Web Services—A Buzz Word with Potentials

János T Füstös

Abstract—*The simplest definition of a web service is an application that provides a web API. The web API exposes the functionality of the solution to other applications. The web API relies on other Internet-based technologies to manage communications. The resulting web services are pervasive, vendor-independent, language-neutral, and very low-cost. The main purpose of a web API is to enable application integration, and bring providers (developers) and requestors (users) together. More specifically, web services allow integration of heterogeneous applications. With solutions in place agents and other applications can connect to the service, and use the predefined processes. A centralized depository keeps track of the available functions and services, and with intelligent recognition and discovery semantics dynamic solutions can be created. Standardized description, self-describing structure, and discoverable providers make it easier to use existing solutions in new services and setups. In a long term it also offers the promise of the automated web.*

Introduction

Web services represent one of the latest and very exciting online technologies. They are based on a uniform and widely accessible interface, and use solutions that are implemented using traditional middleware platforms. (Vasudevan 2001) There are two different ways how they are generally introduced. One assumes that web services provide an overall solution to all existing problems by integrating different solutions, programming languages, applications, and software. This can be really confusing and disseminates false understanding about the true nature and the real opportunities what web services can represent. The other approach is highly technical, uses acronyms like .NET, J2EE, XML, WSDL, WSI, SOAP, ebXML, UDDI, JAX-RPC, JAXM, XAML, BEEP etc., and juggles with them as given and apparently well-known expressions without really explaining their meaning, content, and relationships. This paper takes a middle road, and provides an introduction of web services somewhere between these two approaches. It will use some technical terms with detailed explanations, refers to some of the most important aspects of web services focusing on general business applications, and drafts some new potential areas for service integration.

What is a Web Service?

There are several definitions for web services that emphasize different aspects, and approach the idea from different perspectives. The first introduction using a simple, general explanation can be "A web services is any service that is available over the Internet, uses a standardized XML messaging system, and is not tied to any operating system or programming language." (Cerami 2002) An IBM tutorial provides a more functional description: "Web services are a new breed of Web application. They are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. Web services perform functions that can be anything from simple requests to complicated business processes. A sample Web service might provide stock quotes or process credit card transactions. Once a Web service is deployed, other applications (and other Web services) can discover and invoke the deployed service." (IBM developerWorks 2003) For development purposes the most comprehensive definition comes from the World Wide Web Consortium: "A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-process-able format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards." (W3C Working Group 2004) Web services can also be considered as a new architectural paradigm for shared and reusable applications. They are solutions that apply industry-wide standards, interfaces, and protocols (Chatterjee & Webber 2004). Using existing solutions, servers and protocols web service developers can implement application integration projects, consolidate development efforts, reduce

redundant applications; in work-flow development web services can help to implement a general-purpose integration platform for application systems; and they can make it easier for partners to do business based on similar solutions.

Web Service System Architecture

Web services are built on the top of the HyperText Transfer Protocol. The eXtensible Markup Language provides the portability of solutions between different operating systems, programming languages, and standards. XML acts as a meta-language that allows description of complex structures and also assures interoperability between applications, between functional parts of services, and between servers and clients requesting services. Solutions that can be developed in any programming language handling any kind of data resources will use XML when communicating with other applications. These applications can provide functionality (e.g. common calculations), can request or deliver data (e.g. data stream, map, sound), or can represent building blocks of business processes, for example(payment operation).

Players and Their Roles

It seems that nearly every hardware and software vendor is touting a Web services strategy. The motivation behind Web services is integration: managers consistently list application integration as one of the top three technology issues facing businesses. The gained competitive advantages are two-folded: from a tactical perspective, application integration improves operational efficiency, resulting in reduced costs; from a strategic perspective, application integration enables better access to information resulting in better decisions. Unfortunately, it is very hard to integrate heterogeneous systems. The issue becomes much more challenging when a business tries to integrate its systems with those of its partners, suppliers, and customers. Web services using their generalized solutions and standardized protocols can contribute to these efforts. The integration and cooperation efforts can be approached in general terms when looking at the setup and construct of web services. There are three major roles in the web service architecture:

Service provider—Web service providers create, define and develop web services, and then publish them with access information in service repositories or make them available to web service requestors directly. Originally the services were:

Service requestor—Web service requestors perform a find operation to locate desired services made available by web service providers and then request those services from either web service registries or the publishers directly. Once the service has been located, web service requestors can then bind to those services to their applications and use their functions.

Service registry—Web service registries serve as centralized directories and repositories for web services defined and published by web service providers. The discovery framework offers APIs that allow the client to access services conforming to the various specifications. Service discovery is accessible to client applications, as well as to other parts of the architecture.

Service Layers

The web service protocol stack currently has four main layers. Like with other protocol stacks there is a close connection between the layers, and each of them has a different role. The stack is still evolving. Depending on specific solutions and requirements, other technologies might be added that are not required for all services but deliver additional solutions when needed (for example, authentication, certifications, or registration).

Transport—Web services are generally built on top of HTTP. Because web browsers can handle several additional protocols, data transport can be implemented by using FTP (file transfer), SMTP (mail), and also some newer solutions like BEEP (simultaneous and independent exchange of both textual and binary messages) or instant messaging.

Messaging—Extensible Markup Language (XML) is the lingua franca of web services. Using its own definitions (by implementing name spaces), and individualized data structures (which can be highly structured and complex) described in schemas, XML delivers a flexible way for data exchange that is independent from operating systems, programming languages, and data handling solutions. There are parsers, editors, and validators available on every platform for XML development. There are two different solutions how programs and processes can communicate with each other: RPC and SOAP.

Remote Procedure Call (RPC also called XML-RPC) is using the POST method of the HTTP protocol. Both requests and responses travel in XML format, and arrive in the body of the messages.

The Simple Object Access Protocol (SOAP) defines another method for programs running in one kind of operating system to communicate with programs in the same or another kind of operating systems by using HTTP and XML for information exchange. SOAP specifies how to encode an HTTP header and an XML file so that programs in one computer can call programs in another

computer, and pass information. It also specifies how the called programs can return responses.

Description—Web Service Description Language (WSDL) creates a technology that enables services to publish their interfaces. It specifies the syntax, vocabulary, and the controls for publishing web services. It defines the input and output operations, ports that are involved, includes information about data, data types, location of the service, and also how the information exchange can be bounded to a specific transfer protocol. WSDL is using XML to specify an abstract description of network service operations. Combined with concrete network protocol and message format they define an endpoint for data and service communication.

Discovery—Universal Description, Discovery, and Integration (UDDI) plays a key role in discovery services. It describes and provides a general solution for clients to dynamically find web services. Using a common interface, requestors can dynamically connect to services provided by external partners. A UDDI discovery registry can be thought of as a clearing house for business applications. The registry has two kinds of clients: businesses that want to publish and provide services, and clients who want to obtain services of a certain kind and invoke them as integrated parts of their solutions.

Data delivered by the discovery layer can be divided into three main categories:

- White pages: general information of a given business such as the name, address, telephone number, and other contact information.
- Yellow pages: general classification data that categorizes either the business, or the delivered service. This is based on existing (non-electronic) standards. Data may include industry, service description, and product codes.
- Green pages: technical information about the Web services provided by a given business, pointer to specification description, and information about how to invoke the service.

Solutions

There are several different implementations of web services and architecture. The existing solutions can be categorized as message-, policy-, resource-, or service oriented models depending on what is in the focus of the provided service architecture.

The *Message Oriented Model* looks at the ways how services are offered to agents, what are the structural parts of the messages, how are they delivered by the senders to the receivers. It also looks at delivery policies and covers reliability issues.

The *Policy Oriented Model* focuses on actions and states that are allowed in the setup for software agents, individuals working with the system, or for organizations that are part of the integrated solution.

The *Resource Oriented Model* looks at resources as main elements of the architecture. It focuses on attributes like ownership, location, policies that are related to the resources. In this context the role of a resource (whether it was provided or requested) plays only a secondary role.

In the *Service Oriented Model* the service and related actions are in the middle, and the other elements are built around them. The role of the players (providers or requestors) defines the connections and relationships to other services. This model will connect systems together at both the information and service levels (actually builds on top of the message oriented model), and includes solutions that provide quality of service and security. (Linthicum 2004).

Examples

Software vendors offer web service solutions and frameworks that allow developers and users to create applications.

- OASIS is a non-profit consortium that is coordinating the global development and adoption of e-business standards. Through the member organizations that are working together in technical committees developing recommendations and standard it is promoting open, scalable, and language neutral solutions.
- Oracle offers J2EE web service solutions which use JAX-RPC for remote connection. With additional Java-based elements (JCA Java connectors, Java messaging, standard beans) businesses can extend their ERP systems, and create connections between market-leader solutions like SAP, PeopleSoft, and Siebel.
- HP developed the first commercial implementation of a proposed Business Transaction Protocol. Their Web Services Transactioning system supports both B2B and B2C relationships.
- Sun Microsystems is one of the major developers of web service components and development tools. Their Java APIs, J2EE platform, server-side solutions, and enterprise java beans are highly flexible, scalable, and offer wide range of possibilities and options for development and integration.
- Microsoft supports web service development through their .NET environment. The available programming

tools support several APIs, and rely on different server technologies.

- IBM integrated several web service technologies and offers developing tools with their WebSphere application server. The available solutions support development of J2EE applications, publishing and deploying web services in both an extranet and intranet environment.
- ColdFusion MX from Macromedia can act as a registry, can consume, and also produce (offer) web services. The building blocks are using ColdFusion components.
- BEA Systems, IBM, Microsoft, SAP, and Sun are working together on a joint proposal to the World Wide Web Consortium that will standardize web services addressing. With that solution developers can simplify messaging services, and it will be easier for them to identify and exchange data and services between multiple end points.
- Esri is one of the major developers of map web service products. Their latest WMS solution integrates business data delivery and analytical capabilities with geographical services.
- The fact that Microsoft came out with their MapPoint web service solution indicates that web-based map services might play an important business role in the close future.
- Open GIS Consortium is very active in developing map service specifications for the web. Their Context Document Specification, Web Feature Services, and Web Coverage Services proposals are under review and discussion, and will possibly result in OpenGIS standards.
- XMethod (www.xmethods.com), UserLand (www.xml-rpc.com), OGC (www.opengeospatial.org), OASIS (www.oasis-open.org), and Microsoft (msdn.microsoft.com/webservices) have lists of publicly available web services and solutions.

Conclusion

Web services offer standardized solutions for integrated and distributed environments. They emphasize interoperability, platform independence, and support implementations in different operating systems and programming languages. These technologies are being standardized by W3C and OASIS. Another organization, WS-I, is defining additional guidelines for Web services interoperability. Companies like Sun, Hewlett-Packard, ebPML are working on additional functions to extend web services with security layers, authentication options, choreography and coordination languages, and business languages. According to new research from The Yankee Group some 70 percent of new IT spending is earmarked for integration technologies at the edge of the enterprise. "With the ongoing development of Web services code for workflow, messaging, e-commerce and security standards, enterprises are ready to create service-oriented architectures for software vendors to build next-generation applications …" (The Yankee Group 2004)

References

Cerami, Ethan. 2002. Web Services Essentials. Sebastopol, CA: O'Reilly Associates.

Chatterjee, Sandeep; Webber, James. 2004. Developing Enterprise Web Services: An Architect's Guide. Upper Saddle River, NJ: Prentice Hall PTR, HP Professional Books.

IBM developerWorks. 2003. Web services - The Web's next revolution. http://www6.software.ibm.com/developerworks/education/wsbasics/wsbasics-ltr.pdf.

Linthicum, Dave. 2004. Extending Your SOA for Intercompany Integration. Web Services Journal. 4(6): 12-15.

Manes, Ann Thomas. 2003. Web Services: A Manager's Guide. Upper Saddle River, NJ: Pearson Education.

Ray, Randy J.; Kulchenko, Pavel. 2002. Programming Web Services with Perl. Sebastopol, CA: O'Reilly Associates.

The Yankee Group. 2004. Web services technologies gain wider acceptance; majority of the mid market has plans to deploy in the next 12 months. 2004 U.S. Enterprise Web Services Survey. http://www.yankeegroup.com/public/products/research_note.jsp?ID=11945.

Vasudevan, Venu. 2001. A Web Services Primer. http://webservices.xml.com/lpt/a/ws/2001/04/04/webservices/index.html.

W3C Working Group. 2004. Web Services Architecture. Note 11 February 2004. http://www.w3.org/TR/2004/NOTE-ws-arch-20040211