Information and Software Technology 51 (2009) 1261–1274

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

WS-BPEL Extensions for Versioning

Matjaz B. Juric [a,*], Ana Sasa [b], Ivan Rozman [a]

[a] University of Maribor, FERI, Institute of Informatics, Smetanova 17, SI-2000 Maribor, Slovenia
[b] University of Ljubljana, FRI, Information Systems Laboratory, Trzaska 25, SI-1000 Ljubljana, Slovenia

ARTICLE INFO

Article history:
Received 26 October 2008
Received in revised form 15 January 2009
Accepted 18 March 2009
Available online 1 April 2009

Keywords:
Versioning
BPEL
SOA
Business processes

ABSTRACT

This article proposes specific extensions for WS-BPEL (Business Process Execution Language) to support versioning of processes and partner links. It introduces new activities and extends existing activities, including partner links, *invoke*, *receive*, *import*, and *onmessage* activities. It proposes version-related extensions to variables and introduces version handlers. The proposed extensions represent a complete solution for process-level and scope-level versioning at development, deployment, and run-time. It also provides means to version applications that consist of several BPEL processes, and to put temporal constraints on versions. The proposed approach has been tested in real-world environment. It solves major challenges in BPEL versioning.

1. Introduction

In this article, we address the problem of versioning WS-BPEL (Business Process Execution Language) processes. BPEL has become the de-facto standard for orchestrating services in service oriented architecture (SOA). Versioning is an important topic in software development. It becomes even more important in SOA, where services are orchestrated into processes following the principle of loose coupling.

Services evolve over time in iterations, which result in new service versions. When orchestrated, BPEL processes need to be aware which versions of the services they use. Processes also evolve over time, which results in new process versions. Control over the versions becomes crucial, particularly in long-running processes, as we will explain later in this article.

Support for versioning in SOA is inadequate as neither BPEL nor other specifications such as WSDL (Web Services Description Language) and UDDI (Universal Description, Discovery, and Integration) provide explicit support for versions. In our previous work [22] we have proposed an approach to version services using WSDL and UDDI extensions. In this article, we focus on specific issues related to version support in BPEL. We propose extensions to BPEL to support versioning. We have addressed development, deployment, and run-time versioning of BPEL process themselves, and version control in orchestrations, where processes and services are consumed. We introduce new activities, extend existing activities, and introduce a version handler. We have designed the extensions for WS-BPEL version 2.0 using the standard language extension mechanism. We call the proposed extensions *WS-BPEL Extensions for Versioning*.

The article is organized as follows. In Section 2, we give a brief overview of BPEL. In Section 3, we explain the challenges in BPEL process versioning and highlight the current situation, which is inadequate. In Section 4, we outline our proposed solution, which consists of two parts, process versioning and partner link versioning. In Section 5, we describe the solution to version processes. We propose extension activities and attributes to the process and the scopes. We introduce a new handler type called version handler. In Section 6, we present version extensions to control the invocation of orchestrated (partner link) services and processes. We propose extensions for *invoke*, *receive*, *onmessage*, and *pick* activities and for the event handler. We also propose extensions for BPEL variables. In Section 7, we present the proof of concept, where we describe the implementation of the proposed extensions for versioning. In Section 8, we present related work and discuss the results. In Section 9, we give conclusions.

2. Brief overview of BPEL

BPEL is an OASIS standard [34] and has become the de-facto standard for service orchestration. BPEL is supported by the majority of SOA platforms and development tools [21]. It provides support for executable and abstract business processes. The current version is 2.0, which has been approved by OASIS in April 2007. BPEL 2.0 is an evolution of the previous version 1.1 and introduces several improvements, such as improved variable manipulation, enriched fault handling, improved correlation, local partner links,

* Corresponding author.
E-mail address: matjaz.juric@uni-mb.si (M.B. Juric).

0950-5849/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2009.03.003

dynamic parallel flows, improved loop handling, and extension mechanism, which allows adding extensions to the BPEL language in a standardized way [20]. We use the BPEL extension mechanism to add versioning support. This way it is possible to implement versioning extensions for any available BPEL server.

Other extensions have been proposed for BPEL. BPEL Extensions for Sub-Processes [28] provide the means for the invocation of a business process as a sub-process of another business process, such that its lifecycle is coupled to the lifecycle of the parent process. BPEL Extension for People (BPEL4People) [1] addresses human interactions and introduces a new type of basic activity which uses human tasks as an implementation, and allows specifying tasks local to a process or use tasks defined outside of the process definition. This extension is based on the WS-HumanTask specification [2], which defines human tasks, including their properties, behavior, and a set of operations used to manipulate human tasks. AO4BPEL introduces aspect-oriented extensions to BPEL [7]. AdaptiveBPEL supports the development of differentiated and adaptive BPEL processes and is also based on the aspect-oriented concepts [12]. C-BPEL is an extension that incorporates context information and uses it for service composition [15]. BPEL4Chor is an extension for modeling choreographies [10].

3. Versioning in BPEL

Versioning is very important in software engineering [4], release planning [39] and particularly in SOA development [14,19]. In process orchestration, there are two aspects of versioning, which should be addressed: versioning of the process itself, and versioning of the partner links. Versioning of the process requires that we have means to manage and support the different process versions as a result of continuous development and process improvement. This includes the ability to deploy different versions of the same process and run them simultaneously. Here a special challenges are long-running processes, which can execute several days, weeks, or even months. When we deploy a new version, the existing instances have to be completed according to the old specification of the process. This means that several versions of the same process have to co-exist on the process server. Another challenge is clients who invoke processes. In SOA environments, we usually do not control all the clients a process has. In such cases, it is virtually impossible to upgrade all the clients at the same time as we deploy the new version of a process. This requires not only that different versions of the process co-exist on the same server, but also requires that the clients are able to invoke a specific version of the process.

Second aspect of versioning is the ability that a BPEL process invokes a specific version of a service or a process it consumes through a partner link. This includes the ability to discover version information and to bind to a specific version of a partner link. This means that we have to retain version information not only at development-time but also at run-time. Related to this requirement the BPEL process should also be able to handle situations where a specific version of a partner link service/process is unavailable and invoke another (possibly backward-compatible) version. Finally, the process should also be able to partner link services, which are not versioned.

BPEL currently does not provide any support for versioning. This is a serious drawback, which is addressed by some BPEL servers, such as IBM WebSphere Process Server [33] and Oracle BPEL Process Manager [35]. They provide very limited support for deployment-time versioning and allow deploying different processes under the same name, but with different version numbers. Usually two approaches are used. First is that the only accessible version is the latest version of the process, this is one that has been deployed most recently. Previous versions are only available to finish existing running process instances. The second approach is to publish different versions of the process under different endpoint URLs, which basically means that each process version is published as a separate endpoint. The problem of this approach is that there is no standard naming convention for version information. There is also no standard API to invoke different versions of the same process.

BPEL specification also does not address the problem of long-running processes, and what happens with them when a new version of a process is deployed. Existing instances of processes, which have been started using a previous version, should finish according to the old version specification. Related to this problem is the problem of upgrading services, which are consumed by a BPEL process. Suppose that we have started a long-running BPEL process. While the process is executing a service that this process invokes is upgraded to a new version. Should the already running process invoke the new or the old version of this service? BPEL does not provide any explicit support for invoking a specific version of a partner service. Currently the only solution for developers is to use different endpoint URL names for different service versions. This approach however is very inflexible, makes maintenance difficult, and at the same time considerably reduces the flexibility of such solutions. This approach is also limited to services that we control. For external services we cannot influence when they are upgraded. In such cases, the developers have to be familiar with the behavior of a specific BPEL server implementation to foresee the actual behavior of a running process where services have been upgraded in-between. This is a drawback particularly if we develop BPEL processes for servers from different vendors.

4. Proposed solution for BPEL versioning

The proposed *BPEL Extensions for Versioning* address all objectives, mentioned in the previous section. Our proposed approach provides support for development, deployment, and run-time versioning of processes. The versioning extensions also provide support for invoking specific versions of partner link services or processes. To achieve this we introduce specific extensions to BPEL. These include new activities that enable us to denote versions of processes and scopes (such as <*bpelx:version*>), and extensions to existing activities, including <*partnerLink*>, <*invoke*>, <*receive*>, and <*onmessage*> within <*pick*> and <*eventHandlers*>. We also propose a new handler, called version handler <*bpelx:versionHandlers*>, which is used to define the behavior of the process when a process client invokes a specific version of the process. Version handler can be used for the process or for the individual scopes.

Our proposed extensions enable BPEL developers to use process-level or scope-level versioning and introduce process bunches for versioning applications that consist of several BPEL processes. Versioning of processes and services has become an essential part of SOA development. Our solution makes processes version-aware. This way it simplifies planning for the fact that there will be many change requests [40]. Version-aware processes can consume version-aware services (as described in [22]) and version-unaware services. Our approach supports both version-aware and version-unaware process clients. Version-aware clients can select a process version and query for the version attributes. Our solution works best in environments where processes and involved partner link services use the proposed version extensions. This can be either within the same domain or between the domains. Our solution is also suitable for environments where processes consume partner link services for which we do not control their version cycle. In such cases, an appropriate approach is to develop version-aware façades in order to maximize benefits of the version extensions.