The Essentials of BPEL

Kevin Gravesande, Steve Kim, Rasal Mowla, Al Resptrepo, Carlos Thomas, and Scott Weaver

March 26, 2008
1 Introduction

Web services can facilitate seamless business-to-business integration [1]. The business process execution language for web services (BPEL) is a standard for business process and integration protocols for web services [12]. The distributed nature of BPEL processes, the absence of a single stakeholder, the fact that partner services can dynamically change their functionality and/or QoS, and the fact we can define abstract processes and look for actual services at run-time, preclude design time validation of such systems [2]. The objective of these technologies is to automate process execution across people and systems [5]. BPEL represents business processes in XML. The BPEL specification was written by Microsoft, IBM, and BEA [8]. BPEL is a mixture of WSFL and XLANG; PD4J is the basis of for the Java extension of BPEL (BPELJ) [8]. BPELJ source code can be XML with embedded Java or Java with annotated XML [8].

According to [8] BPEL builds on and extends web services for interacting processes. The process consist of web services definition language (WSDL) and BPEL files. WSDL is the standard web service format. A BPEL process can be executable or abstract. BPEL processes have exactly one activity [8]. BPEL process can define a set of variables and pass those variables as inputs \outputs to web services [8]. Processes can be bound to the input of a inbound activity, output of a synchronous invoke & assigned a value with the assign activity [8].
2 Use Case Scenario

In various business case scenarios, BPEL can be used to provide full automation via Web services. BPEL is the language that cooperates to carry the automation with the business logic at its foundation. The following use case scenarios, online bookseller, travel agency, loan agency, and banking, provide an introduction to how BPEL is being implemented to perform an automated business process.

2.1 Online Bookseller

Consider what might happen when an online bookshop receives a book order online. In a Web services world, the booksellers computerized order system directly contacts the suppliers systems, places the order, receives notifications of its status, and informs the client about the status until all items are ready for shipment. From the bookshop's point of view, this business process involves the cooperation of several different Web services. To achieve full automation requires automating not only the Web services themselves but also the interactions among them. [9]

2.2 Travel Agency

In another scenario a traveler wants to book a flight. She therefore submits a trip order to a travel agency of her choice. The travel agency requests the price for the selected route and date from a set of airlines. Each airline responds and the travel agency selects the airline offering the best price. The agency orders a ticket from this airline and passes the traveler's email address on to the airline. As soon as the airline has confirmed the booking, the travel agency sends the itinerary to the traveler and the airline issues the eTicket. The airlines not selected for booking, stop waiting for the order as soon as a timeout occurs. [7] Figure 1 shows a sample choreography modeled using BPMN for the traveler’s flight booking. Figure 2 shows the messages being sent between the different entities.
2.3 Loan Agency

In another scenario, a customer wants to apply for a loan at the bank. Her application is processed by the assessor and sent via a web application where the loan process is done. The loan process retrieves credit worthiness and makes a decision. After the decision is made, the customer is informed via email of the status of her loan.

Figure 3 shows a part of the composite loan processing web service that includes Loan Process, Assessor, and Customer. The individual services may be specified in BPEL or, in simple cases, WSDL. The directed edges represent the communication links among the individual services. Note that Assessor may contact Customer directly. As a
result, there is no single web service that can keep track of the “global” state of the service execution.

2.4 Bank

In this last scenario, a customer uses an ATM machine to check his balance, retrieve cash and or make a deposit. The ATM terminal is connected via secured line to host computer, and bank computer. The user is either authenticated, in which case the transaction is processed, or denied access which may result in a message or a confiscation of his ATM card.

Figure 4: How ATMs Work [4]
3 Building a Business Process

A BPEL process indicates the order in which Web services should be invoked. The BPEL process requests a service from a particular Web services and then responds to the original caller. One invocation can depend on the output of the previous invocation.

BPEL uses Web services Definition Language (WSDL) in its definition since it communicates with Web services. Through construction of loops, variables, the copying of variables and exception handlers during invocation, BPEL can describe a complex business process.

A BPEL process consists of defined steps; each step is called a "activity." BPEL supports primitive as well as structure activities. Primitive activities represent basic constructs and are used for common tasks, such as the following:

- Invoking other Web services, using `<invoke>`
- Waiting for the client to invoke the business process by sending a message, using `<receive>` (receiving a request)
- Generating a response for synchronous operations, using `<reply>`
- Manipulating data variables, using `<assign>`
- Indicating faults and exceptions, using `<throw>`
- Waiting for some time, using `<wait>`
- Terminating the entire process, using `<terminate>`

These can then be combined with other primitive activities to define complex algorithms that specify the steps of the business process. To combine primitive activities, BPEL supports several structure activities. The most important are:

- Sequence (`<sequence>`), which allows the definition of a set of activities that will be invoked in an ordered sequence
• Flow (<flow>) for defining a set of activities that will be invoked in parallel

• Case-switch construct (<switch>) for implementing branches

• While (<while>) for defining loops

• The ability to select one of several alternative paths, using <pick>

Furthermore, each BPEL process will also define partner links, using <partnerLink>, and declare variables, using <variable>.

A BPEL process designer usually goes through the following design steps:

• Get familiar with the involved Web services

• Define the WSDL for the BPEL process

• Define partner link types

• Develop the BPEL process:
  – Define partner links
  – Declare variables
  – Write the process logic definition.

For instance, to book a flight, as described earlier, the following steps will take place: the client invokes the business process, specifying the client’s name, the destination, the departure date, and the return date. The BPEL business process first checks the client travel status, assuming that a Web service exists through which such checks can be made. Then the BPEL process will check the price for the flight ticket with two airlines: American Airlines and Delta Airlines. Again assume that both airline companies provide a Web service through which such checks can be made. Finally, the BPEL process will select the lower price and return the travel plan to the client.
4 BPEL Specification

Business Process Execution Language for Web Services (BPEL4WS) provides a means and a language to formally specify business processes and interaction protocols. By doing so, it extends the Web Services interaction model and enables it to support business transactions. BPEL4WS defines an interoperable integration model that should facilitate the expansion of automated process integration in both the intra-corporate and the business-to-business spaces.

BPEL4WS is layered on top of several XML specifications: WSDL 1.1, XML Schema 1.0, and XPath1.0. WSDL messages and XML Schema type definitions provide the data model used by BPEL4WS processes. XPath provides support for data manipulation. All external resources and partners are represented as WSDL services. BPEL4WS provides extensibility to accommodate future versions of these standards, specifically the XPath and related standards used in XML computation.

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 Web Service interfaces, and the structure of the relationship at the interface level is encapsulated in what we call a partner 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.
5 Players in the Market

BPEL has very important vendor backing including IBM with their WebSphere Business Integration product, BEA with their WebLogic Integration product, Microsoft with BizTalk Server 2004, Oracle with Oracle Business Integration, and several BPM specialists (including Fuego, SeeBeyond with ICAN, webMethods, FileNet, Staffware, Vitria). A full-featured BPM solution includes a process design tool, a runtime engine, administration and monitoring, and support for current integration technologies (for example, XML, web services, J2EE or .NET, and B2B). Examples of some of the products are shown in the following figures.

![BEA's WebLogic Workshop process design editor](image6.png)

Figure 6: BEA’s WebLogic Workshop process design editor [8]

![The NetBeans BPEL editor](image7.png)

Figure 7: The NetBeans BPEL editor [9]
6 How BPEL Works

6.1 A Loan Institution Scenario

Before we look at the details of how BPEL works, let’s take a deeper look at the loan scenario. Suppose a loan institution wants to automate their loan request process. Their current method requires a client to visit the loan office and talk with a loan officer. The loan officer would take the client’s information and the client would leave while the loan officer spent the rest of the day talking with financial institutions to find the client the best loan available.

The loan institution realizes they could utilize the web more effectively, so they developed a web site allowing clients to make requests online. This simplified requests by funneled all requests made online to a loan agent via email. However, that loan agent still spent time on the phone or computer, tracking down loan possibilities. The only part of the scenario that was automated was the request process. Everything else required human attention mainly because the loan institution and the financial institution are two different parties.

BPEL can be brought to bear to handle the whole process, even the part of the process between different institutions. It allows the loan institution to accept the request from the client, then forms a request to external entities like the financial institution, waits for the institution to respond, then crafts a reply to the client. The only time a loan officer would have to touch a request would be in circumstances that needed human decision. For example a client may be in excellent standing with the loan institution, but no other financial agency will offer a loan.

This process can be automated as long as the players involved have Web services available to each other. BPEL simply needs to “define the relation of such simple activities to each other in order to know how and when to run them. [3]” So BPEL would orchestrate the acceptance of the loan request, would issue a process to send the loan request to a “Web services enabled financial institution, [3]” then wait for the financial institution to reply followed by a creation of the loan acceptance letter back to the client.
6.2 Dividing BPEL into sections to Better Understand

So how do we get BPEL to handle all the simple activities. In order to understand the process, let us divide it into two basic phases, the “Building-of Phase” and the “Execution-of Phase.”

6.2.1 The Building-of Phase

The Building-of Phase includes the design, development, and deployment environment where the companies business analysts together with developers set out to develop the building-blocks for business processes that will be used in the Execution-of Phase [10].

The business analysts and developers use tools that generally support both a graphic and source-code environment. These tools may be as simply as plug-ins for drawing tools or integrated development environments, to fullblown standalone environments [10]. Once such tool is Oracle’s BPEL Process Manager that has an Eclipse or JDeveloper plug-in (See Figure 8) [11].

These tools should also support a service repository such as Subversion [6], to allow collaboration and version control. Let us say, for example, that the business analysts and developers use Oracle’s BPEL Process Manager plug-in in an Eclipse environment. They work individually on their assigned processes on their local machine. When they are ready, they use Eclipse’s SVN plug-in to update the Subversion repository. For a version to be rolled-out for production, the processes would then be migrated from the repository to the run-time process database which will be accessed by the BPEL execution engine.

In her diagram, BPEL Environment Overview [10], Brenda M. Michelson shows the Business Analysts and Integrators/Developers on the left interacting with the Building-of Phase, shown in the diagram as a green box. Within the green box reside the tools, whether graphical or textual, used
to develop the BPEL processes and stored in a repository for version and collaboration control.

6.2.2 The Execution-of Phase

The Execution-Phase includes a BPEL Server that contains the business process definitions from the Building-of Phase, a Web Services framework and the BPEL execution engine. “The BPEL execution engine services collectively provide the process lifecycle requirements (instantiation, communication, dehydration/hydration, correlation, transaction management, compensation, and termination) as spelled out in the specification [10].”

As diagramed in Figure 9, Michelson shows the run-time environment on the right. Clients access the system through the Client Services which will kick off the necessary business process, as in the case of a client requesting a loan.

Other institutions, such as the financial institution, may want to access services defined at the loan institution. Partner Services in Figure 9 is where the other institution business processes access the loan institution services.
7 The Importance of BPEL

BPEL is a language for specifying business process behavior based on Web Services. These processes in BPEL export and import functionality by using Web Service interfaces exclusively. BPEL is an orchestrating language: it sets down exactly how the Web services will cooperate to carry out the overall business process. BPEL is an XML based programming language that is; you write BPEL programs in XML. Because XML wasn’t designed with programmers in mind, the programming results aren’t prime examples of elegance. Fortunately, you rarely need to write in BPEL by hand. Most BPEL programs are written using special graphical editors that let you describe the business process diagrammatically and then automatically generate the corresponding BPEL code.

7.1 The Purpose

*Programming in the large* generally refers to the high-level state transition interactions of a process. BPEL refers to this concept as an Abstract Process. A BPEL Abstract Process represents a set of publicly observable behaviors in a standardized fashion. An Abstract Process includes information such as when to wait for messages, when to send messages, when to compensate for failed transactions. *Programming in the small*, in contrast, deals with short-lived programmatic behavior, often executed as a single transaction and involving access to local logic and resources such as files and databases. BPEL’s development came out of the notion that programming in the large and programming in the small required different types of languages.

There were ten original design goals associated with BPEL:

1. Define business processes that interact with external entities through Web Service operations defined using WSDL 1.1, and that manifest themselves as Web services defined using WSDL 1.1. The interactions are abstract in the sense that the dependence is on port Type definitions, not on port definitions.

2. Define business processes using an XML-based language. Do not define a graphical representation of processes or provide any particular design methodology for processes.
3. Define a set of Web service orchestration concepts that are meant to be used by both the external (abstract) and internal (executable) views of a business process. Such a business process defines the behavior of a single autonomous entity, typically operating in interaction with other similar peer entities. It is recognized that each usage pattern (i.e. abstract view and executable view) will require a few specialized extensions, but these extensions are to be kept to a minimum and tested against requirements such as import/export and conformance checking that link the two usage patterns.

4. Provide both hierarchical and graph-like control regimes, and allow their use to be blended as seamlessly as possible. This should reduce the fragmentation of the process modeling space.

5. Provide data manipulation functions for the simple manipulation of data needed to define process data and control flow.

6. Support an identification mechanism for process instances that allows the definition of instance identifiers at the application message level. Instance identifiers should be defined by partners and may change.

7. Support the implicit creation and termination of process instances as the basic lifecycle mechanism. Advanced lifecycle operations such as “suspend” and “resume” may be added in future releases for enhanced lifecycle management.

8. Define a long-running transaction model that is based on proven techniques like compensation actions and scoping to support failure recovery for parts of long-running business processes.

9. Use Web Services as the model for process decomposition and assembly.

10. Build on Web services standards (approved and proposed) as much as possible in a composable and modular manner.
8 Conclusion

In this paper we have discussed few essential of understanding BPEL such as, the importance BPEL, players in the market, business use of BPEL and how BEPL works. In summery, BPEL basically builds on the foundation of XML and Web services and uses an XML based language that support the Web services stack including, SOAP, WSDL, UDDI and more. It is basically the integration of two workflow languages; Web Serves Flow Language and XLANG.

Some may argue that BPEL is not widely used where it supposed to be used for instance, business analyst or process analyst. However, BPEL has become the de-fac-to standard in the industry. When all major IT vendors including IBM, Microsoft, Oracle and SAP agree on any given standard, not much room is left for others.
References


