SDN Community Contribution

(This is not an official SAP document.)

Disclaimer & Liability Notice

Modularization and reuse in WS-BPEL

Applies To:

The technology discussed in this article is relevant for SAP NetWeaver. More precisely, it is related to layers that deal with the process integration and business process management.

By: Ivana Trickovic

Company: SAP AG

Date: 13 Oct 2005

Abstract

This article explains briefly the need for an extension to the Web Services Business Process Execution Language (WS-BPEL) that supports modularization and reuse. It discusses the requirements for such extension and possible ways how to resolve them. It should help readers understand why SAP and IBM have worked jointly on a proposal for sub-processes in WS-BPEL.

Introduction

The Web Services Business Process Execution Language, version 2.0 (WS-BPEL) has gained a lot of traction recently. The language is suitable primarily for designing processes which orchestrate activities of different software components exposed as Web services. This is an important step towards building fully automated business processes. The model introduces the process logic as a first-class element while reusable business logic is encapsulated in Web services.

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. For that it is necessary to have a language that supports modularization and reuse. The WS-BPEL language does not provide a feature that supports that in a portable and interoperable way.

The *WS-BPEL Extension for Sub-processes – BPEL-SPE* white paper[1] published by SAP and IBM, describes how the WS-BPEL language needs to be extended in principle to enable reuse of WS-BPEL process fragments or processes within a process or across multiple processes. The value of the proposal is that it outlines an approach to support modularization and reuse in a direct way in WS-BPEL. The proposal covers different scenarios, from factoring processes into reusable process fragments that can be reused within a single WS-BPEL process to processes that can be invoked as WS-BPEL processes or can be treated as part of another process.

The sections below explain further the requirements and possible approaches how to address them. It is discussed why the approach taken in the white paper is preferable over the others.

---

[1] The white paper is available at SAP Developer Network and IBM developerWorks.

Problem Statement and Goal

Modularization in WS-BPEL is about the ability to write process fragments once and use them multiple times within the same process or across multiple processes. If a process fragment needs to be changed it is not required to go through all usages of that process fragment to apply the change. Writing process fragments that can be reused in different places is desired practice especially in case of complex and large business processes. For that it is necessary to have a language that supports modularization and reuse. The latest version of the WS-BPEL language, which is currently under an OASIS standardization process, does not provide a way to do that. It does not support a definition of WS-BPEL fragments that can be invoked from within the same or from different WS-BPEL processes.

It has been argued that this is a problem of modeling tools and no particular feature is required in the WS-BPEL language to support this more directly. One may also argue that a process can be decomposed into WS-BPEL processes which would be exposed as Web services and invoked using the WS-BPEL `invoke` activity. However, a WS-BPEL fragment is similar to a WS-BPEL scope in that the lifecycles of the calling process and the called WS-BPEL fragment are not independent. For example, if the calling process terminates the called WS-BPEL fragment has to be terminated too. This means that the termination signal must be propagated down to the chain of WS-BPEL fragment calls. The same is true in case of compensation. The other direction is interesting as well. For example, if a WS-BPEL fragment is explicitly terminated the signal must be propagated to the calling process. Obviously, the lifecycle dependency between the calling process and the called process fragment is an important aspect, which cannot be fully met by using the WS-BPEL `invoke` activity.

Modularization and reuse have been identified as a requirement for WS-BPEL, although a resolution has been postponed for a future version of the language. Some vendors have already provided proprietary implementations. Standardization of these concepts is important to obtain portability and interoperability. The white paper discusses requirements and outlines concepts needed to support modularization and reuse, in a portable, interoperable way.

Possible approaches

There are different approaches of how the problem of the modularization and reuse in WS-BPEL could be resolved. Three approaches are briefly described below.

Macros in WS-BPEL. Macros are a mechanism used in programming languages for code factoring. They range from simple text substitutions to code transformations. Macros as text substitutions have been thoroughly discussed within the OASIS WS-BPEL Technical Committee. In WS-BPEL, they would have similar structure as WS-BPEL scopes, except the ability to see variables defined in the parent scope(s). A macro would not need to be a valid WS-BPEL process but just a fragment of WS-BPEL code and a pre-processor would take care to complete text substitution in the target process. It would occur as a scope in the target WS-BPEL process. Syntax and static analysis of a WS-BPEL process could be performed as soon as all usages of macros are resolved. This approach is complex from the usability point of view. Also, it does not make possible to use a WS-BPEL process exposed as a Web service as a (reusable) part of another WS-BPEL process.

Using the WS-BPEL onEvent structure. Fragments of WS-BPEL code could be defined as event handlers and as such they can be called multiple times within the same process instance. However, this approach requires an intra-process communication mechanism which is not provided in the language yet. Alternatively,

partner links could be used for modeling intra-process communications, but initially they have been introduced for designing interactions with partners. Also, this approach does not address cases where WS-BPEL fragments are defined as WS-BPEL processes. In that case, a WS-BPEL process would be exposed as a Web service and the same invocation mechanism, that is the WS-BPEL `invoke` activity, would be used as for other Web services. That means this approach has same drawbacks as the language today does.

Sub-processes in WS-BPEL. The idea is to define a new structure – called sub-processes - in WS-BPEL that represents WS-BPEL fragments. These sub-processes would be defined either locally within a process and reused only within that process or as a WS-BPEL process and reused across other WS-BPEL processes. Additionally, an accompanying invocation mechanism would be defined that supports calling sub-processes in the context of a WS-BPEL process. The mechanism must ensure that a sub-process is tightly coupled in terms of its lifecycle to the calling process, as explained in the previous section.

The third approach seems to be the most general one covering reusability within a single process as well as across multiple processes and being extensible to include invocation of sub-processes defined in different infrastructures. Therefore, the WS-BPEL Extension for Sub-processes – BPEL-SPE white paper uses the third approach; that is it introduces the notion of sub-processes. This new structure respects some syntactical restrictions; for example, a sub-process implements a request-response operation from the point of view of the calling process and no interaction beyond the initiating request message and the final response message is allowed between the sub-process and the calling process. A WS-BPEL process is used to define a fragment of WS-BPEL code that can be reused across multiple WS-BPEL processes. In this case, the WS-BPEL process respects also the same syntactical restrictions mentioned above. In additional, the white paper discusses different invocation scenarios and introduces an appropriate coordination protocol used for interoperable invocation of sub-processes across infrastructures from different vendors.

Summary

This article briefly discusses the problem process designers are facing using the WS-BPEL language with respect to modularization and reuse of WS-BPEL process fragments or processes. It concludes that this is not just a modeling tool issue but should be addressed more directly in the language itself. It discusses possible approaches and explains why the approach taken in the WS-BPEL Extension for Sub-processes – BPEL-SPE white paper co-developed by SAP and IBM is preferable over the others.

Author Bio

Ivana Trickovic is a standards architect in SAP's Platform Ecosystem Industry Standards Group. Her work focuses on technology standards concerning the area of business process management and Web services.