BUSINESS PROCESS EXECUTION LANGUAGE FOR WEB SERVİCES (BPEL4WS)

Middle East Technical University
Department of Computer Engineering
D.Canturk, Ankara
Nov,2003

Abstract

Web services are standard-based software components that can be accessed over the Internet by other software components. The main goal of the web services is system integration, where web standards are supposed to provide a universal interoperability between applications. The functionality of a web service can vary from simple remote procedure call-like operation to complex systems, where a web service either initiates or is a part of a workflow, which accesses and combines information from multiple sources. Because the system integration usually requires more than just the ability to invoke simple operations, there is a need for tools to model complex business interactions. Business Process Execution Language for Web Services (BPEL4WS) is an XML-based workflow process language, which defines the notation for specifying business processes based on web services. BPEL4WS is a combination of previously competing workflow languages, WSFL and XLANG. In the paper I will describe the features of BPEL4WS, and show how the best features of WSFL and XLANG are adopted to the BPEL4WS.

Contents

1.Introduction

Until recent years, web means collection of pages that people or companies compose for the purpose of advertisement or information sharing with the static text or images. Nowadays web becomes a service provider. Web services are ordinary software that can be accessed over the by other software. Once web service published, other applications and web services can discover and invoke it. Moreover these web services are considered as part of workflow processes.

Web services are introduced for making the all applications operable independent from its platform. Therefore a web service can be either a single remote procedure call (RPC) or a complex system. In addition, web service is not only a part of and also initiates process which collects information from different sources and constructs and output.

In my thesis I am working on about Business Process Execution Language for Web Services (BPEL4WS) that enables to specify business process based on web service. BPEL4WS is combination of WSFL and XLANG with the best features of both. The specification of BPEL4WS is an ongoing work collaboration of big companies, such as BEA, IBM, and Microsoft. The initial BPEL4WS version 1.0 from July 2002 has been updated to the current version 1.1 on March 2003. In my work, I will study on "what are missing in BPEL4WS and how can be adopted to it?".

2.Related Work

Web service depend on Simple Object Access Protocol (SOAP) for messaging, Web Service Description Language (WSDL) for interface description and Universal Description, Discovery and Integration (UDDI) for discovery and integration. On the other hand current 1.1 version specifications of BPEL4WS is depend on WSDL 1.1, XML schema 1.0 and XPath 1.0, all off these three dependency WSDL is the most important one because BPEL4WS is built on top of it. Since WSDL provides stateless and single run process interactions, there is a need for another layer on top of WSDL for modeling the complex business processes. BPEL4WS provides stateful and long running interactions between two more parties containing sequences of messaging both synchronous and asynchronous for modeling complex process.

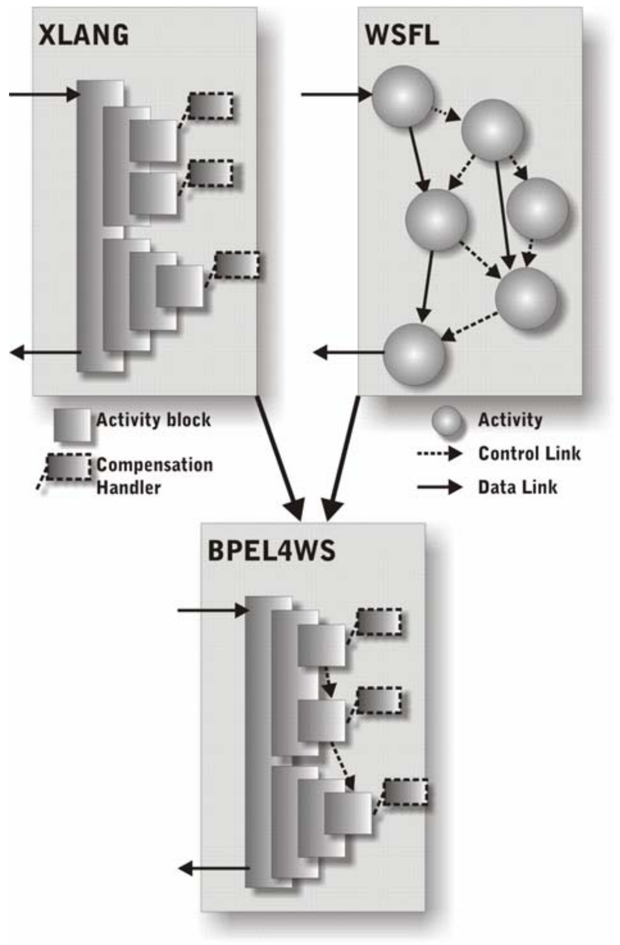
BPEL4WS is not the only XML-based process modeling language. There are plenty of them. For example, Web Service Choreography Interface (WSCI) is one of them, which is the most serious competitor for BPEL4WS. WSCI has supports from Sun, BEA and SAP. WSCI provides flow of message exchanging to the web services participating choreography interface with other services. The main purpose of WSCI is application-application integration. Thus, it has the problem "How to APIs that are web enabled cooperate with each other?". Business Process Modeling Language (BPML) is another language that models the abstract and executable process, which points to all aspect of business process such as transactions and their compensations, data management, concurrency and exception handling.

3.Background

Integrating complex systems requires more than WSDL because model for business interactions assumed as sequences of both synchronous and asynchronous message exchanging, within stateful and long running interactions between participating sides. To define such a model formal message exchanging protocol is required. These business protocols should be described in a platform-independent manner since the main goal is system integration, and different systems may have different running platforms. XML is the ideal language for describing platform independent workflows. According to pervious section there are many XML-based language for modeling business protocols. Two of them are essential for BPEL4WS, which are Web Service Flow Language (WSFL) and XML business process language (XLANG).

WSFL is IBM's XML language for describing business protocol in a web service manner. It was designed to be a part of web service technology framework and complements the existing modeling such as SOAP, WSDL and UDDI in a dependent way to these models. Unlike to XLANG, WSFL is not limited to block structures, but allows directed graphs. These directed graphs can be nested but must be acyclic since the process may have parallel execution and conditional branches. However, WSFL cannot satisfy the transactional requirements of the process. In other words you cannot compensate the failure of some points in the process. Therefore, WSFL processes cannot be considered as atomic. In WSFL the smallest unit of work is defined as activity. Between these activities there are two kinds of links exists which are control and data link where control link points to the order of execution and data link points to the flow of data between activities.

XLANG, Microsoft's XML business process language, is block structured language providing basic control flow structures such as *sequences, switch* for conditional routing, *which* for looping, *all* for parallel routing, *pick* for race conditions based on external triggers. One of the core ideas of XLANG is to support atomicity in the process unlike to WSFL. That is, in XLANG defined compensation handlers can be called wherever a failure occurs in the process to rollback the process. For example in XLANG you can model the following workflow "if all of the activities in the workflow succeeds, the whole workflow succeeds but in case of any failure, the whole workflow fails and need to be rolled back to initial state.". The other important feature of XLANG is correlation and instantiation. Long running business process usually requires a same instance of activity or a set of activities for the groups of transactions. In XLANG you can create the new instance of any activity and the messages as well as the data properties can be grouped in a correlation set specific to instance.



BPEL4WS is the combination of WSFL and XLANG with the best features of these two. BPEL4WS uses the rich control structures from WSFL and atomicity from XLANG. It has the following control structures: sequences, parallel flows, conditional branches and looping. To define an atomic process BPEL4WS allow to define fault and compensate handlers. Figure on the left visualize the combination.

4.Overview to BPEL4WS

The first and initial version of BPEL4WS, version 1.0, was released in July 2002. The second version, 1.1, was released in March 2003. The version 1.1 clarifies the core concepts, adds some functionality, and enhances some of the features specified in the version 1.0. The new terminology used here is "business protocol" specified as abstract processes, however ebXML has worked on this concept since fall 1999 and provide a specification on 2001.

4.1 Abstract and Concrete Executable Process Modeling

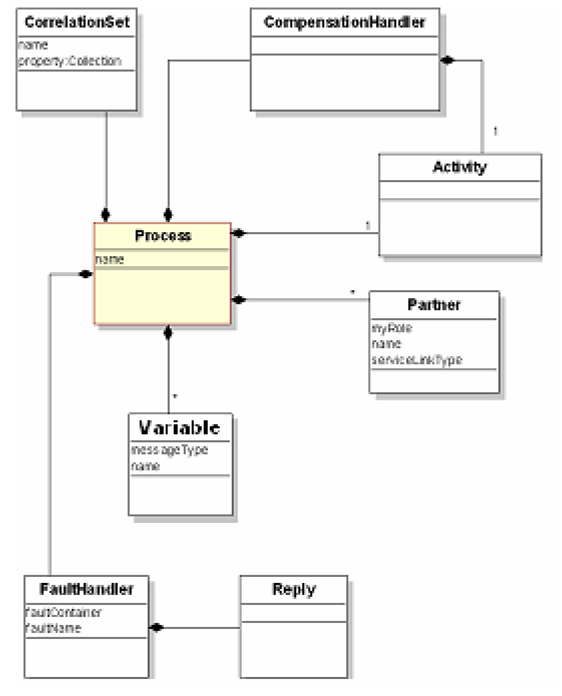
The basic concepts of BPEL4WS can be applied in one of two ways: modeling abstract processes and executable concrete processes.

A BPEL4WS process can define a business protocol role, using the notion of abstract process. For example, in a supply-chain protocol, the buyer and the seller are two distinct roles, each with its own abstract process. Their relationship is typically modeled as a service link. Abstract processes use all the concepts of BPEL4WS except data handling because in an abstract process private aspects of the behavior is not revealed. Therefore, abstract processes handle only protocol-relevant data. To identify protocol-relevant data, BPEL4WS provides a way as message properties. In addition, abstract processes use nondeterministic data values to hide private aspects of behavior.

Second way of using BPEL4WS is to define an executable business process. The logic and state of the process determine the nature and sequence of the web service interactions performed at each business partner, and thus the interaction protocols. In the modeling of executable business process, the process in considered to initiate by receiving a message from some partner, executing a set of actions, and finally replying to the partner. For the execution of the activities the BPEL4WS provides a rich set of control structures, and possibility to invoke other web services.

BPEL4WS defines a model and a grammar for describing the behavior of a business process based on interactions between the process and its partners. The interaction with each partner occurs through WSDL, and the structure of the relationship at the interface level is kept in a service link. The BPEL4WS process defines how multiple service interactions with these partners are coordinated to achieve a business goal, as well as the state and the logic necessary for this coordination. BPEL4WS also introduces systematic mechanisms for dealing with business exceptions and processing faults. Finally, BPEL4WS introduces a mechanism to define how individual or composite activities within a process are to be compensated in cases where exceptions occur or a partner requests reversal.

4.2 The General Structure of BPEL4WS Workflow



Partner section specifies the external web services invoked from within the workflow. Partner definition has also reference to the WSDL documents of the web service. Variables contain the data that flows within the workflow. In the previous version 1.0 of BPEL4WS, the variables were called as "Containers". Correlation sets are used to bind a set of operations to a service instance. Fault handlers are used in catching failures and the sequence comprises the actual workflow logic. Compensation handlers are used to implement compensating actions in the case of transaction rollback. Finally, the actual business logic is represented as a group of activities, which are executed in a structured way. Invoking web services does the actual work in the activities. The business logic may include basic control flow structures, such as sequence, switch (for conditional routing), while (for looping), and pick (for race conditions based on external triggers). In the next section these features and concepts are described in more detail.
A BPEL4WS workflow instance is initiated when the initial input message is received. The workflow terminates in one of the following ways:

- When the last activity in the workflow terminates (this is the normal way to terminate the workflow).
- When a fault occurs, and it is not handled appropriately (this is considered as an abnormal termination).
- When a workflow is terminated explicitly by a terminate activity.
- If the workflow cannot proceed, for instance due to some error, and a compensation handler for the workflow is called and executed.

5 BPEL4WS Concepts

5.1 Service Linking Between Partners

One of the most important features of BPEL4WS is the modeling of cross-enterprise business interactions. While the WSDL already is able to express the functionality of one service, both at abstract and concrete levels, it is not able to model a set of services composing or taking part to a business process

In BPEL4WS the relationship between business processes is considered peer-to-peer, thus requiring a two-way dependency at the service level. To model such dependencies, a concept of service link type is specified. A service link type defines the roles played by each of the services and specifies the portTypes provided by each role. For instance, a service link type for BuyerSellerLink could specify two roles, Seller- and Buyer-role, and both of these roles are mapped to use a specific portType. In BPEL4WS, asynchronous message passing between partners is the primary way of exchange information. Despite of the above explanations, the difference between an abstract process, a business protocol and a service interface is not seemed to be explain very well in the specifications. In any case, there seem to be no elements or series of elements supporting the concept of business protocol as defined in the specifications.

The services with which a business process interacts are modeled as partners. Each partner is linked or characterized by a service link type. Each partner specification includes the partner name, the corresponding service link type, and the role played by the partner itself, the role played by the peer partner. For instance, a partner named FoodSeller can be linked to the BuyerSellerLink as described earlier, and the role for the FoodSeller itself would in this case be the Seller, and the peer partner role would be the Buyer. According to the specifications, "partner" is a seemed to be required attribute of these activities, which impose us that messages can only be exchanged with the other companies' services. In particular a process could not be used to describe the message exchange between any two internal components (e.g. order entry and billing).

5.2 Data Handling

5.2.1 Expressions

BPEL4WS specifies several types of expressions: boolean-valued, deadline-valued, duration-valued, and general expressions. Boolean expressions typically are used in transition conditions, join conditions, while conditions, and switch cases. Deadline-valued expressions can be found for instance in until- and wait-structures. An example of a duration-valued expression is the for-loop attribute. General expressions are used in assignments.

5.2.2 Variables

Variables provide the means for holding messages that constitute the state of a business process. The messages held are typically those that have been received from partners or are to be sent to the partners. However, BPEL4WS does not restrict to this, but allows variables to store any "temporary" data that are never exchanged with partners. In BPEL4WS, there is no data flow between activities, but instead, the activities inside the workflow exchange information using global variables.

The type of variable must be a WSDL message type. However, BPEL4WS allows the message type to be inlined in the variable declaration rather than being defined in a WSDL document. The variables can be specified as the input or output variables for invoke, receive, and reply activities. More detailed description of these activities will be given later.

5.2.3 Assignment

The assign activity can be used for copying data from one variable to another, as well as to construct and insert new data using expressions. The assign activity is defined inside a <assign>-block, where a copy operation is performed. The copy operation is defined in a block with two subelements: <from> and <to>. The to element may refer to a part in a variable, service of a partner, or a property. In addition to these, the <from> element may also refer to an expression, or a literal.

5.3 Correlation

Business processes between organizations may be long-running workflows, which are possibly composed of multiple conversations, and which need to maintain state of execution. Because web service operations are by definition stateless operations, BPEL4WS adds the support for execution state by specifying correlation.

A group of operations can be specified to belong to the same correlation set within a service instance. A set of correlation tokens is defined as a set of properties shared by all messages in the correlated group. In the beginning of the workflow execution, all the correlation sets are uninitiated (notice the analogy to the object-oriented languages, where the object instance needs to be created before its use).

Correlation sets can be used in messaging activities, such as receive, reply, and invoke. Within these activities, a correlation set can be initiated, if needed. After initiated, the values of the properties for the correlation set remain valid within the corresponding scope of the activity until its completion.

5.4 Basic activities

Each activity has an optional standard attributes: a name, a join condition, and an indicator whether a join fault should be suppressed if it occurs. A join condition is used to specify requirements about concurrent paths reaching at an activity.

5.5 Control Structures

Sequence, Switch, While, Pick and Flow are the control structures of BPEL4WS. A sequence activity contains one or more activities that are performed sequentially. The switch activity consists of an ordered list of one or more conditional branches defined by <case> elements, followed optionally by an <otherwise> branch. The <case> branches of the switch are considered in the order in which they appear. The while structure supports iterative activity. The iterative activity is performed until the given boolean while condition no longer holds true. The pick activity awaits the occurrence of one of a set of events and then performs the activity associated with the event that occurred. The form of pick is a set of branches of the form event/activity, and exactly one of the branches will be selected based on the occurrence of the event associated with it before any others. The most fundamental semantic effect of grouping a set of activities in a flow is to enable concurrency.

6.Discussion

In my thesis, I want to find out what is missing in BPEL4WS and how missing part will be adopted to it. There are two versions of BPEL4WS, version 1.0 and 1.1, of which the latest version has just been published. Although the BPEL4WS combines the best features of WSFL and XLANG, there are still some open issues in the specification. For instance, there is no way to transform data between activities. Instead, common global variables are used as the way to implement data flow between activities. But, WSFL allows dataflow between activities. I said before BPEL4WS combines best features of both but I cannot understand why this functionality has not been adopted to BPEL4WS. One way to do this would be to extend the <assign> activity to cover copying data from activity to another.

The other serious problem is the lack of user interactions. There is no ways to specify any kind of user interaction to control or affect the execution of workflow, or help to deal with exceptions and compensations. BPEL4WS allows only modeling machine-to-machine interactions. I suggest that some kind of user interaction, at least today's B2B processes, need to be added.

This model of BPEL4WS is used in a static way. That is, workflows are constructed by using the static bindings of web services. When this workflow let run, same web services will be called. Web service notion is being widely accepted web service invoking failure due to unavailability of the service occurs more frequent then today, so dynamic web service binding can handle this problem. If the BPEL4WS workflow is failed due to a crash or down of network connection of the web service being used, dynamic web service binding replace this service with the alternative one that is searched and found from registries. In notion is included in the semantic web technology.

Lack of data transforms between activities, user interactions, and dynamic web service bindings are the current missing on BPEL4WS in my decisions. In the next stages part of my thesis work, I tried to adopt these features to BPEL4WS

7.Conclusion

The web is moving from a collection of web pages to a collection of web services, and further, to a collection of workflows composed of web services. In this thesis work I study on BPEL4WS, which defines the notation for specifying business processes based on web services. BPEL4WS presents the combination of previously competing specifications, WSFL and XLANG, and combines the best features from these two. The main contribution from the WSFL is the rich capabilities to support control flows, parallel execution, and synchronization. XLANG brings the concept of atomicity to the BPEL4WS in the forms of compensations and correlations.

Although the contribution of the big companies IBM and Microsoft, current specification of the BPEL4WS has some open points. With the addition of the required functionalities BPEL4WS play the most important role in the web service composition area.

Referances

- W3C, "SOAP Version 1.2 Part 0: Primer," N. Mitra, Ed., Dec. 2001, Working Draft, available at: http://www.w3.org/TR/soap12-part0/.

- W3C, "Web Services Description Language (WSDL) Version 1.2," R. Chinnici, M. Gudgin, J.-J. Moreau, and S. Weerawarana, Eds., Jan. 2003, W3C Working Draft 24, available at: http://www.w3.org/TR/2003/WD-wsdl12-20030124/.

- OASIS Consortium, "UDDI Version 3.0, Published Specification," July 2002, Working Draft, available at: http://www.oasis-open.org/committees/uddi-spec/.

- W3C, "XML Schema Part 0: Primer," D. C. Fallside, Ed., May 2001, W3C Recommendation, available at: http://www.w3.org/TR/xmlschema-0/.

- W3C, "XML Path Language (XPath), Version 1.0," J. Clark and S. DeRose, Eds., Nov. 1999, W3C Recommendation, available at: http://www.w3.org/TR/xpath/.

- A. Arkin, S. Askary, S. Fordin, W. Jekeli, K. Kawaguchi, D. Orchard, S.Pogliani, K. Riemer, S. Struble, P. Takacsi-Nagy, I. Trickovic, and S. Zimek, "Web Service Choreography Interface 1.0," 2002, available at: http://dev2dev.bea.com/techtrack/wscispecprint.jsp.

- A. Arkin, "Business Process Modeling Language (BPML), Working Draft 0.4," 2001, available at: http://www.bpmi.org/.

- T. ebXML Business Process Project Team, "ebXML Business Process Specification Schema, Version 1.01," J. Clark and S. DeRose, Eds., May 2001, Available at: http://www.ebxml.org/specs/ebBPSS.pdf.

- A. Banerji, C. Bartolini, D. Beringer, V. Chopella, K. Govindarajan, A. Karp, H. Kuno, M. Lemon, G. Pogossiants, S. Sharma, and S. Williams, "Web Services Conversation Language (WSCL)." HP, 2001.

- Workflow Management Coalition, "Workflow Process Definition Interface XML Process Definition Language, Version 1.0," Oct. 2002, Document Number WFMC-TC-1025, available at: http://www.wfmc.org/standards/docs/TC-1025_10_xpdl_102502.pdf.

- S. F. Society, "SDL Homepage," Nov. 2000, available at: http://www.sdl-forum.org/SDL/index.htm.

- W. van der Aalst and A. ter Hofstede, "Invited Talk: Workflow Patterns: On the Expressive Power of (Petri-net-based) Workflow Languages," in Fourth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools (CPN'02), Aug. 2002, available at: http://www.daimi.au.dk/CPnets/workshop02/cpn/papers/Aalst.pdf.