Web Services Orchestration and Choreography

Combining Web services to create higher level, cross-organizational business processes requires standards to model the interactions. Several standards are working their way through industry channels and into vendor products.

Chris Peltz
Hewlett-Packard Company

In a recent Computer Sciences Corporation survey,[1] senior information technology executives ranked "connecting to customers, suppliers, or partners electronically" as the top global IT management issue. Web services offer standards-based mechanisms for addressing this issue. However, existing methods for creating business processes are not designed to work with cross-organizational components. Nor are these methods flexible enough to handle the technical interfaces that Web services introduce.

The terms *orchestration* and *choreography* describe two aspects of creating business processes from composite Web services. The two terms overlap somewhat, but Figure 1 illustrates their relationship to each other at a high level. Orchestration refers to an executable business process that can interact with both internal and external Web services. The interactions occur at the message level. They include business logic and task execution order, and they can span applications and organizations to define a long-lived, transactional, multistep process model.

Orchestration always represents control from one party's perspective. This differs from choreography, which is more collaborative and allows each involved party to describe its part in the interaction. Choreography tracks the message sequences among multiple parties and sources—typically the public message exchanges that occur between Web services—rather than a specific business process that a single party executes.

PROCESS DESIGN REQUIREMENTS

Proposed orchestration and choreography standards must meet several technical requirements for designing business processes that involve Web services. These requirements address both the language for describing the process workflow and the supporting infrastructure for running it.

First, asynchronous service invocation is vital to achieving the reliability and scalability that today's IT environments require. The capability to invoke services concurrently can also enhance process performance. Implementing asynchronous Web services requires a mechanism to correlate requests with each other. Software architects commonly use correlation identifiers for this purpose.

The process architecture must also provide a way to manage exceptions and transactional integrity. In addition to handling errors and time-out constraints, orchestrated Web services must ensure resource availability for long-running distributed transactions. Traditional ACID (atomicity, consistency, isolation, and durability) transactions are typically not sufficient for long-running, distributed transactions because they cannot lock resources in a transaction that runs over a long time. The notion of *compensating transactions* offers a way to undo an action if a process or user cancels it. With compensating transactions, each method exposes an undo operation that a transaction coordinator can invoke if necessary.

Web services orchestration must be dynamic, flexible, and adaptable to meet changing business needs. A clear separation between the process logic and the

Published by the IEEE Computer Society
0018-9162/03/$17.00 © 2003 IEEE

Authorized licensed use limited to: SUNY Buffalo. Downloaded on March 30,2010 at 09:28:34 EDT from IEEE Xplore. Restrictions apply.

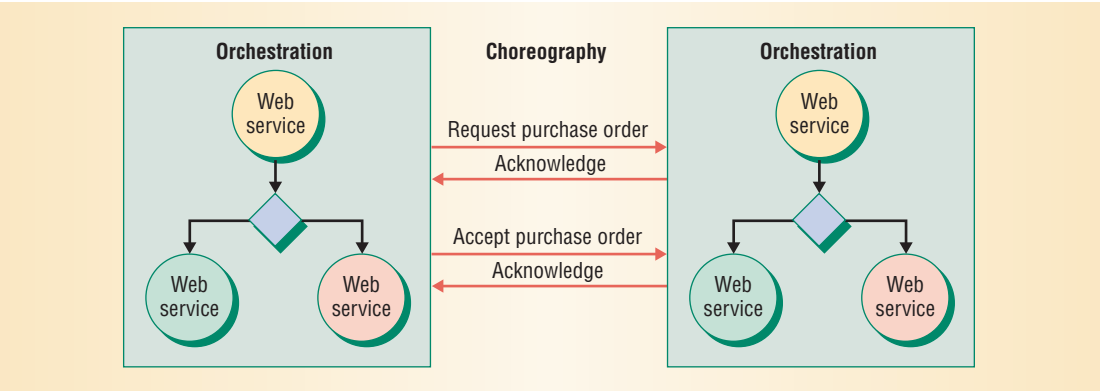

Figure 1. Orchestration versus choreography. Orchestration refers to an executable process. Choreography tracks the message sequences between parties and sources.

Web services used promotes flexibility. An orchestration engine can usually achieve this separation. The engine handles the overall process flow, calling the appropriate Web services and determining what steps to complete. This approach lets an organization swap out services in the overall process flow.

Finally, process designers must be able to compose higher-level services from existing orchestrated processes. Exposing these processes through their own Web service interfaces accomplishes this goal of recursive combination.

EMERGING STANDARDS

Early work to design a business process by combining Web services included Microsoft's XLANG and IBM's Web Services Flow Language (WSFL). The companies later combined these standards to form Business Process Execution Language for Web Services (BPEL4WS), one of the emerging specifications currently available to meet Web services orchestration requirements.

Early work

Microsoft initially developed XLANG to support sequential, parallel, and conditional process flows for its BizTalk Server. XLANG uses the World Wide Web Consortium's (W3C) Web Services Description Language (WSDL, www.w3.org/TR/wsdl12/) to describe Web service interfaces. XLANG focuses on the creation of business processes and the message exchange behaviors among Web services. It also includes a robust exception-handling facility, with support for long-running transactions.

WSFL describes both public and private process flows. It defines the data exchanges as well as the execution sequence (flow models) and the mapping of each step in the flow to specific operations (global models). WSFL exposes a WSDL interface, allowing recursive composition. It supports exception handling but has no direct support for transactions.

UN/CEFACT, the United Nations Center for Trade Facilitation and Electronic Business, developed the Electronic Business using Extensible Markup Language. The ebXML standard provides a suite of middleware components to facilitate collaboration between trading partners. The suite includes the Business Process Specification Schema. The BPSS protocol can define both the choreography and communication protocols between Web-based services.

Hewlett-Packard also developed the Web Services Conversation Language. WSCL outlines a simple standard for modeling the interaction sequences between Web services. HP submitted WSCL as a W3C Note in March 2002; the recently created W3C Web Services Choreography working group is currently considering it.

A variety of specifications and standards have been introduced to address orchestration, but this article focuses on three of the more recent initiatives.

Business Process Execution Language

Microsoft, IBM, Siebel Systems, BEA, and SAP coauthored version 1.1 of the BPEL4WS specification, which they released in May 2003. The specification—called BPEL, for short—models the behavior of Web services in a business process interaction.[2] It provides an XML-based grammar for describing the control logic required to coordinate Web services participating in a process flow. An orchestration engine can then execute this grammar, coordinating activities and compensating the overall process when errors occur.

BPEL is a layer on top of WSDL. The WSDL interface defines the specific operations allowed, and BPEL defines how to sequence them. WSDL describes the public entry and exit points for every BPEL process, and WSDL data types describe the information that passes between process requests. Additionally, WSDL can reference external services that the BPEL process requires.

BPEL provides support for both executable and abstract business processes. An *executable process* models the behavior of participants in a specific business interaction, essentially modeling a private workflow. An *abstract process* or business protocol specifies the public message exchanges between parties. Business protocols are not executable and do not convey a process flow's internal details. Essentially, executable processes model orchestration while abstract processes model the choreography of services.

Authorized licensed use limited to: SUNY Buffalo. Downloaded on March 30,2010 at 09:28:34 EDT from IEEE Xplore. Restrictions apply.

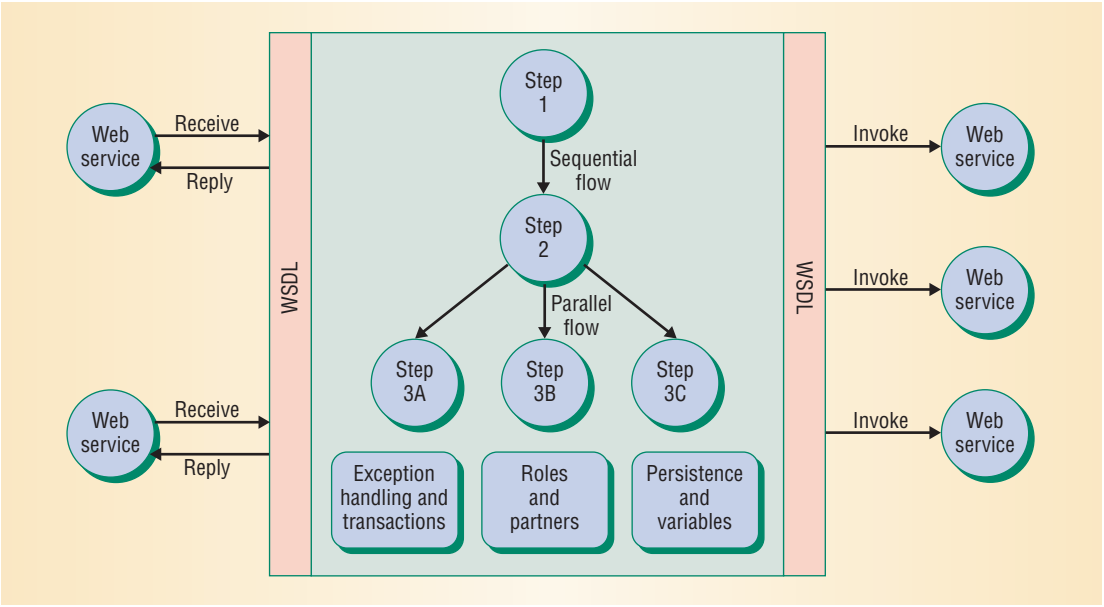

Figure 2. BPEL4WS process flow. The specification supports structured activities to manage the overall process flow as well as basic activities that involve interactions with services external to the process itself.

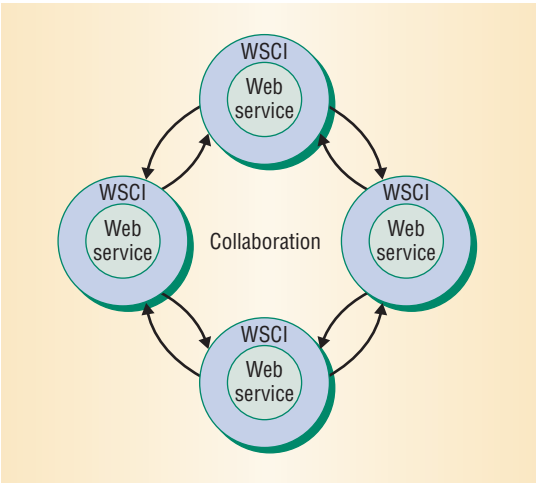

Figure 3. WSCI collaboration. The interface specification addresses only the observable behavior between Web services and not the definition of an executable business process.

The BPEL4WS specification includes support for both basic and structured activities. A *basic activity* is an instruction that interacts with something external to the process itself. For example, basic activities would handle receiving, replying, or invoking Web services. In a typical scenario, a BPEL executable process receives a message. Then it might invoke a series of services to gather additional data and subsequently respond to the requestor. In Figure 2, the receive, reply, and invoke messages all represent basic process activities.

Structured activities manage the overall process flow, specifying the sequence for referenced Web services. These activities also support conditional looping and dynamic branching. They are essentially BPEL's underlying programming logic for BPEL.

Variables and partnerLinks are two other important BPEL elements.

- A *variable* identifies the specific data exchanged in a message flow. When a BPEL process receives a message, it populates the appropriate variable so that subsequent requests can access the data. Variables are used to manage data persistence across Web services requests.
- A *partnerLink* could be any service that the process invokes or any service that invokes the process. Each partnerLink maps to a specific role that it fulfills in the business process.

BPEL also provides a robust mechanism for handling transactions and exceptions, building on top of the WS-Coordination and WS-Transaction specifications. Developed jointly by Microsoft, IBM, and BEA, these corollary specifications provide the support necessary to manage and coordinate business activity operations.

To group a set of activities in a single transaction, BPEL uses a *scope* tag. The tag signifies that the enclosed steps should either all complete or all fail. Within this scope, a developer can specify compensation handlers that the BPEL orchestration engine can invoke in case of error.

Like the Java programming language, BPEL handles exceptions through throw and catch clauses.

Web Services Choreography Interface

Sun, SAP, BEA, and Intalio developed the WSCI specification,[3] which defines a collaboration extension to WSDL. The specification became a W3C Note in August 2002. WSCI defines the overall choreography or message exchange between Web services. The specification supports message correlation, sequencing rules, exception handling, transactions, and dynamic collaboration.

As Figure 3 shows, WSCI describes only the observable behavior between Web services. It does not address the definition of executable business processes as BPEL does. Furthermore, a single WSCI interface describes only one partner's par-

Authorized licensed use limited to: SUNY Buffalo. Downloaded on March 30,2010 at 09:28:34 EDT from IEEE Xplore. Restrictions apply.

ticipation in a message exchange. A WSCI choreography would include a set of WSCI interfaces, one for each partner in the interaction. In WSCI, no single process manages the interaction.

Each WSCI action represents a unit of work that maps to a specific WSDL operation. WSCI extends WSDL, describing how to choreograph the available WSDL operations. In other words, WSDL describes the entry points for each available service, and WSCI describes the interactions among WSDL operations.

WSCI supports both basic and structured activities. An *action* tag defines a basic request or response message. Each activity specifies the WSDL operation involved and the specific participant that performs it. A choreography can then invoke external services through a *call* tag. WSCI supports a wide variety of structured activities, including conditional looping and both sequential and parallel processing.

The following WSCI interface creates a purchasing process that contains two sequential activities, Receive Order and Confirm. Each activity maps to a WSDL portType, and WSCI establishes a correlation between them:

```
<process name="Purchase"
     instantiation="message">
   <sequence>
      <action name="ReceiveOrder"
           role="Agent" operation=
           "tns:Order">
      </action>
      <action name="Confirm"
           role="Agent" operation=
           "tns:Confirm">
         <correlate correlation=
           "tns:ordered"/>
         <call process="tns:
           Purchase"/>
      </action>
   </sequence>
</process>
```

WSCI supports both business transactions and exception handling. A process designer can set up specific transactional contexts within this WSCI interface, similar to the scope activity in BPEL4WS. The context defines a set of activities for which any failure will roll back the entire group.

Business Process Management Language

BPML is an XML-based language for describing business processes. Business Process Management Initiative (www.BPMI.org)—a nonprofit corporation chartered by Intalio, Sterling Commerce, Sun, and CSC—developed the specification. Initially, BPML was designed to support processes that a business process management system could execute. The BPML Last Call Draft, however, released in November 2002, also incorporated WSCI support. BPML and WSCI share the same underlying process execution model, so developers can use WSCI to describe public interactions among business processes and reserve BPML for developing private implementations.

BPML provides process flow constructs and activities similar to BPEL.[4] Basic activities for sending, receiving, and invoking services are available, along with structured activities for handling conditional choices, sequential and parallel activities, joins, and looping. BPML also lets a business process developer schedule tasks to run at specific times.

The language was designed to manage long-lived processes and so includes features to support persistence. XML exchanges occur between participants, using roles and partner definitions similar to the BPEL constructs. BPML also supports recursive composition to build aggregate business processes from subprocesses.

BPML supports both short- and long-running transactions, using a scoping technique similar to BPEL to manage the compensation rules. Process designers can nest processes and transactions. Finally, the language includes a robust exception-handling mechanism.

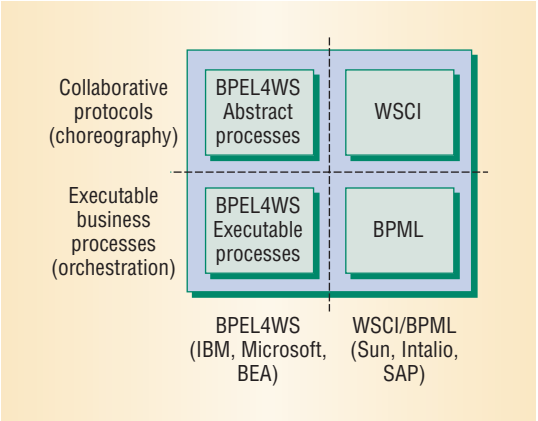

Figure 4. Relationships among orchestration and choreography standards: Business Process Execution Language for Web Services, Web Services Choreography Interface, and Business Process Management Language.

ORCHESTRATION AND CHOREOGRAPHED COLLABORATION

BPEL, WSCI, and BPML all take somewhat different approaches to orchestration and choreography. Both BPEL and BPML provide capabilities to define an executable business process, whereas WSCI's approach is more choreographed and collaborative—requiring each message-exchange participant to define a WSCI interface.

Figure 4 illustrates one way to categorize these specifications and standards. The *y*-axis distin-

Authorized licensed use limited to: SUNY Buffalo. Downloaded on March 30,2010 at 09:28:34 EDT from IEEE Xplore. Restrictions apply.

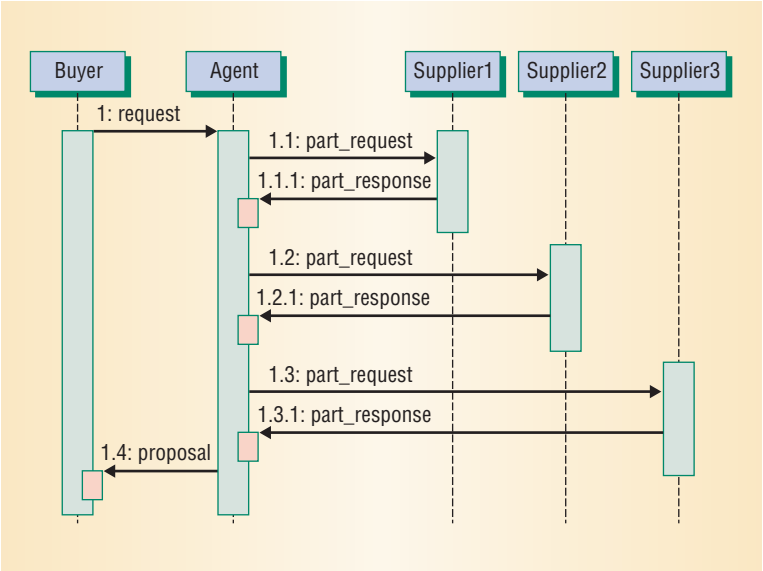

Figure 5. Case study sequence diagram. The manufacturer's buyer issues a request for proposal to a purchasing agent. The agent issues component requests to individual suppliers and composes the proposal from them.

guishes between collaborative protocols and executable business processes. A collaborative protocol refers to choreographed message exchanges between multiple business parties, while executable processes are private workflows controlled by a single entity. The *x*-axis shows the two emerging initiatives in the overall space: BPEL and WSCI/BPML considered together. BPEL supports both executable and collaborative processes, while BPML and WSCI can work together so that BPML models the execution of a business process while WSCI models the choreography among Web services.

CASE STUDY

To illustrate the concepts for orchestrating Web services, consider a purchasing system scenario in which a manufacturer wants to build a product configuration using a list of available suppliers. The manufacturer's buyer works through a purchasing agent to fulfill the inventory requirements. The agent communicates with suppliers that offer the required components. After identifying the suppliers for a complete configuration, the purchasing agent sends the buyer a proposal. The buyer then either orders the parts or cancels the request. Figure 5 shows a simplified sequence diagram for the process.

Each participant—buyer, agent, and supplier—has a WSDL description of the exposed input and output interfaces. This case study demonstrates the purchasing agent's perspective on the workflow and the public interfaces it exposes. In addition to accepting requests from the buyer and sending requests to suppliers, the purchasing agent must handle transactional integrity for the overall process. For example, if a supplier cannot fulfill a request because of low inventory or network problems, the agent must manage the process appropriately, either rolling back the request or providing some mechanism for placing an incomplete order.

Since it describes an orchestrated process, this case study could use either BPEL or BPML. However, I selected BPEL because it includes low-cost tools and technical tutorials that appear to cater more to developers.

The first step in documenting a BPEL process is to define it. This starts with a root-level <process> tag that names the process and lists its XML namespace references.

Next, BPEL defines the process partners and their roles. In BPEL, the <partnerLinks> and <partnerLink> tags can implement the three basic roles: buyer, purchasing agent, and supplier. A WSDL document defines the <partnerLinkType> tag that <partnerLink> references.

The following code sample defines a partner link between the buyer and the purchasing agent:

```
<partnerLinks>
   <partnerLink name="Buyer"
        partnerLinkType=
        "po:requestQuoteLinkType"
      myRole="Purchaser"
        partnerRole="Requestor"/>
   <partnerLink name="Supplier1"
        partnerLinkType=
        "po:requestQuoteLinkType"
      myRole="Requestor"
        partnerRole="Purchaser"/>
   <!—Set up other suppliers used in
        this process —>
</partnerLinks>
```

The purchasing agent's perspective requires only the buyer and supplier partner relationships to be defined. When the buyer interacts with the agent, the buyer is the requestor and the purchasing agent acts as the receiver. The roles are reversed when the agent interacts with one of the suppliers.

The process must also manage the information flow between partners. BPEL uses variables to persist process information. In this scenario, the buyer makes an initial purchase request; the agent then constructs quote requests for each supplier with a part number and quantity, and the suppliers respond with pricing information. The agent then constructs a proposal for the buyer.

Potentially four variables model this interaction:

```
<variables>
   <variable name="request"
      messageType="po:request"/>
   <variable name="part_request
      messageType="po:part_request"/>
   <variable name="part_quote"
```


Authorized licensed use limited to: SUNY Buffalo. Downloaded on March 30,2010 at 09:28:34 EDT from IEEE Xplore. Restrictions apply.

```
      messageType="po:part_quote"/>
   <variable name="proposal"
      messageType="po:proposal"/>
</variables>
```

The process must also correlate the requests with each other. For example, we can assign a unique identifier to each quote and to the final proposal. BPEL documents these identifiers using the <correlationSet> tag:

```
<correlationSets>
   <correlationSet name="Quote"
      properties="cor:quoteID"/>
   <correlationSet name="Proposal"
      properties="cor:proposalID"/>
</correlationSets>
```

A key part of the BPEL document defines the steps required to handle the request. The <sequence> tag runs activities sequentially; the <flow> tag runs them in parallel; and the <receive>, <reply>, and <invoke> tags define the basic activities required to interact with the Web services using WSDL.

```
<sequence>
   <receive name="receive" partnerLink=
         "Buyer" operation="request"
         variable="request"
         initiate="yes">
   </receive>
   <flow name="supplier_flow">
      <invoke name="quote_supplier1"
         partnerLink="Supplier1"
         operation="request_quote"
         inputVariable = "part_request"
         outputVariable="part_quote">
      </invoke>
      <!— invoke other suppliers as
         part of the process, done in
         parallel —>
   </flow>
   <reply name="reply" partnerLink=
         "Buyer" operation=
         "send_proposal variable=
         "proposal">
   </reply>
</sequence>
```

The sequence begins upon receipt of the buyer's request. The <flow> tag executes a parallel set of activities to contact each supplier for part quotes. Each activity references a specific WSDL operation and uses the available variables for input and output. Upon receiving the supplier responses, the purchasing agent constructs a message replying to the buyer. The agent could use the BPEL <assign> tag and the W3C's XPath language for addressing parts of an XML document to take the suppliers' containers and build a final proposal back to the buyer.

The final step is managing exceptions in the scenario. For example, if an error occurs in contacting a supplier, the agent may want to send a message back to the buyer. BPEL4WS includes fault handlers for these error conditions.

```
<faultHandlers>
   <catch faultName=
         "cantFulfillRequest">
      <invoke partnerLink="buyer"
         operation="sendError"
         inputVariable="fault"/>
   </catch>
</faultHandlers>
```

The process may also require compensation handlers. For example, if the agent cannot contact a supplier, the process must offer a way to roll back the order. The BPEL <scope> tag sets a transactional context by grouping related activities together. In this scenario, the three parallel invocations to the suppliers might be a good candidate for a scope declaration.

The completed process needs an orchestration engine for deployment. Both IBM alphaWorks (www.alphaworks.ibm.com) and Collaxa (www.collaxa.com) offer good development tools for designing, deploying, and running business processes built with BPEL and WSDL.

FUTURE INITIATIVES

While BPEL4WS, WSCI, and BPML work their way through standards processes and into vendor product implementations, other enhancements and issues relevant to Web services orchestration are emerging.

IBM researchers have proposed a peer-to-peer model of e-business interaction.[5] They compare current Web services to a vending machine—a set number of buttons that can be pressed in a predefined order. They propose a conversational model—more like a telephone call with flexible, dynamic exchanges between the parties at each end. At this time, IBM's Conversation Support for Web Services is the only proposal that claims to support this capability.[6]

Security is also a concern because orchestration exposes Web services interfaces on HTTP, which can present a number of security challenges. The software industry is discussing several Web services

Authorized licensed use limited to: SUNY Buffalo. Downloaded on March 30,2010 at 09:28:34 EDT from IEEE Xplore. Restrictions apply.

security standards, including XML Digital Signatures and Encryption, Security Assertion Markup Language, and WS-Security. These standards are intended to fill specific Web services requirements for authentication, authorization, and message-level security. The orchestration and choreography standards presented here do not offer direct support for security.

The other area receiving a great deal of attention is Web services management. Organizations need an end-to-end management infrastructure—for applications, systems, and networks—that gives them flexible control over business processes involving Web services, including control of specific steps in the process. A robust management infrastructure must both monitor the environment and provide capabilities to adapt and optimize it in real time. The need for this type of management will become increasingly important as Web services are combined, aggregated, and orchestrated to form more meaningful business processes. Hewlett-Packard has submitted an open, extensive Web services management framework (http://devresource.hp.com/drc/specifications/wsmf/WSMF-WSM.jsp) to the industry consortium Oasis (www.oasis-open.org) to address this specific need.

Computer Wants You

Computer **is always looking for interesting editorial content. In addition to our theme articles, we have other feature sections such as Perspectives, Computing Practices, and Research Features as well as numerous columns to which you can contribute. Check out our author guidelines at**

http://computer.org/computer/author.htm

for more information about how to contribute to your magazine.

Innovative Technology for Computer Professionals
Computer

BPEL4WS has major companies supporting it, including Microsoft, IBM, and BEA. Moreover, its supporters submitted the specification to Oasis in April 2003, further broadening its support. Sun, Intalio, and SAP initially submitted the WSCI specification to the W3C, which has assigned it to the Web Services Choreography working group. While the Oasis BPEL technical committee is focused on standardizing the BPEL4WS specification, WS-Choreography is chartered with defining a choreography language.

While the industry appears to be embracing the BPEL initiative, it is still unclear what part WSCI and WS-Choreography will play. Vendor backing and tools support will clearly influence the software industry's adoption rate. ■

References

1. Computer Sciences Corp., "13th Ann. Critical Issues of IS Management Survey," 2000; www.csc.com/survey.
2. S. Weerawarana and C. Francisco, "Business Process with BPEL4WS: Understanding BPEL4WS, Part 1," research report, *IBM developerWorks*, Aug. 2002; www-106.ibm.com/developerworks/webservices/library/ws-bpelcol1/.
3. A. Arkin et al., *Web Service Choreography Interface 1.0*, 2002; www.sun.com/software/xml/developers/wsci/wsci-spec-10.pdf.
4. R. Shapiro, "A Comparison of XPDL, BPML, and BPEL4WS," draft document, Cape Visions, May 2002; http://xml.coverpages.org/Shapiro-XPDL.pdf.
5. J.E. Hanson, P. Nandi, and D. Levine, "Conversation-Enabled Web Services for Agents and e-Business," *Proc. Int'l Conf. Internet Computing*, Computer Science Research, Education, and Applications (CSREA) Press, 2002, pp. 791-796.
6. S. Kumaran and P. Nandi, "Conversational Support for Web Services: The Next Stage of Web Services Abstraction," research report, *IBM developerWorks*, Sept. 2002; www-106.ibm.developerworks/webservices/library/ws-conver/.

Chris Peltz *is a senior software consultant in Hewlett-Packard's Developer Resources Organization (devresource.hp.com), where he provides technical and architecture consulting to customers exploring J2EE and Web-based solutions. His current research addresses several emerging Web services trends, including Web services orchestration and Web services management. Peltz received an ME in engineering management from the University of Colorado. Contact him at chris.peltz@hp.com.*

Authorized licensed use limited to: SUNY Buffalo. Downloaded on March 30,2010 at 09:28:34 EDT from IEEE Xplore. Restrictions apply.