variables that are referenced and accessed by interaction activities and data manipulation activities (e.g. `<assign>` activity). The control flow between activities can be structured either in a block-based manner by nesting structured activities like `<sequence>` (for sequential control flow), `<flow>` (for parallel control flow) and `<if>` (for conditional branches in the control flow) activities, or graph-based by defining `<links>` (i.e. directed edges) between activities in a `<flow>` activity; both styles can be used intermixed.

In order to enable communication that is compliant to the Basic profile [10] of the WS-Interoperability Organization (WS-I)[1], i.e. without using WSDL operations of type *notification* and *solicit-response*, BPEL introduces the concept of a partner link type which is defined as an extension to WSDL. A partner link type binds two port types, namely a port type the process offers to a partner and a port type the process requires from the corresponding partner. This way it defines a channel between two abstract business partners (roles) through which the partners exchange messages; the roles correspond to port types. If a process interacts synchronously with a partner, such a channel is just unidirectional, i.e. the corresponding partner link type contains a single role.

[1] http://www.ws-i.org/

In order to establish a contract (i.e. an agreement between two partners which message channel to use), BPEL's partner links reference a partner link type and specify which role is taken by the process itself (`myRole`) and which role is taken by the partner (`partnerRole`).

The interaction activities [1] (`<receive>`, `<reply>`, `<invoke>`, `<pick>`) and event handlers are used to define the actual message exchange corresponding to a partner link, i.e. data transmitted and style of communication (synchronous vs. asynchronous). For that purpose, interaction activities reference a partner link and a WSDL operation. Receiving activities (i.e. `<receive>` and `<pick>`), the `<reply>` activity and the event handler reference an operation of the process's port type, whereas the `<invoke>` activity references an operation of the partner's port type. Note that a synchronous invocation of a process is specified via a receiving activity and a matching `<reply>` activity.

3 BPEL4SWS

As shown in the previous section BPEL makes use only of WSs to enable service composition. Partner interfaces are described using WSDL; they are hard-coded within the process logic. As a result, only services that implement the WSDL interface used in the BPEL definition can be used as activity implementations and services that provide the same functionality but implement other interfaces cannot be used. This hampers integration of functionally equal services.

Semantic Web Services describe services not in terms of an interface but rather describe their functionality and capability semantically and in a machine processable manner. For that reason Semantic Web Services increase the level of integration and can be considered an integration layer on top of Web Services where services are discovered based on their functionality and not based on their signature.

To enable the usage of Semantic Web Services technology within business processes there is a need for a process language that does not specify partner services using their WSDL description, but rather allows using higher level semantic descriptions. BPEL$^{\text{light}}$ [11] decouples process logic and interface defintion (but still is applicable in a WS-* environment) and therefore makes for a good candidate as a basis for a process execution language for Semantic Web Services.

Indeed, BPEL4SWS uses BPEL$^{\text{light}}$ as basis and allows to attach SWS descriptions to BPEL$^{\text{light}}$ such that SWS frameworks like OWL-S and WSMO and corresponding implementations can be used to discover and select SWS that implement the functionality required for an activity. In addition both, the SWS description and the process itself are partly grounded to WSDL to facilitate WS-* based communication (see section 3.5). Current SWS frameworks use ontologies as data model to facilitate semantic discovery. For that reason, SAWSDL is used to enable a seamless mapping of data between its XML representation and its ontological representation. This is also needed because in BPEL4SWS WSs and SWSs can be used intermixed.

3.1 BPEL$^{\text{light}}$

BPEL$^{\text{light}}$ extends BPEL 2.0 via additional elements in a separate namespace[2] which act as a replacement for the WSDL-based interaction model. It provides a WSDL independent interaction model and (re-)introduces the concept of a partner.

BPEL$^{\text{light}}$ defines a mechanism to describe the communication between two partners without any dependency on WSDL. Therefore it introduces the `<conversation>` element. This element plays the role of a WSDL-less partnerLink, facilitates grouping of interaction activities and thus enables defining a complex message exchange between two partners. Similarly to the `<partnerLink>`, which is defined in the `<partnerLinks>` block, every `<conversation>` is defined in a `<conversations>` element.

In addition, WSDL independent interaction activities are needed. Due to the fact that the `partnerLink` and `operation` attribute of interaction activities and event handlers defined in BPEL are mandatory, these activities have a WSDL dependency. Consequently, BPEL$^{\text{light}}$ introduces new interaction activities without the WSDL dependency using the `<extensionActivity>` mechanism. The new activity type introduced in BPEL$^{\text{light}}$ is presented in Listing 1. A set of interaction activities, that form a message exchange with a single partner, are grouped using the `<conversation>` element.

```
<extensionActivity>
   <bl:interactionActivity name="NCName"
                           inputVariable="NCName"?
                           outputVariable="NCName"?
                           mode="in-out|out-in"?
                           conversation="NCName"
                           createInstance="yes|no"?
                           standard-attributes>
      standard-elements
   </bl:interactionActivity>
</extensionActivity>
```

Listing 1. BPEL$^{\text{light}}$'s `<interactionActivity>`

The `interactionActivity` can be configured such that it behaves like any of the basic interaction activities BPEL defines (`receive`, `reply` and `invoke`). Additionally, BPEL$^{\text{light}}$ defines WSDL independent `pick` and `eventHandler` elements. This is further described in [11]. This way BPEL$^{\text{light}}$ enables modelling arbitrary message exchange patterns or service interaction patterns [12].

BPEL 1.1 has the notion of a partner that comprises multiple partner links. However, the `<partner>` element has been removed in BPEL 2.0 because grouping partnerLinks is considered a deployment issue and the partner is not evaluated during runtime but rather is only used for documentation purpose. BPEL$^{\text{light}}$ reintroduces the notion of a partner. A partner in BPEL$^{\text{light}}$ comprises multiple conversations, and thus expresses that multiple conversations have to take place with one and the same business entity. Additionally, BPEL$^{\text{light}}$ enables

[2] `xmlns:bl=http://iaas.uni-stuttgart.de/2007/BPELlight`

naming the partner and thus identifying a concrete organisation. The syntax of the new <partner> element is shown in listing 2. Within a process multiple partners can be specified.

```
<bl:partners>
  <bl:partner name="NCName"
              businessEntity="QName">+
     <bl:conversation name="NCName"/>+
  </bl:partner>
</bl:partners>
```

Listing 2. The <partner> element.

BPEL^{light} includes an extension of the <assign> activity that enables copying a partner identification into the <partner> element. Therefore the <to> specification is extended with a partner attribute that defines to which partner definition the concrete partner instance information (business entity) is copied. This is similar to copying an endpoint reference to a partner link in conventional BPEL. A partner can only be set if its corresponding conversations have not been established yet.

3.2 Attachment of SWS Descriptions

According to the composable approach we take for BPEL4SWS we do not encode the semantic descriptions in the BPEL4SWS process model. Instead WS-PolicyAttachment [13] is used to add semantic annotations. In general, annotations can be attached anywhere, i.e. on the activity level as well as on the conversation level. We advocate attaching the SWS descriptions to conversations. The meaning of the semantic annotations on the conversation level is described in the following sections. BPEL4SWS differentiates between two different types of conversations, 'providing' and 'consuming'. A 'providing' conversation is a conversation with a partner, where the partner uses a service the process provides via the conversation, a 'consuming' conversation is a conversation with a partner, where the process uses a service, the partner service provides.

3.3 Using OWL-S

OWL-S (Web Ontology Language for Services) [7] was the first approach towards describing services semantically. It uses ontologies as data model and describes a service in terms of Service Profile, Service Model and Service Grounding. The capabilities of a service in terms of 'inputs', 'outputs', 'preconditions' and 'effects' are described in the Service Profile. The service model describes in which order messages have to be exchanged to consume the service's functionality and the Service Grounding defines which WSDL operations of a concrete service have to be used to exchange these messages.

OWL-S describes the execution of a Web service as a collection of remote procedure calls. We argue that this is only correct for a small percentage of the cases in business processes [14] since typically the communication is asynchronous.

OWL-S describes a self-contained service and has no notion of two partners (requester and provider) that provide means to invoke each other. It is designed to ground to all four kinds of WSDL (1.1) operations: one-way, request-response, notification and solicit-response. However, due to the WS-I Basic Profile [10] the WSDL operations of type notification and solicit-response must not be used. The lack of a partner model is the major deficiency of OWL-S since WS-* based and WS-I compliant asynchronous communication is not considered.

Nevertheless, OWL-S can be used in the context of BPEL4SWS, but only for the cases that use synchronous communication only: an OWL-S service is attached to a conversation and the OWL-S service model describes the sequence of BPEL4SWS interaction activities associated with the conversation. An OWL-S description attached to a 'providing' conversation is grounded to the WSDL interface that describes the process, an OWL-S description attached to 'consuming' conversations is not grounded, because the WSDL interface implemented by the partner service is assumed to be unknown during design time. This way dynamic service discovery independent of WSDL port types is enabled.

3.4 Using WSMO

Compared to OWL-S, WSMO (Web Service Modeling Ontology) is the more promising approach because its conceptual model enables standards (WS-*) based asynchronous communication. WSMO distinguishes between the description of a service (WSMO Web Service) and the description of requirements a client has on a service (WSMO Goal). Both descriptions are based on ontologies and contain a functional description that semantically describes what a service provides or a client requests in terms of 'preconditions', 'assumptions', postconditions' and 'effects' and an interface description, the so called WSMO choreography [15]. Thus WSMO enables expressing what kind of functionality a service provides, which message exchange is needed to consume its functionality as well as what a client aims to achieve and which message exchange the client will have. This message exchange can be grounded to WSDL operations (of type request-response and one-way) on every side of an interaction in such a way that both synchronous and asynchronous communication is enabled in a standards (WS-*) based and interoperable manner (i.e. WS-I compliant).

In order to use the WSMO framework for BPEL4SWS, WSMO goals are attached to 'consuming' conversations and Semantic Web Service descriptions are attached to 'providing' conversations. The choreography of goal and Web service describe the sequence of the BPEL4SWS interaction activities associated with the conversations they are attached to. This way the process can be discovered by WSMO implementations using its WSMO Web Services description and activity implementations of BPEL4SWS can be discovered by submitting the attached WSMO goal to a WSMO enabled middleware. Similarly to OWL-S, the discovery of both, the process itself and its activity implementations is independent of WSDL port types.

Since WSMO is more suitable for business processes due to its support for asynchronous communication we focus on using the WSMO framework for discovery of Semantic Web Services.

3.5 Grounding to WSDL

BPEL4SWS uses (semantic) Web Services as activity implementations and is exposed as Web Services as well as Semantic Web Services. Thus, the BPEL$^{\text{light}}$ interaction model is partly grounded to WSDL. To preserve the decoupling of process logic and activity implementation definition this is done within an artefact called 'grounding file' and not within the BPEL$^{\text{light}}$ description.

The grounding for the semantic 'consuming' conversation only specifies which WSDL operations are provided by the receiving activities. This is illustrated in Listing 3. This way, an engine implementation can resolve an incoming message to a certain activity within the process model. An `invoke`-like `<interactionActivity>` simply sends (and receives) a message to (and from) a SWS based middleware.

```
<grounding process="QName">

   <activity name="NCName"
             portType="QName"
             operation="NCName"/>*

</grounding>
```

Listing 3. partial grounding

```
<grounding process="QName">
   <conversation name="NCName"
                 partnerLinkType="QName"
                 myRole="NCName"
                 partnerRole="NCName"/>*
   <activity name="NCName"
                 operation="NCName"/>*
</grounding>
```

Listing 4. full grounding

For conventional 'consuming' and for 'providing' conversations in general the grounding is more complex (see Listing 4). In this case the conversation is grounded to a partner link type which is required to support WSDL based asynchronous communication. Therefore, it has to be specified which role of the partner link type the partner service and the process itself take. In addition to the grounding of the conversation to the partner link type, all interaction activities, including the invoking ones have to be grounded to WSDL operations.

Using this 'full' grounding for 'consuming' conversations means that conventional Web Services are used. In this case a semantic description must not be attached to the conversation. Whenever a sending activity is performed, the engine implementation looks up the grounding file for the operation it has to invoke and whenever a message is received it can be dispatched to an activity in the process model using the information given in the grounding file.

By specifying also the partner role for a 'providing' conversation, a process that is exposed as a Semantic Web Service can also be consumed like a conventional WS. The grounding file specifies how incoming messages are resolved to activities and which operations have to be used by sending activities.

In case the process is discovered using its SWS description and asynchronous communication is used, the process does not call back (invoke) the partner directly using the operation specified in the grounding file but rather sends (and receives) a message to (and from) a SWS based middleware.

Enabling exposing a process as both, WSDL service and SWS is of utmost importance because if the process would only be described semantically, i.e. exposed as a SWS, and conventional invocation would not be supported, most of the clients (not supporting Semantic Web Service technology) would not be able to use its functionality. In this case, building a semantic business process and therefore a Semantic Web Service would not increase but rather reduce the number of clients, which is a knockout criterion.

3.6 Dualism of data representation using SAWSDL

Existing Semantic Web Service frameworks are using ontologies as data model to facilitate semantic service discovery and their grounding defines which communication infrastructure is used to invoke a service. Since a BPEL4SWS process is exposed as a conventional WS for backwards compatibility, its semantic description is grounded to WSDL. Hence, there is a need to transform between the ontological and XML representation of data. SAWSDL [9] provides semantically annotated data types as means to describe the so called lifting and lowering of data. It introduces the concepts of modelReference, liftingSchema and loweringSchema. The modelReference identifies the concept to which the XML data can be lifted, and the liftingSchema defines how the lifting can be done. The loweringSchema can be used to lower the data again from an ontological level to its XML representation.

4 Implementation

To demonstrate the capabilities of BPEL4SWS a prototypical BPEL4SWS engine was implemented [16]. It is based on the open source Apache ODE engine[3] and currently supports invocation of WS as well as synchronous invocation of WSMO Web Services.

5 Related work

Mandell and McIlraith [17] identified the shortcomings of BPEL with respect to the flexibility of service and partner discovery. They presented a proxy-based approach where service requests are delegated to a discovery service through a locally bound WSDL interface, i.e. they mix different levels of abstractions in the process model: service und infrastructure service. They use OWL-S to semantically describe the activity implementations of a BPEL process. Whether the language is extended has not been presented. Asynchronous and stateful communication between services is not discussed.

Meteor-S [18] also takes a proxy-based approach where all interactions are bound to virtual partners, hosted by a process configuration module. The process configuration module delegates the service requests to concrete services either

[3] http://incubator.apache.org/ode

bound during deployment or during runtime. As the proxy is stateful, it enables creating an execution plan in case it is required to invoke several operations in order to achieve the specified goal. Asynchronous interaction between the process and the proxy or stateful interaction via multiple synchronous invocations between the proxy and the process is not discussed. Like in the previous approach, it is not known whether the language is extended and how the semantic annotation of the interaction activities is done, i.e. whether a single `<invoke>` activity or a complete partner link is described semantically.

In contrast to the already mentioned approaches Karastoyanova et al. [19] present an extension of the language, namely an extension to the `<invoke>` activity. Their approach also uses OWL-S to describe activity implementations and only allows for synchronous invocation of OWL-S services.

6 Conclusion

In this paper we presented BPEL4SWS, a flexible and comprehensive approach for composing Web Services and Semantic Web Services. By allowing for describing activity implementations semantically, i.e. using SWS concepts, BPEL4SWS enables application integration on a higher level of abstraction. The presented framework is composed of a set of specifications and is by design independent of any specific SWS technology. In contrast to other approaches the interfaces of the SWS based middleware are not hard-wired in the process model. Instead the BPEL language is extended to facilitate specifying activity implementations semantically. Interfacing the middleware is considered to be part of the configuration of the system. That is, BPEL4SWS clearly distinguishes between different levels of abstraction. BPEL4SWS provides support for asynchronous communication which is essential for business processes. It uses an XML–ontology dualism for representing data to support Semantic Web Service technology as well as Web Service technology.

Acknowledments

The work published in this article was partially funded by the SUPER project[4] under the EU 6th Framework Programme Information Society Technologies Objective (contract no. FP6-026850).

References

1. Weerawarana, S., Curbera, F., Leymann, F., Storey, T., Ferguson, D.: Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More. Prentice Hall PTR Upper Saddle River, NJ, USA (2005)
2. Burbeck, S.: The Tao of e-business services. IBM Corporation (2000)

[4] `http://www.ip-super.org/`

3. Krafzig, D., Banke, K., Slama, D.: Enterprise SOA: Service-Oriented Architecture Best Practices (The Coad Series). Prentice Hall PTR Upper Saddle River, NJ, USA (2004)
4. Leymann, F., Roller, D.: Production workflow. Prentice Hall (2000)
5. A. Alves et al.: Web Services Business Process Execution Language version 2.0. Committee specification, OASIS (January 2007)
6. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: Web Services Description Language (WSDL) 1.1 (2001)
7. Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., et al.: OWL-S: Semantic markup for web services. W3C Member Submission. World Wide Web Consortium (2004)
8. Lausen, H., Polleres, A., Roman, D.: Web Service Modeling Ontology (WSMO). W3C Member Submission (2005)
9. Farrell, J., Lausen, H.: Semantic Annotations for WSDL and XML Schema. W3C Recommendation (August 2007)
10. Ballinger, K., Ehnebuske, D., Ferris, C., Gudgin, M., Liu, C., Nottingham, M., Yendluri, P.: Basic Profile Version 1.1. WS-I Specification (2004)
11. Nitzsche, J., van Lessen, T., Karastoyanova, D., Leymann, F.: BPEL$^{\text{light}}$. In: 5th International Conference on Business Process Management (BPM), To appear. (September 2007) Brisbane, Australia.
12. Barros, A., Dumas, M., ter Hofstede, A.: Service interaction patterns: Towards a reference framework for service-based business process interconnection. Technical Report FIT-TR-2005-02, Faculty of Information Technology, Queensland University of Technology, Brisbane, Australia (March 2005)
13. Bajaj, S., Box, D., Chappell, D., Curbera, F., Daniels, G., Hallam-Baker, P., Hondo, M., Kaler, C., Malhotra, A., Maruyama, H., et al.: Web Services Policy Attachment (WS-PolicyAttachment). W3C Member Submission (April 2006)
14. Nitzsche, J., van Lessen, T., Karastoyanova, D., Leymann, F.: WSMO/X in the Context of Business Processes: Improvement Recommendations. International Journal of Web Information Systems, ISSN: 1744-0084 (2007)
15. Roman, D., Scicluna, J., Nitzsche, J.: D14 V 0.4: Ontology-based Choreography (2007)
16. van Lessen, T., Nitzsche, J., Dimitrov, M., Karastoyanova, D., Konstantinov, M., Cekov, L.: An Execution engine for BPEL4SWS. In: 2nd Workshop on Business Oriented Aspects concerning Semantics and Methodologies in Service-oriented Computing (SeMSoc) in conjunction with ICSOC, To appear. (September 2007) Vienna, Austria.
17. Mandell, D., McIlraith, S.: Adapting BPEL4WS for the Semantic Web: The Bottom-Up Approach to Web Service Interoperation. Proceedings of the Second International Semantic Web Conference (2003) 227–241
18. Verma, K., Gomadam, K., Sheth, A., Miller, J., Wu, Z.: The METEOR-S Approach for Configuring and Executing Dynamic Web Processes. LSDIS METEOR-S project 6–24
19. Karastoyanova, D., Leymann, F., Nitzsche, J., Wetzstein, B., Wutke, D.: Parameterized BPEL Processes: Concepts and Implementation. In: 4th International Conference on Business Process Management (BPM). (September 2006) Vienna, Austria.