Web Services: The Next Dimension in e-Business Computing

Billy Lim[1]

[1]Applied Computer Science Department
Illinois State University
Normal, IL 61790-5150, USA

Abstract

For many years, software reuse and systems interoperability have been primary goals of many information technology organizations, especially those that rely heavily on computer networks. These organizations have software applications that use the Internet to transfer data and conduct business transactions. Object-oriented (OO) technology has been utilized to accomplish these goals with relative success over the years. But there are many hurdles that OO technology could not overcome. One of them is due to lack of standards. An object (software component) developed in one vendor's technology cannot easily communicate with another vendor's. Another difficulty is due to the fact that the majority of software applications reside behind firewalls – security barriers that restrict communication between networks.

Web Service, a self-describing service that can easily be consumed over the Web, is the latest trend in the industry to address the problems identified above. It is based on a technology called SOAP, which in turn is based on XML, a technology that is widely known as the language of the next-generation Web. Through XML and HTTP, both international standards and the latter being a firewall-friendly protocol that everyone uses to surf the Web, it is now possible for involved parties across networks to communicate and produce/consume a service in a uniform manner.

This article describes the brand new technologies behind Web services and the opportunities of adopting the burgeoning technologies in IT organizations. The three major underlying technologies behind Web services are detailed. Lastly, the various issues and challenges that must be addressed by the stakeholders are also discussed.

1. Introduction

The computer industry has advanced tremendously in the past several decades and has impacted our lives greatly in its advancement. Many would agree that when comparing the progress of hardware versus software, the former has advanced more noticeably than the latter. This is partly because there has been lack of standards for developing software and thus it is extremely difficult to integrate disparate systems to form a larger one to solve an organization's problem.

For many years, software reuse and systems interoperability have been primary goals of many IT (information technology) organizations, especially those that rely heavily on computer networks. These organizations have software applications that use the Internet to transfer data and conduct business transactions. Object-oriented (OO) technology, which became popular in the 80's and mainstream in the 90's, has been utilized to accomplish these goals with relative success over the years. Nevertheless, there are many hurdles that OO technology could not overcome.

One of them is due to the aforementioned lack of standards. An object (software component) developed in one vendor's technology cannot easily communicate with another vendor's. To help visualize the magnitude of this problem, think of a VCR and a TV that simply cannot be hooked up because the vendors choose to use different connection schemes. (That, fortunately, is not the case in the real world or they will be more VCRs that are showing 12:00 on the panel!) Another difficulty is due to the fact that the majority of software applications reside behind firewalls – security barriers that restrict communication between networks. Here, even if two systems use the same protocol to communicate, the security of firewall prevents the communication to take place.

Web Service [1, 2, 3, 4, 5, 6, 7] is the latest buzzword in the industry right now to address the problems identified above. It is based on a technology that is in the final stages of approval by the W3C (World Wide Web Consortium), the international standard body that oversees all Web related technologies. It also has strong support from major players such as Microsoft, IBM, Sun Microsystems, Hewlett-Packard, and Oracle. As such, it is projected to be a strong technology that many IT organizations will investigate and adopt if proven viable. In fact, Gartner's Group VP Daryl Plummer compared Web Services with previous attempts and stated that this time things may be different because "With Web Services, all the major vendors are on board with their support."

In a nutshell, Web Services can simply be thought of as self-describing services that are HTTP (Hypertext Transfer Protocol)-addressable. This means that one can shop for a software service much like one can shop for goods on the Web, using exactly the same protocol. Web Services relies on SOAP (Simple Object Access Protocol), WSDL (Web Service Description Language), and UDDI (Universal Description, Discovery, and Integration) as the underlying technologies for involved parties to communicate and produce/consume a Web service, as shown in Figure 1. Here, a scenario that shows a brokerage house registering its stock quote service with a registry and a financial software finding and consuming the service is depicted.

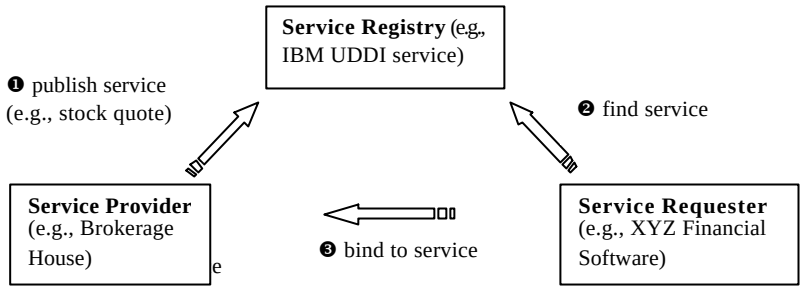

Figure 1: Life Cycle of a Web Service Execution (Registry, Lookup, and Consumption)

Behind the scene, SOAP uses HTTP as the protocol to transmit its message, which is in XML (eXtensible Markup Language) format. This powerful combination of HTTP and XML, both standards of W3C, provides a fully extensible mode of communication between software systems. The interoperability and scalability of Web Services means that developers can rapidly create large applications and larger Web Service from smaller ones. This adds another dimension to the Web; instead of just person-to-person or person-to-system, it also handles system-to-system.

Given that HTTP is a firewall friendly protocol and XML is becoming more and more popular as a standard for data exchange, it is not surprising that Web Services has received so much attention in the industry, including a brand new Web Services Journal for the coverage of the topic [8]. It is the buzz that all major IT corporations are talking about and it is also the centerpiece of the .NET campaign of Microsoft Corporation [9], a technology that Microsoft is staking its future in, with $4 billion initial investment.

This new way of application development is mimicking how hardware vendors have been producing hardware components for years. Now, the software vendors even have the Web, one of the most important revolutions in the computer industry, on their side. As stated by Bill Gates, Chairman and Chief Software Architect of Microsoft, "The power of the XML Web Services model is amazing. A company offering an online electronic-payment service can expose its service to partners, so that they can deliver it as part of their own offering – regardless of what platform they are using. An airline can link its online reservation system to that of a car-rental partner, so travelers can book a car at the same time they book a flight. An online auction company can notify bidders when they are outbid or have won an auction, or could partner with other firms to offer alternative shipping, fulfillment or payment options. XML Web services help your business break free of its boundaries." [9]

Recent studies on Web services have also shown the growth and acceptance of the technology. According to ZapThink, a market research firm, the market for Web Services platforms, application development suites, and management tools is expanding from a $ 380 million (US) market in 2001 to over $ 15.5 billion (US). Also, recent survey by market research firm TechMatrix (of 450 IT professionals and consultants) finds that 65% of small and midsize companies and 35% of large companies have adopted Web services to automate business processes between trading partners and for internal application integration.

2. Web Services: Technical Underpinnings

Web Services is the next chapter in solving the distributed computing challenge. Earlier attempts at this were proprietary solutions from Microsoft, Sun Microsystems, and the Object Management Group, an industry consortium. These attempts include the Distributed Component Object Model (DCOM), Remote Method Invocation (RMI), and the Common Object Request Broker Architecture (CORBA). All these suffer the drawbacks of being proprietary, excessively complex, or unsuited to connect applications over the Internet where the requirement is for 'loosely coupled' versus the 'tightly coupled' systems, the former being one that is more appropriate to link applications that reside behind the firewall of a single organization.

The technical underpinnings of Web Services aim to reinforce the philosophy of being 'standard' by adopting the building block approach of using prior Internet protocols and standards as components of Web Services. As given in Section 1, the building blocks include HTTP, adopted as the transport protocol, and XML, used as the message format

of the messages that are transferred between co-operating applications. The other pieces of the technical puzzles are described below.

SOAP

The Web Services communications protocol, Simple Object Access Protocol (SOAP), is an adaptation of XML-RPC and is what facilitates the machine-to-machine communication nature of Web Services. Using existing protocols that can easily traverse firewalls, such as HTTP, means that many of the problems that exist when attempting to stitch together applications that reside in separate organizations behind firewalls are avoided.

SOAP arises from the realization that no matter how ingenious the current middleware offerings are, they need a WAN wrapper. Architecturally, sending messages as plain XML has advantages in terms of ensuring interoperability. The middleware players seem willing to put up with the costs of parsing and serializing XML in order to scale their approach to wider networks.

Figure 2 below shows an example of SOAP messages – a request and a response. As can be seen, a SOAP message consists of a SOAP envelope, which in turns contains an optional SOAP header and a mandatory SOAP body. Here, a request to a stock quote service for Microsoft (stock symbol "MSFT") is made and the response of $58 as the price is given.

SOAP Request	SOAP Response
POST /StockQuote HTTP/1.1 Host: www.stockquoteserver.com Content-Type: text/xml; Content-Length: 323 SOAPAction: Some-Namespace-URI#GetLastTradePrice <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-ENV: encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"> <SOAP-ENV:Body> <m:GetLastTradePrice xmlns:m="Some-Namespace-URI"> <symbol>MSFT</symbol> </m:GetLastTradePrice> </SOAP-ENV:Body> </SOAP-ENV:Envelope>	HTTP/1.1 200 OK Content-Type: text/xml; charset="utf-8" Content-Length: nnnn <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-ENV: encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"> <SOAP-ENV:Body> <m:GetLastTradePriceResponse xmlns:m="Some-Namespace-URI"> <Price>58</Price> </m:GetLastTradePriceResponse> </SOAP-ENV:Body> </SOAP-ENV:Envelope>

Figure 2: SOAP envelopes representing a request and a response

UDDI

UDDI provides a mechanism for clients to dynamically find other Web services. Using a UDDI interface, businesses can dynamically connect to services provided by external business partners. A UDDI registry has two kinds of clients: businesses that want to publish a service (and its usage interfaces) and clients who want to obtain services of a certain kind and bind programmatically to them. Table 1 below is an overview of what UDDI provides [10]. UDDI is layered over SOAP and assumes that requests and responses are UDDI objects sent around as SOAP messages.

Information	**Operations**	**Detailed information** (supported by lower-level API)
White pages: Information such as the name, address, telephone number, and other contact information of a given business	**Publish**: How the provider of a Web service registers itself.	**Business information**: Contained in a *BusinessEntity* object, which in turn contains information about services, categories, contacts, URLs, and other things necessary to interact with a given business.
Yellow pages: Information that categorizes businesses. This is based on existing (non-electronic) standards	**Find**: How an application finds a particular Web service.	**Service information**: Describes a group of Web services. These are contained in a *BusinessService* object
Green pages: Technical information about the Web services provided by a given business.	**Bind**: How an application connects to, and interacts with, a Web service after it's been found	**Binding information**: The technical details necessary to invoke a Web service. This includes URLs, information about method names, argument types, and so on. The *BindingTemplate* object represents this data.

		Service Specification Detail: This is metadata about the various specifications implemented by a given Web service. These are called *tModels* in the UDDI specification

Table 1: Basic concepts behind UDDI

WSDL

WSDL (Web Service Description Language) provides away for service providers to describe the basic format of Web service requests over different protocols or encodings. WSDL is used to describe *what* a Web service can do, *where* it resides, and *how* to invoke it. In other words, a WSDL document describes a Web service's interface and provides users with a point of contact.

A complete WSDL service description describes two pieces of information – an application-level service description, or abstract interface, and the specific protocol-dependent details that users must follow to access the service at concrete service end points. This separation accounts for the fact that similar application–level service functionality is often deployed at different end points with slightly different access protocol details. This separation allows the reuse of abstract definitions of message interfaces.

Once the complete WSDL description is specified, developers can use WSDL as the input to a proxy generator that produces client code according to the service requirements. WSDL can also be used as an input to a dynamic invocation proxy, which can then generate the correct service requests at runtime. This will relieve the user and the developer of the need to remember or understand all the details of service.

3. Web Services: Practices, Issues, and Challenges

As promising as they are, Web Services is new and there are various open issues that surround this technology. Concerns such as security, privacy, and authentication capabilities are at the forefront of this. Many in the industry consider these open issues as the biggest stumbling blocks for Web Services.

To address the security issue, the W3C has issued XML-Signature Syntax and Process as a W3C Recommendation Design to provide a foundation for secure Web services. The proposed XML Signature standard, developed by a joint working group formed by W3C and the Internet Engineering Task Force (IETF), represents "a critical foundation on top of which we will be able to build more secure Web services," said Tim Berners-Lee, W3C director. XML Signature, which use the latest advances in applied mathematics cryptography to create and verify XML Signatures, offers "basic data integrity and authentication tools" that are designed to provide the level of security the W3C considers essential for Web services applications, said Berners-Lee.

From the vendor's viewpoint, IBM, Microsoft, and VeriSign have published WS-Security, a Web Services security specification. This represents an effort toward interoperable Web services that companies can deploy without the fear of exposing sensitive data or violating privacy practices. Rather than a complete security solution, WS-Security is a building block to be used in conjunction with other Web services and application specific protocols to accommodate a wide variety of security models and encryption technologies. This is an attempt to ensure that the proposal does not further concern the skeptics in the industry, who have already expressed concern that the current specification could lead to a move to set up a "toll way" for using Web services.

Another issue that needs to be resolved has to do with contracts and billing. The service providers will clearly want to earn revenue based upon service consumption. Thus, negotiated contract between the consumer and the provider must be established. This may involve a potentially large number of contracts and hence the management of the contracts and scalability will be a challenge. This problem is exacerbated by the need to maintain different pricing models based on different characteristics and usage. This is because there are no widely deployed standards for usage and billing today.

Another challenge is to more tightly integrate Web Services with the ideas and practices from distributed systems, the Web, database systems, and XML. This may prompt a new design, architecture, and/or engineering of middleware that support Web Services. This is the focus of the upcoming issue of IEEE Internet Computing (January/February 2003) entitled "Middleware for Web Services."

The above describes the current practices, issues, and challenges of Web Services. Based on the current state of technologies, the IDC predicts that Web Services will enter most organizations in three distinct phases. The "adoption table" is given in Table 2 below.

Timeframe	Activities
2002 (within the firewall)	Simplified app integration Increased developer productivity
2004 (contained external users)	Simplified business-partner connectivity Richer app functionality Subscription-based services
2006 to 2008 (fully dynamic search and use)	Casual / ad-hoc use of services New business models possible Commoditization of software Pervasive use in nontraditional devices

Table 2: Phases of Web Services penetration into organizations

4. Summary and Conclusions

This paper overviews the technologies behind Web Services and hints the opportunities of adopting the burgeoning technologies in IT organizations. The three major underlying technologies behind Web services, SOAP, UDDI, and WSDL, are detailed. It also describes the various issues and challenges that must be addressed by the stakeholders before Web Services can become ubiquitous. This is because building an ecosystem of fully integrated Web services (like the "fully dynamic search and use" phase given in Table 1 above) is much more complex than some have led others to believe. They are numerous characteristics to consider (some are described in Section 3 above) and these must be accounted for before Web Services can fully fulfill its promises.

While organizations contemplate when and how to leverage Web Services for their businesses, vendors and standard organizations are busy introducing new technologies to help the organizations take the dive. Researchers are also busy working on the next step for Web Services. For example, in the special issue of IEEE Intelligent Systems on "Intelligent Web Services" [11], articles that range from markup languages that enable intelligent service to operate over Web content to architectures that permit the location and composition of services to enact a variety of goal-directed behavior over the Web can be found.

With all of the above efforts, it is noteworthy to point out that while many believe it is very likely that the lower-level Web Services specifications such as SOAP, WSDL, and UDDI will be agreed upon, the more business-critical, higher-level specifications dealing with security, reliability, transaction control, and business process will be pulled in many different directions. Thus, the industry may be faced with the possibility that the open nature of Web Services will become corrupted with proprietary implementations, rendering interoperability a difficult goal to achieve.

If/when that happens, instead of asking "are you implementing Web Services," one will have to ask "what kind of Web Services are you implementing?" [12]. This will defeat all the inherent benefits and promises of Web Services. In anticipating this possibility, organizations such as the Web Services Interoperability (WS-I) organization are emerging to help pull some order to this chaos by specifying profiles of specification stacks that are applicable to different implementation scenarios.

References

1. Benfield , S., "Web Services: XML's Killer App," *Java Developers' Journal*, Vol. 6, No. 4., 2001.
2. Kiely, D., "WSDL for Defining Web Services," Cover Story, *XML Magazine*, Vol. 2, No. 4, August/September, 2001.
3. McCright, A., "Writing Your First Web Service: A Tutorial," *Web Services Journal*, June, 2001.
4. McDougall, P., "Decoding Web Services," *InformationWeek*, Oct 1, pp. 28-37
5. Dyck, T., "Web Services Wave" (the Cover Story: Web Services Wake-Up Call), *eWeek*, Vol. 18, No. 35, September, 2001.
6. Booch, G., "Web Services: The Economic Argument," *Software Development*, Vol. 9, No. 11, November, 2001.
7. Curbera, F., et al., "Unraveling the Web Services Web," *IEEE Internet Computing*, march/April, 2002.
8. *Web Services Journal*, Sys-Con Publications Inc., Montvale, New Jersey.

9. Gates, W., *Microsoft .NET Today*, an open letter to the Developers & IT Professionals, June 14, 2001.
10. Venu Vasudevan, "A Web Services Primer," http://www.xml.com/pub/a/2001/04/04/webservices/index.html, April 04, 2001.
11. Special Issue on "Intelligent Web Services," *IEEE Intelligent Systems*, January/February, 2002.
12. Bloomberg, J., Schmelzer, R., "The Pros and Cons of Web Services," *ZapThink Research Report*, May 2002.