Process Definition Capabilities and Web Services

By: Ivana Trickovic

Company: SAP AG

Date: 27 Dec 2005

Table of Contents

Introduction

Discussions about Web services typically start and end with considerations of basic formatting of messages, delivery options, static interfaces describing messages that the service can consume and/or produce, services metadata, transactional properties, reliability and security. Often, however, behind services there are processes, which could also be hard-coded, driving those services and making them long running entities. In that case services can offer alternative operations to service consumers and must comply with the underlying processes. But the order in which multiple alternative operations may be invoked is not always obvious to service consumers. Also, building new applications by composing existing Web services is a new aspect of a Service-Oriented Architecture (SOA). These two examples show that process definition capabilities play an important role in SOA. Process definition capabilities could be used to extend the static interface of Web services and add some behavioral aspects, such as ordering constraints over the messages the service may exchange. The capabilities could be used to build new applications or business processes out of existing Web services.

The Web Services Business Process Execution Language (WS-BPEL, BPEL4WS or BPEL for short) provides those missing process definition capabilities and it is considered as one of the key building blocks of SOA. Version 1.1 published in May 2003 has been taken as input for an OASIS standardization process. The first version was an important step towards consolidating the area of business process execution languages. The language includes many concepts known from its predecessors – WSFL [5] and XLANG [6]. Currently, the language is under an OASIS standardization process which shall be finalized in the first part of 2006. Even before finalization of the standardization activity, many implementations of version 1.1 are available as open source or in commercial products, making it de-facto standard.

This article discusses the most compelling usage scenarios and gives a short overview of the language features. It explains the language support in SAP NetWeaver and discusses the prospect of the future work.

Usage Scenarios

Services Orchestration. With the evolution of SOA and adoption of Web services the role of services orchestration became more important. The ability to compose processes out of services provides flexibility in creating and building new applications. The prerequisite for this is, however, alignment with basic principles of SOA. Encapsulation of the business logic into independent, autonomous services which offer interfaces as the only means to interact with the service is seen as the basic principle of SOA. This implies that it is necessary to separate the business logic from the presentation logic and the process flow logic. While business logic would be encapsulated within Web services, the process flow logic would be part of a new layer – services orchestration.

The model for services orchestration may follow traditional workflow models. It may use the same constructs for sequencing a set of Web services interactions which may happen in a certain order or in parallel. What is new is a mechanism used to associate messages sent by a service with a particular process instance. Also, the ability of handling faults in processes based on Web services and allowing a set of interactions to be undone play important role.

Observable behavior of Web Services. In order to properly interact with a service the consuming application, which could also be another process as well, must be aware of the contract the service offers. The contract contains messages the service may send and consume, different policies, indicating which conditions must be met and which assertions must be fulfilled. But in addition, the service shall expose its behavioral interface, indicating constrains on the order in which messages that the service can consume and send must be exchanged. This behavioral interface indicates that behind the service a process is running. The process may be implemented using different technologies or programming languages.

Such behavioral interface extends the static interface of Web services. WSDL [3] is used to describe the static interface of Web services, which includes definitions of individual operations only. This may be adequate for Web services participating in stateless message exchanges. For Web services, which participate in longer conversations it is necessary to describe the behavior of the services in terms of dependencies, either logical or temporal, among exchanged messages.

Overview of the Language

The BPEL language introduces the features needed to specify Web services-based business processes. It is an XML-based language which is built on top of WSDL [3], XPath [9], and XML Schema [10], [11].

The language uses both block-oriented and graph-oriented control flows. Block-oriented control flow could be understood as a special form of graph-oriented control flow. On one side it has less expressive power than graph-oriented control flow, but on the other side it is simpler and more effective approach for process engineers because it is less error-prone. The modeling constructs assure per definition that the process definition at the end is well formed. Graph-oriented control flow requires some verification capabilities to be supported by modeling tools. However, using only the block-oriented approach may be too restrictive. It was a design decision to allow both approaches and leave up to process engineers to use preferred one.

The language provides a set of primitive and structured activities. The set of primitive activities includes Web services interaction activities (consuming a message sent by an external service and sending a message to an external service by invoking a corresponding operation) and waiting for some time doing nothing. The set of structured activities includes activities known from other programming languages, such as ordered sequence of activities, iterations, conditional paths, parallel execution of multiple activities, and alternative exclusive paths. The language also allows handling messages that may be received asynchronously within a certain time frame.

The language provides a mechanism to identify the target service dynamically, based on values in incoming messages. The set of services participating in a process as partners can vary during the execution of the process itself. For example, a process can start interaction with one service which will introduce another service as a partner during the process execution. For this a mechanism is provided that allows handling of services' endpoint references.

The language provides a mechanism to partially "undo" a given set of activities. Web services are considered to be autonomous entities and their relationships with other Web services participating in the same business process are loose. This requires a different model for error handling and transactional behavior within the process context. The language provides a means to explicitly define which parts of the process must be managed in transactional way and which activities must be performed to partially undo the outcome of a set of activities.

The language provides a mechanism to identify the target process instance. A service can be engaged in multiple conversations in parallel. A mechanism is needed to deliver messages to the correct instance of the process implementing such service. The language introduces the concept of correlation which describes values that must be carried in messages to identify the correct process instance.

The language introduces data manipulation capabilities. Processes typically deal with process flow logic only, while application messages are generated by business logic encapsulated within Web services. However, process models must provide data manipulation capabilities in order to be able, for example, to aggregate a number of messages received from different source Web services and send the aggregated message to a target Web service. The language introduces some minimal data manipulation capabilities, and relies on other specifications which focus on the manipulation of XML documents for more advanced data manipulation features.

The language follows the idea of building Web services specifications in a composable and modular way. It is built on top of specifications which are considered to be the core of the XML technology and Web services stack. Technologies such as reliable messaging, distributed transactions on Web and security are absolutely necessary for executing processes both across enterprises and within an enterprise. Nevertheless, they are out of scope of the language and other specifications must be used in conjunction with the BPEL specification to provide these capabilities.

The language introduces the same concepts for both executable processes and observable behavior of Web services. With this approach it is easier to build valid executable completions of an abstract process, which provides the observable behavior of a Web service. A valid executable completion of an abstract process is an executable BPEL processes that follows the same order concerning the interactions with a particular partner or a set of partners as the abstract process it implements. The language provides in the upcoming version 2.0 a set of rules that must be followed in order to get a valid executable completion of an abstract process. For example, it is allowed to add additional Web service interaction steps but they must refer to partner links not used in the abstract process. Also, the presence or order of Web services interactions already present in the abstract processes must not be changed, and it is not allowed to change, using an explicit data manipulation activity, the endpoint references of partner links used in the abstract process.

BPEL abstract processes complement abstract WSDL interfaces and the UDDI model. They define dependencies between service operations in the context of a message exchange. Technical note "Using BPEL4WS in a UDDI" [2] describes the relationships between the three models and suggests how BPEL abstract processes can be used in a UDDI Registry.

Support in SAP NetWeaver

SAP NetWeaver provides process integration capabilities and supports various integration scenarios including enterprise integration processes, collaborative processes crossing enterprise boundaries and integration of

human user interactions within processes. It offers support for both top-down as well as bottom-up development of processes.

SAP NetWeaver Business Process Engine is the runtime environment allowing the execution of processes that can take place between heterogeneous components with an enterprise as well as between business partners. The model of integration processes is compliant with the BPEL specification. A pluggable BPEL export/import interface is provided to import integration processes developed using other tools in the Integration Repository of SAP NetWeaver where the integration knowledge is stored. The model of integration processes supports the standard process flow constructs introduced in BPEL 1.1, but goes beyond them and offers more advanced features. For example, it supports multiple executions of the same set of activities either in parallel or in sequence, where the number is not defined at design time but at runtime and is based on the value of a process container element. In addition, it is possible that the process proceeds with the execution before all parallel activities complete and in this way executes more efficiently. The process model offers a rich set of constructs needed to build complex integration processes. The engine provides some basic data manipulation capabilities, while relying on external services for more complex data manipulation capabilities. The BPEL extensibility mechanism is used to export those advanced features into the BPEL 1.1 format and some of them will be introduced in the upcoming version 2.0 of the specification.

Beyond Version 2.0

The upcoming version 2.0 is long awaited specification. It contains many improvements compared to the BPEL 1.1 specification. However, version 2.0 misses not just interesting but also needed features.

Data manipulation capabilities are rather restricted in version 2.0 and this part must be revisited in a future version of the specification. Many scenarios, such as collecting multiple messages in one process container element which is used by subsequent activities, will be a pain point for process engineers.

Transactional semantics is another area that must be clarified.

The language lacks the support for modularization and reuse. Writing processes or process fragments that can be reused in different places within a process or even across multiple processes is desired practice especially in case of complex and large business processes. Version 2.0 does not provide a feature that supports that in a portable and interoperable way.

Automated processes extended with the aspects of human user interactions gained increasing attention recently. Version 2.0 primarily focuses on automated business processes and human user interactions are seen, rather, as an orthogonal aspect.

Papers [7] and [8] published by SAP and IBM, show directions how the last two issues could be addressed in the current BPEL proposal.

Summary

Automation of business processes both across enterprises and within an enterprise has been in the focus of the industry for some time. Web services seemed to be good basis, but process definition capabilities were missing. The Web Services Business Process Execution Language emerged as the most promising model for business processes based on Web services. This article explains the main usage scenarios and gives an overview of the language. Also, the support of the language in SAP NetWeaver is briefly discussed.

Currently, the language is under an OASIS standardization process. In parallel to the standardization processes, the areas currently not addressed in the specification have been discussed and proposals for modularization and reuse [7], and human user interactions [8] in BPEL were published.

References

[1] *Business Process Execution Language for Web Services (BPEL4WS)*, Version 1.1, BEA Systems, IBM, Microsoft, SAP AG and Siebel Systems, May 2003

[2] *Using BPEL4WS in a UDDI*, Committee Technical Note, OASIS UDDI TC, June 2004

[3] *Web Services Description Language (WSDL)*, Version 1.1, W3C Note, March 2001

[4] *Web Service Business Process Execution Language (WS-BPEL)*, Version 2.0, Working Draft, OASIS Technical Committee, September 2005

[5] *Web Services Flow Language (WSFL)*, Version 1.0, Frank Leymann, IBM, May 2001

[6] *Web Services for Business Process Design (XLANG)*, Satish Thatte, Microsoft, 2001

[7] *WS-BPEL Extension for Sub-processes (BPEL-SPE)*, White Paper, IBM, SAP AG, September 2005

[8] *WS-BPEL Extension for People (BPEL4People)*, White Paper, IBM, SAP AG, July 2005

[9] *XML Path Language (XPath)*, Version 1.0, W3C Recommendation, November 1999

[10] *XML Schema Part 1: Structures*, W3C Recommendation, October 2004

[11] *XML Schema Part 2: Datatypes*, W3C Recommendation, October 2004

Author Bio

Ivana Trickovic is a standards architect in the SAP NetWeaver Industry Standards team. Her work focuses on technology standards concerning the area of business process management and Web services. She is a member of the OASIS WS-BPEL Technical Committee.