# Web Services: Technical Evolution yet Practical Revolution?

Amit Sheth and John A. Miller,
*University of Georgia*

Web services are here to stay. Both business and technical considerations will provide them with the staying power and tailwind to survive the hype. Despite their lack of a major technical breakthrough, Web services will initially succeed owing to the right confluence of evolutionary technological choices, low barriers and entry costs, and their strong standards-based approach. In the long run, however, they must be empowered by semantics and coupled with Semantic Web technologies to fulfill broader, longer-term promises.

## Evolution in the right direction: Reasons for likely success

Progress in software componentry has been ongoing. The Web is becoming more than just a collection of documents; applications and services are coming to the forefront. History shows that incremental technological progress can produce dramatic effects, such as the advancement from FTP to Archie to Veronica to the Web. Web services might have a similar fate.

Several technical and technology-driven business advantages present themselves, boding well for a quicker and more successful adoption of Web services.

### Low initial complexity

The first consideration is the low initial complexity (although the complexity of fully mature technology might be substantially higher), with low entry barriers and costs. Although the Web now consists of many components and technologies—several of which were added as the Web gained wider acceptance and matured—it indeed started with a remarkably small, simple core technology. That simple core—free availability and low- or no-cost initial adoption—played an important role in its growth and acceptance.

Unlike the Web, the Corba architecture was complex from the start (considering its various services and facilities) and required seasoned developers to use it. Also, most businesses had to rely on object request brokers that were not free, which made starting pilot projects—something large businesses always do before adopting any new technologies—difficult.

However, like the Web, the basic components of Web services are relatively simple, and the learning curve is low. XML is already known to the likely adopters and accepted as the data exchange standard by enterprises. Standards such as SOAP and WSDL are simple to use and learn. (For an explanation of these and other acronyms, see the "Glossary" sidebar.) Although UDDI as a whole (with all the details of white, yellow, and green pages) is not simple, the basic parts needed to get started are relatively easy to learn and use. Also, Web services can be developed with essentially no initial technology cost. Furthermore, just as the Web is used not only for publishing or sharing data but also as an application infrastructure, the complexity of Web services can be incrementally increased as more functionality is added.

### The hype

The second consideration is the hype surrounding Web services, and the industry's ability to sustain interest over time. Consider XML. Although it has been significantly hyped—for example, it only addresses the syntactic interoperability issues, not semantic ones—it has succeeded. This is because it solves a limited yet important and well-understood business problem, and there is a groundswell of commercialization efforts around the use of XML technology. Some enterprise software such as Enterprise Resource Planning quickly adopted XML, broadening its adoption by businesses. In comparison, corresponding commercial activities did not support the hype of expert systems and other artificial intelligence technologies.

In the case of Web services, various software segments, including application servers and Enterprise Content Management,[1] have found them worthy enough to adopt. This gives the hype of Web services an ability to sustain with follow-through development and adoption.

### Standards

The third consideration is the use of more widely accepted standards. XML already enjoys wide adoption in enterprises. Standardization efforts are well aligned and are structured to act much faster than what happened with Corba. Some in industry also see Web services as a natural follow-up to Electronic Data Interchange and ebXML, which already have significant commercial success.[2]

### Loosely coupled architectures

The fourth consideration is that Web services are a good choice for loosely coupled architectures.[3] This means not allowing object references or requiring statefulness. Although the Corba architecture was more suitable for intra-enterprise environments, the technical features and choices of Web services make them more reusable and thus more appropriate for interenterprise and global environments.

### Languages

Finally, Web services do not use languages that look like programming languages (that is, there's no interface definition language). This is what the industry is trying to accomplish with Web services.

## Incremental advances in software componentary

The introduction of the object-oriented programming paradigm in the 1980s promised to make reusable software components a reality. Although there were certainly some gains in software reuse and programming productivity, the change was not substantial. The 1990s saw further attempts to increase software reuse, focused on the idea of software components and extended to distributed components. A notable effort in this area was Corba. It let components on different machines, using different operating systems and even different languages, communicate. From one viewpoint, Web services represent an evolution of Corba. We contend that Web services are more convenient than Corba and that eventually incremental improvement will lead to widespread use.

One difficulty in using distributed technology such as Corba is dealing with languages. Corba tried to eliminate this problem by moving the language into the background by defining an IDL. Although this is a good idea, an IDL looks much like C++, and developers must also understand language bindings to use Corba. If there was a way to replace language-like IDLs with higher-level specification, surely communication could be simplified using meaningful messages. Fortunately, Web services, defined as "self contained, self-describing modular applications that can be published, located, and invoked across the Web,"[4] automatically get the Web-wide scope rather than primarily the enterprise-wide scope (at least initially) for Corba. In the near term, we see the use of a few well-targeted Web services within and across enterprises for well-targeted B2B applications and as integration points between enterprise software. Examples include integrating a document management system and portal management system within an enterprise, or interenterprise interfaces between different parts of a supply chain or ERP solution, which already uses XML for data exchange.

## Intermediate to long term: Processes, QoS, security, and semantics

For the intermediate term, we predict distinct advances in

- Web processes
- Technical support for quality of service
- Technical support for security

Although we expect researchers to address QoS and security in the intermediate term, we expect support for Web processes to take longer, because initial research is just starting on these topics.[5] Because Web services are components, if individual services are limited in their capability, we can compose existing Web services to create new functionality in the form of Web processes. Web service composition is the ability to take existing services (or building blocks) and combine them to form new services.[6] In carrying out this composition task, we should be concerned about the efficiency and the QoS that the composed process will exhibit when it executes. This task of composing services to create efficient Web processes is analogous to designing a workflow.

Web service composition is an active area of research, with academic and industrial research groups proposing many languages. IBM's WSFL[7] and Microsoft's XLANG[8] were two of the earliest languages to define standards for Web services composition. Both extended WSDL, W3C's standard language used to describe the syntactic aspects of a Web service. BPEL4WS[9] is a recently proposed specification that represents the merging of WSFL and XLANG. It combines WSFL's graph-oriented process representation and XLANG's structural construct-based processes into a unified standard for Web services composition. Another effort in this area is ebXML, which lets enterprises conduct business over the Internet using an open XML-based infrastructure.

In contrast to these commercial XML-based standards, researchers are developing a unique Web service markup language called DAML-S.[10] According to the group of researchers working on DAML-S, it supplies Web service providers with a core set of markup language constructs for describing the properties and capabilities of their Web services in unambiguous, computer-interpretable form. DAML-S markup of Web services will facilitate the automation of Web service tasks including automated Web service discovery, execution, interoperation, composition and execution monitoring.[10]

If Web services are to provide not only an enterprise-wide but also a global architecture for application interoperability and integration, we will need to enhance Web services and processes with semantics. Many applications will require a Web process created from composing several Web services. In the intermediate term, we can create enterprise-scale Web processes (or service compositions) by adopting workflow management technology. However, to achieve Web services' full Web-wide and global potential, semantics holds the trump card. Using semantics involves describing resources with formal, machine-readable description. Resources in this case are Web services, and given that they wrap applications, they must be described both functionally and operationally.

As already recognized by DAML-S,[10] Web Service Modeling Framework initiatives,[11] the Work Group on Web Services at a recent Semantic Web workshop,[12] and others, semantics can play a critical role in developing Semantic Web services. This can lead to better discovery of Web services for reuse in a global, Web-scale environment that is not limited to one enterprise or a static collection of enterprises and where current syntax-based techniques do not work. Strong support for Web services discovery is also a prerequisite to developing Web processes[13,14] and addressing various issues of composition, interoperability, and execution.[15,16] There is demonstrable progress in developing semantics-based solutions when dealing with data, such as to achieve better interoperability and integration of information resources. However, the challenges of dealing with applications that are enabled by Web services are even more difficult. They will involve substantial further research and engineering that consider both functional and operational perspectives. The Large Scale Distributed Information Systems Lab's Meteor project is one of the projects looking at developing semantics-based solutions to QoS, discovery, and composition issues in the context of Web processes (see http://lsdis.cs.uga.edu/proj/meteor/SWP.htm).

## References

1. R. Perry and R. Lancaster, *Enterprise Content Management: Expected Revolution or Vendor Positioning*, The Yankee Group, Boston, 2002.
2. A. Kotok, " The E-Business Continuum: Web Services, ebXML and EDI," WebServices.Org, June 2002, www.webservices.org/index.php/article/articleview/479/1/24.
3. D. Austin et al., "Web Services Architecture Requirements," World Wide Web Consortium, 2002, www.w3c.org/TR/wsa-reqs.
4. D. Tidwell, "Web Services—The Web's Next Revolution," IBM tutorial, 29 Nov. 2000, www-105.ibm.com/developerworks/education.nsf/webservices-onlinecourse-bytitle/BA84142372686CFB862569A400601C18?OpenDocument.
5. J. Cardoso and A. Sheth, *Semantic e-Workflow Composition*, tech. report, Large Scale Distributed Information Systems Lab, Dept. of Computer Science, Univ. of Georgia, Athens, Ga., 2002.
6. G. Piccinelli, *Service Provision and Composition in Virtual Business Communities*, tech. report HPL-1999-84, Hewlett-Packard, Palo Alto, Calif., 1999; www.hpl.hp.com/techreports/1999/HPL-1999-84.html.
7. F. Leymann, "Web Service Flow Language (WSFL) 1.0," IBM, Armonk, N.Y., 2001, www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf.
8. S. Thatte, "XLANG: Web Services for Business Process Design," 2001, www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm.
9. F. Curbera et al., "Business Process Execution Language for Web Services," 2002, www-106.ibm.com/developerworks/webservices/library/ws-bpel.
10. A. Ankolekar et al., "DAML-S: Web Service Description for the Semantic Web," *Proc. Int'l Semantic Web Conf.*, Springer-Verlag, Berlin/Heidelberg, 2002, pp. 348–363.
11. D. Fensel and C. Bussler, "The Web Service Modeling Framework WSMF," 2002, http://informatik.uibk.ac.at/users/c70385/wese.

12. A. Sheth and R. Meersman, “Amicalola Report: Database and Information Systems Research Challenges and Opportunities in Semantic Web and Enterprises,” *ACM SIGMOD Record*, vol. 31, no. 4, Dec. 2002.

13. S. Narayanan and S. Mcllraith, “Simulation, Verification and Automated Composition of Web Services,” *Proc. 11th Int’l World Wide Web Conf.*, W3C, 2002, pp. 77–88.

14. J. Cardoso et al., *Modeling Quality of Service for Workflows and Web Service Processes*, tech. report, Large Scale Distributed Information Systems Lab, Dept. of Computer Science, Univ. of Georgia, Athens, Ga., 2002.

15. J. Cardoso et al., “Semantic Web Services and Processes: Semantic Composition and Quality of Service,” tutorial at Federated Conferences (CooPIS, DOA, ODBASE), 2002; http://lsdis.cs.uga.edu/lib/presentations/SWSP-tutorial-resource.htm.

16. S. Chadrasekaran et al., *Composition Performance Analysis and Simulation of Web Services*, tech. report, Large Scale Distributed Information Systems Lab, Dept. of Computer Science, Univ. of Georgia, Athens, Ga., 2002.

## Web Services: *Quo Vadis*?

Christoph Bussler, *Oracle Corporation*
Alexander Maedche, *FZI Research Center for Information Technologies at the University of Karlsruhe*
Dieter Fensel, *Leopold Franzens Universität Innsbruck*

Web services have been touted for some time now as the technology-based silver bullet solution for many significant integration problems in information technology. These integration problems are of so different a nature due to the specific required functionality that we can’t avoid asking right now, “Web services: *Quo vadis*?”

Here, we illustrate the immense discrepancy between the magnitude of the integration problems and the simplistic functionality that Web services provide. We propose the WSMF as the future direction of Web services so that they can cope even with the most real complex integration problems. (For an explanation of this and other acronyms, see the “Glossary” sidebar.)

### Web services at the crossroads

Let’s spotlight the state of the art of Web services and the situation of the “movement” (or “hype,” as some might be more than happy to state). For the following reasons, we see Web services as being at the crossroads:

- They have been touted as the silver bullet for a (too) wide array of integration problems, currently being addressed by complex integration solutions such as B2B integration technology1 as well as simple technology such as Java for remote server invocation.
- Many competing, conflicting, and overlapping standards exist that address Web service functionality (see www.oasis-open.org/cover/sgml-xml.html).
- Many research projects are repackaging existing work as Web services work.
- Many start-up companies, including Cape Clear (www.capeclear.com), Actional (www.actional.com), and Intalio (www.intalio.com), provide technology implementing Web services technology.
- Major infrastructure companies (such as BEA, IBM, Microsoft, and Oracle) already provide implementations of Web service technology.
- Broad journalistic coverage of Web services exists in the trade press, such as *eAI Journal* (www.eaijournal.com) and ebizQ (www.ebizq.net).
- Web services have no underlying real conceptual integration model and are only defined as an implementation technology, such as SOAP (see www.w3.org/TR/soap12-part1/ and www.w3.org/TR/soap12-part2).
- No one has analyzed available and proven solutions such as EDI translation technology[2] or reported on lessons learned from deployments that could advance Web services technology functionality (more than 300,000 Electronic Data Interchange deployments exist worldwide—see www.disa.org/x12org/about/faqs.cfm).

This situation cannot continue indefinitely. We must consolidate the Web services space, focusing Web services technology on a well-defined and well-manageable set of integration problems. This includes the consolidation of the multitude of Web service standard proposals. Current Web services expectations are too high and too varied for one technology to solve all the given integration problems.

### The silver bullet problems

Here we discuss some of the silver bullet problems, clearly showing the vast spectrum that Web services are said to solve.

First, SOAP was originally the acronym for the Simple Object Access Protocol. *Nomen est omen* (the term reveals its intent): the underlying paradigm follows the client-server model. The SOAP specification in conjunction with WSDL (see www.w3.org/TR/wsdl) clearly enables the definition of the external interface of a server in a particular way but not of its clients, violating the general peer-to-peer approach that most integration solutions require. The only pattern this supports is the one-way invocation with and without results between the defined roles of client and server.

In this sense, SOAP, in conjunction with WSDL, clearly implements yet another RPC model. This is also explicitly called out in the SOAP specification. Because it is based on the ubiquitous HTTP protocol in conjunction with XML as the message syntax, computer system boundaries are easy to overcome and the notion of the “better RPC” or even “better distributed-object-management technology” was born, leading to the wide awareness of Web services in the developer community.

Second, because SOAP messages can be transported over HTTP to implement the runtime invocation, bridging computer system boundaries is clearly easy owing to the commodity of the HTTP protocol. If only packaged applications like Enterprise Resource Management systems, such as Oracle or SAP (www.sap.com), were to expose WSDL interfaces, then integrating them with Web services would be easy, fast, cheap, manageable, and so on (better along all possible dimensions).

In such a scenario, Web services would be the “better Enterprise Application Integration solution,” overcoming the current costly integration technology deployments. This is because the whole integration world would be a nice homogeneous sea of SOAP messages implementing WSDL operation invocations.

Finally, because we can easily exchange SOAP messages over the Internet, the proposal of using Web services as the only and superior way to implement B2B interactions between companies (interenterprise B2B communication) is not far off. The vision in this case is to very rapidly replace EDI, SWIFT (Society for Worldwide Interbank Financial Telecommuication, www.swift.com), and other existing B2B standards