print back

**URL of the article: http://www.jax-magazine.com/itr/online_artikel/psecom,id,741,nodeid,147.html**

# BPEL: Rounding Off the Essentials

Related Technologies, Misconceptions, and Extensions

*by Pranam Kolari and Li Ding*

BPEL (BPEL4WS) is the Business Process Execution Language for Web Services, a popular Web Service Composition Language and possibly the future standard. In this article we present the capabilities of BPEL and its associations with other technologies and formalisms. We further discuss some of the existing BPEL engines, their extensions to BPEL and motivation behind these extensions.

Web Services are touted to be the one of the biggest technology drivers over the next decade. Gartner predicts that the market for web services based solutions will be $30 billion in 2005. Fu et al [3] specify the goal of web services as \to have a collection of network resident software services accessible via standardized protocols, whose functionality can be automatically discovered and integrated into applications or composed to form more complex services". The W3C has put together a set of specifications (WSDL, SOAP, UDDI) which form the basic infrastructure required for web services. The more comprehensive Web Services stack contributed to by multiple standardization efforts now consists of a complete suite of language specifications ranging from Policies (WS-Policy), Transaction (WS-Transaction) to Agreement (WS-Agreement) and composition (BPEL). In this article we discuss BPEL, a composition language for Web Services.

The specification of BPEL was initiated with the primary objective of standardizing Web service composition in the B2B domain. It draws heavily upon earlier work on WSFL (Web Service Flow Language) from IBM and XLang from Microsoft, and is widely supported by many B2B integration solution providers. The specification is currently in version 1.1. BPEL can be used either within an organization or across organizations (B2B applications) where multiple web services are part of a workflow. The primary requirement from such applications is the notion of loosely coupled systems which allows adapting the workflow with minimal monetary costs and strict time constraints. Service Oriented Architectures (SOA) is the first step in this direction, of which BPEL forms an integral part.

- **Where does BPEL fit?** BPEL was designed to target two application scenarios i.e. to be able to represent *abstract* and *executable* business processes. *Abstract processes* are used for inter-organizational sharing of business workflows. This is used to communicate complex processes to partners giving only sufficient detail for B2B integration without divulging the private aspects of the process. *Executable processes* are a more complete definition of the workflow and can be executed by a BPEL execution engine (i.e. it is syntactically and semantically complete).
- **Who will use it, why and what for?** BPEL can be used by solution architects to specify business processes for the purposes of sharing across organizations, and by developers to implement a workflow. Note that multi-party integration can be implemented using traditional programming languages as well. But what makes BPEL stand apart is its applicability to the Service Oriented Architecture (SOA) and its conformance to the goals set by SOA.
- **Does it supplant any older technologies?** BPEL is expected to replace XLang and WSFL as a web service composition standard adopted by B2B solution providers. Since it is highly liked to be the standard for web service composition, it will replace older and less mature composition languages.
- **Where do popular languages like Java fit in?** Java and other languages will continue to be used. The services are exposed using web services languages and composed using BPEL, but the services themselves will continue to be implemented in traditional languages. BPEL just replaces these languages in areas where looser coupling is to be employed to realize SOA. BPEL is also a language at a higher level of granularity (i.e at the service level of abstraction).

**Language Constructs**

The BPEL vocabulary provides a set of constructs to describe web service compositions. Some of the BPEL constructs can be compared to features provided by traditional programming languages, but all these constructs operate at a much higher level of abstraction. In the interest of giving equal attention to other aspects of BPEL, we will only briefly introduce available constructs.

The key constructs provided by BPEL can be classi¯ed based on the aspects of business integration(B2B) that they address.

- **Variables** - Web services are stateless software entities. Unlike web services, Business processes (i.e. composition of web services) are required to maintain some amount of state information which is enabled by

the use of variables. Variables are of XML Schema Type, and BPEL also specifies the possible access mechanisms for these variables using XPath notations.

- Partner Links - Partner Links represent the global relationship (or conversational aspect) of the business process. They are used to specify how the process interacts with other atomic services(and complex processes).
- **Event Handling** - A business process is also associated with events, which can be handled concurrently with other activities of the process. Events are usually used to provide a means for status enquiries by interested parties.
- **Exception Handling** - Exceptions occur during a process, when the normal execution of the process is interrupted due to a fault. Such conditions are handled using fault handlers which perform corrective measures. These constructs are analogous to the try and catch constructs available in programming languages.
- **Compensation Handling** - Compensation handling is similar to database rollback. In BPEL, when distributed services are part of a particular business process, rollback of a process presents some challenges. BPEL provides an explicit compensation construct which can be used to invoke services which are responsible for rolling back operations.
- **Correlation sets -** At any instant of time, a BPEL execution engine manages multiple instances of the same process definition. Correlation sets are a subset of fields of BPEL variables which can be used to identify the right BPEL instances.
- **Activities** - BPEL provides constructs which are used to specify different kinds of activities in the business process. invoke, receive, reply is used to invoke other services, to wait for invocation by other services and to send back responses. Similar to traditional programming languages BPEL also provides control flow constructs like switch, flow, sequence, link. The flow construct is drawn from WSFL and sequence construct is drawn from XLang.

A BPEL process specified using the above language constructs can be executed using a BPEL compliant execution engine.

**BPEL Extensions and Engines**
One of the misconceptions with BPEL is that it should provide all possible features required for any application domain. Though BPEL does provide constructs which can be used to express many business scenarios, it is not designed with the intention of being a utopia to all business situations.

We repeat a section of the BPEL specification with a specific purpose. The design principle behind BPEL is not rightly understood by many users. It forms an important guide for the designers of BPEL and for extensions provided by some of the current BPEL engines. Even where private implementation aspects use platform-dependent functionality, which is likely in many if not most realistic cases, the continuity of the basic conceptual model between abstract and executable processes in BPEL4WS makes it possible to export and import the public aspects embodied in business protocols as process or role templates while maintaining the intent and structure of the protocols. This is arguably the most attractive prospect for the use of BPEL4WS from the viewpoint of unlocking the potential of Web Services because it allows the development of tools and other technologies that greatly increase the level of automation and thereby lower the cost in establishing cross-enterprise automated business processes." Tool developers are expected to provide additional constructs to BPEL based on business needs, maintaining a careful balance between portability and requirement. BPELJ, BPEL+ are some of the extensions, and some others are currently in development (like BPXL). We introduce some execution engines with an emphasis on the extensions they provide.

**Oracle(previously Collaxa) Process Manager** provides additional constructs for manipulation of BPEL variables. It targets extensions to the BPEL assign construct, which is insufficient in some application domains where variable sizes(e.g XML Schema unbounded array) dynamically change at runtime.

**IBM Process Choreographer** is yet another BPEL engine which provides some extensions(BPEL+) to capture the notion of "staff" activities. BPEL does not make a distinction between services which involve only machine based computation and manual input. Processes with manual input are usually long running and asynchronous with additional requirements to specify role players and access restrictions. BPEL+ also supports minimal inclusion of Java snippets into a BPEL process definition.
An additional BPEL extension is the BPEL-J(BPEL-Java) specification (http://www-106.ibm.com/developerworks/webservices/library/ws-bpelj/) which combines the benefits of both BPEL and Java. BPELJ ¯ts nicely into domains where some additional language features of Java are to be used, at the cost of portability which draws lower priority. At the time of writing this article, BPEL-J engines are still in developement.

Many open-source implementations (e.g. Active BPEL - http://www.activebpel.org/) of BPEL engines are either available or currently under development. Such implementations provide solution providers a good starting point to create extensions suiting their requirements.

**Mapping BPEL in Model Driven Architecture**
BPEL based processes are currently being designed by architects (and developers) through the use of a BPEL specific editors which provide graphical interfaces to process designers. The Model Driven Architecture put forward by the OMG requires interoperability between higher level models and BPEL so as to exploit all the advantages of Model Driven Architectures(MDA). In other words, mappings should exist between various meta-models and BPEL.

The research community is currently addressing automatic mappings to BPEL. This draws from past work in compiler based

technologies to translate from one language to the other. Specifically, there has been work in the translation from UML like languages which allow specification of processes in a much higher level of abstraction. The mapped BPEL process is then worked on by developers to create an executable process. Readers interested in exploring this area or using MDA with BPEL, in their solutions are referred to Baina et al.[1], Bezivin et al [2] and Koehler et al [4].

**Validating BPEL Processes**

One of the issues often overlooked by many users of BPEL is the issue of correctness of the process specification. The global behavior of multiple interacting business process in either an intra-organization or a B2B setting has to be analysed and veri¯ed for correctness before their use in real world.

Verification of correctness can be de¯ned informally as the answer to the question - Does the global behavior of the B2B system result in a state from which the required business goal cannot be reached?". Recently there has been work on applying techniques from modal checking to answer this question. This involves transforming BPEL processes into finite automata and verification of correctness of the behavior of communicating finite automata using model checking (e.g SPIN - http://spinroot.com/spin/whatispin.html) software. Though this process appears trivial, it requires signi¯cant research from the view of mapping BPEL, and bounding state space search. In most cases, in complicated business processes the message exchange patterns could lead to computationally infeasible searches by the model checking software.

One of the advantages of BPEL in the above scenario, is that its modularity allows easy verification of correctness which is critical to many business processes. Robust verification is largely ignored in traditional programming language based systems since communication aspects are not explicit as in BPEL. Readers interested in further details on these formalisms and tool support are referred to Fu et al[3] and Baina et al[1].

**Conclusion**

In this article we have presented BPEL, with a specific purpose of not just introducing the reader to language constructs, but also to give an overview of related technologies, misconceptions and extensions. Though BPEL addresses many issues for Service Oriented Architecutures, the true potential of BPEL is yet to be exploited.

**References**


- [1] K. Bana, B. Benatallah, F. Casati, and F. Toumani, *Model-driven web service development*, in Proceedings of 16th International Conference on Advanced Information Systems Engineering (CAISE04), 2004. Best Paper Award.
- [2] J. Bezivin, S. Hammoundi, D. Lopes, and F. Jouault, *Applying mda approach to b2b applications: A road map*, in Proceedings ofWorkshop on Model Driven Development (WMDD 2004) at ECOOP 2004, 2004.
- [3] X. Fu, T. Bultan, and J. Su, Analysis *of interacting bpel web services*, in WWW'04: Proceedings of the 13th international conference on World Wide Web, ACM Press,2004, pp. 621-630.
- [4] J. Koehler, R. Hauser, S. Sendall, and M. Wahler, *Declarative techniques for model-driven business process integration*, IBM Systems Journal, 44(1) (2005).


**About the Authors**


*Pranam Kolari is a Ph.D student and research assistant in the Department of Computer Science and Electrical Engineering at UMBC. His research interests include Semantic Web and Web Services. Pranam has a B.E. in Computer Science from Bangalore University(UVCE) and an M.S in Computer Science from the University of Maryland Baltimore County. In the past, Pranam has worked on projects at IBM Global Services, Bangalore and IBM T.J. Watson Research Center, New York. He currently holds a Ph.D fellowship from IBM Toronto Labs.*

*Li Ding is a PhD student and graduate research assistant in the Department of Computer Science and Electrical Engineering at UMBC. His research interests include artificial intelligence, semantic web and trust. He has a BS and an MS in computer science from Peking University, China. He leads the Semantic Web Search engine(http://swoogle.umbc.edu) effort at UMBC.*


---