View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by AIS Electronic Library (AISeL)

Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2003 Proceedings

Americas Conference on Information Systems (AMCIS)

December 2003

Web Services

Mark Gaynor
Boston University

George Wyner
Boston University

Bala Iyer
Boston University

Jim Freedman
Boston University

Follow this and additional works at: http://aisel.aisnet.org/amcis2003

Recommended Citation

Gaynor, Mark; Wyner, George; Iyer, Bala; and Freedman, Jim, "Web Services" (2003). *AMCIS 2003 Proceedings*. 421.
http://aisel.aisnet.org/amcis2003/421

This material is brought to you by the Americas Conference on Information Systems (AMCIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in AMCIS 2003 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.

WEB SERVICES

Mark Gaynor
School of Management
Boston University
mgaynor@bu.edu

George Wyner
School of Management
Boston University
gwyner@bu.edu

Bala Iyer
School of Management
Boston University
bala@acs.bu.edu

Jim Freedman
School of Management
Boston University
jfreedma@bu.edu

Abstract

Many believe Web Services are the best technology to integrate heterogeneous information systems, both internal and external to the organization. This tutorial provides an overview of web services technologies, including XML—the language that describes any type of data, SOAP—the electronic envelope to put the XML encoded data in, WSDL—the language to describe how to use a web service, and UDDI—the yellow pages of web-services.

Introduction

Web services enable flexibility in designing systems because they allow interchangeability of services by promoting a loosely coupled modular design. While not a new idea, Web Services is the first widely accepted set of standards defining a distributed computing environment. In this paper we set forth a brief overview of Web services by explaining why this new technology is of importance to practitioners and describing its key components and how they work. We discuss the strategic advantage of the Web services architecture—the flexibility to integrate heterogeneous information systems within and across organizational boundaries. Next, we examine the technology of Web services: we begin by discussing data representation using XML (extensible markup language) to encode any type of data flexibly. In addition to XML, other important protocols include SOAP (simple object access protocol), WSDL (web service description language) , and UDDI (universal description, discovery, and integration). SOAP is the messaging protocol used to encapsulate the XML encoded content for a Web service. SOAP defines the electronic envelope containing Web service data and is itself defined using XML. WSDL defines the method by which a Web service is to be invoked. UDDI provides a mechanism for registering a Web service in a "Yellow Pages" of Web services. These protocols, which are described in this paper, allow interoperability among disparate systems.

Web Services—The Big Picture for Business

The label "web services" refers to "loosely coupled, reusable software components that semantically encapsulate discrete functionality and are distributed and programmatically accessible over standard Internet protocols" (Sleeper 2001).

Several key elements of this definition warrant further discussion.

- Reusable Software Components: The concept of reusable software components is explained by the theory of modular design (Alexander 1964; Simon 1996).

Web Services applets semantically encapsulate discrete functionality, in the same way that objects encapsulate functionality in an object oriented system. This notion of encapsulation is an important element of thinking about modularity. For example, Parnas

(1972, p. 1056) discusses the advantages of designing a module "to reveal as little as possible about its inner workings." Baldwin and Clark (1997) refer to Parnas' concept of information hiding as hidden design parameters. This concept of focusing on the specific needs of a particular application without regard to other functions is a key element of achieving synergistic specificity (Schilling 2000). Encapsulating specific process knowledge within a discrete object is also a key element of sharing that capability within the framework of modular design.

- *Programmatic Accessibility.* Web Services are programmatically accessible. Unlike web sites and desktop applications, Web Services are not designed exclusively for direct human interaction, and do not necessarily have a user interface. Rather, web services operate at the code level; they are called by and exchange data with other software. An example of accessibility is the dynamic processing of an order, where the line items are priced through one called module, and the tax is assessed through another called module. The inner workings of the program are transparent to the system user.

- *Distribution.* Web Services are distributed over standard Internet protocols. They use the existing application protocols such as HTTP, ftp, or SMTP and conform to the standards of the Internet. This element is important for the successful adoption of web services. The success of the Internet is largely due to the simplicity and flexibility of the layered architecture of the technology that supports packaging and transporting data. Web Services are designed to use the packaging and transportation mechanism already adopted by firms worldwide.

Both a technical and a conceptual meaning is associated with Web Services. From a technical point of view, web services are a set of application-level standards built on top of TCP/IP – the standard transport within the Internet. These web services standards describe, precisely, how to discover and then invoke a web service.

Strategic Advantages

Web services promise competitive advantage to organizations in the form of superior integration, both internally among existing systems and externally with value chain business partners (Graham et al. 2002, pp. 10-13). Web services also enable information systems flexibility, which is important in a rapidly changing environment and critical to the success of dynamic business networks.

Technical Advantages

Principal advantage claimed by web services proponents are (1) access to data in legacy systems and (2) rapid development. Legacy applications capture the business rules and process logic of the organizations that they serve. The dynamic business environment drives organizations to integrate their information rapidly in unpredictable ways. Web Services provides a standard component architecture that creates a vehicle for disparate systems to share information seamlessly. Each of the legacy systems are "wrapped" by mapping the existing interface, which can range from a well organized API to a raw file transfer, to an interface consistent with the over-all Web Services Architecture.

Web Services Context: Definitions, Background and History of Distributed Computing

Web services can best be understood in terms of an ongoing effort to develop applications that can communicate with each other over a network. This capability is referred to as distributed computing. Distributed computing has a long history; web services are simply the latest incarnation.

Distributed computing is, at its heart, resource sharing across a network. In a distributed computing environment, the application must be broken up into components. This approach of breaking a system into components is a key element in the development of complex systems. A software component is a physical packaging of executable code (resource) with a well-defined and published interface. The key questions to ask are: How do you discover a resource? How do you know how to use a resource? What infrastructure do you need to make distribution work?

Web Services emerge as a standard for distributed computing. To see this, consider (in briefest form) the history of distributed computing, going back to the early days of the Unix operating system. Over the years a number of initiatives were undertaken to create a standard infrastructure for distributed computing. Components must have an infrastructure in place for the concept to work. Competing component models include Microsoft's DCOM (Distributed Component Object Model), The Object Management Group's CORBA (Common Object Request Broker Architecture), and Sun's EJB (Enterprise JavaBeans). The problem with these systems is not technical, but political: nobody won the standards battle.

Web Services Architecture

A key element of the promise of web services is that they are based on a set of widely accepted technical standards (Gaynor 2002). These standards together constitute an architecture for web services. Let's begin first by looking at the individual standards, which comprise this architecture, and then at how they fit together.

What Are the Main Standards for Web Services, and What Do They Do?

XML is a standard for describing data independent of platform and application. XML is the foundation on which the remaining standards are built as well as being the method by which components encode the data they exchange.

SOAP is a standard for packaging the information exchanged with web services. SOAP provides the envelope used to identify the sender and recipient and other header information required to get the information to its destination and have it understood properly.

WSDL provides a standard method for describing what a web service does and how to invoke it.

UDDI is the standard for registries, which describe large numbers of web services and thereby facilitate the search (possibly at run time) for appropriate services to meet application requirements.

In addition to these core standards, a number of additional standards[1] may play an important role in the emerging infrastructure for web services (Kleijnen and Raju 2003). For example, ebXML is a set of standards for supporting electronic commerce using XML-based messaging (**www.ebxml.org/geninfo.htm**).

Overview of How These Pieces Fit Together

To understand how these components fit together into a working system, consider the example of a simple web service to report the Manufacturer's Suggested Retail Price (MSRP) of an item. This pricing service would be listed, using UDDI, in a directory of Web Services along with directions for its use defined with WSDL. The client (e.g. an internet shopper) can look up the web service and call it to find the price of a given item. Both the request and response are sent in a SOAP message. All of this data are, of course encoded using XML, the *lingua franca* of web services.

XML—Extensible Markup Language

As it becomes the most popular way to encode data in a system-independent format, XML is becoming the *lingua franca* of the Internet. XML, because of its flexibility and extensibility, is able to encode just about any type of data imaginable. Many web service protocols such as SOAP and WSDL are defined using XML. Many vendors support XML, including most database vendors, and provide XML translation to and from their proprietary data format. It is this vendor acceptance that is propelling XML to its lofty role as the language of the Internet.

A language such as XML is needed to define data and its associated structure because human languages are not specific enough to define data unambiguously. On a hot day, if you tell someone you think his shirt is "cool," what do you mean? Do you like the shirt, or do you think it will be cool on such a hot day? Even in face-to-face communications, the English language is not always precise enough to avoid misunderstanding, which is why a precise data description language such as XML is required.

---

[1]See, for example, the standards being developed by the UN's Organization for the Advancement of Structured Information Standards (OASIS) (**oasisopen.org**).

Languages such as XML are called markup languages because information is enclosed between tags that define the context of this data. The idea of adding meta-information (i.e. data about the information) that explains, but is not part of the data, is not new. A complex specification called Standard Generalized Markup Language (SGML) existed in the mid 1980s. It provided the framework to define HTML in the early 1990s and, finally, XML by 1998.

This meta-information defines attributes about the data such as how to display it or what the data represents. Use of XML is split between two main applications: formatting data for consumption by people or formatting data for consumption by computers. Document-centric XML describes how data is structured for display. It includes tags such as <b> …</b> to bold a piece of text as it is displayed to a monitor or printed. The other main use of XML is data-centric XML, which describes structured information from databases, data structures used in computer programs, and other sources of complex data. Most data-centric XML documents are designed to be parsed by a computer. The data-centric use of XML, which places data in the context of what it means, is the base technology behind web services.

In XML (and HTML), data is enclosed between tags that define the context of the data. Because tags can be nested, hierarchical complex structured data representation is promoted. What follows is a simple example of a data-centric XML document describing information about one of the author's books.

```
<book>
   <title>Network Service Investment Guide</title>
   <author>Mark Gaynor</author>
      <email>mgaynor@bu.edu</email>
   <publisher>Wiley</publisher>
   <ISBN>0471-21475-2</ISBN>
</book>
```

This example is one representation of this information. Note the hierarchical nature of this data: each book has at least one author, and associated with each author is contact information. One nice (but inefficient) attribute of XML is the readability of its ASCII encoding. This simple example illustrates how information about this book might be encoded with XML.

A program must know two things about the data it is processing: (1) how this data is represented and (2) what this data is. In the preceding XML example, the author is Mark Gaynor, but how is this information represented? In this case, it is not hard to deduce that this data is represented as a string. However, a precise way is needed to describe this specification to prevent misunderstanding about how data is typed. XML allows one XML document to be linked to a special document, called a schema, which defines precisely how the data in the associated data document is typed. To see how this schema works, consider the example of a web service checking inventory for some specific quantity of a particular item. The data this web service needs is the code for the item and how many items to check for. One choice is to make the item code a string and the quantity an integer (assuming you can check only for whole numbers of items). The code that follows is a simplified XML representation of this information for a web service called check_item:

```
<xsd:schema xmlns="check_item"
         xmlns:xsd"http://www.w3.org/2001/XMLSchema"
         targetNamespace="check_item"

      <xsd:simpleType>
      <xsd:attribute name="item_code" use="required">
         <xsd:simpleType>
               <xsd:restriction base="xsd:string">
            <xsd:pattern value="[A-Z]-\d{5}"
         </xsd:restriction>
      </simpleType>
   <xsd:attribute name="item_num" use="required">
      <xsd:simpleType>
         <xsd:restriction base="xsd:integer">
            <xsd:minExclusive value="0"/>
            <xsd:maxExdlusive value="1000"/>
         </xsd:restriction>
      </simpleType>
```

This first element is a string representing the unique code of the item to check for inventory. This string consists of a single alpha character (between A and Z), followed by a hyphen, and then five numbers. For example, B-49899 is a valid item code, but B9-999 is not. The second parameter is the quantity needed. Its range is greater than zero and less than 1000. This example demonstrates how XML is used to describe data precisely. In this case, the data is represented without ambiguity; as long as you understand the rules of XML encoding, you can't misinterpret the data.

Here is an example of how part of the XML request will look to the server when requesting inventory for 500 items of product D-23456:

```
<item_code="D-23456">
<Item_num="500">
```

This piece of code contains data that fits into the definition provided by the metadata in the XML schema associated with this instance of its use. While the schema seems hard to read, its highly structured nature is easy for a computer to parse.

This very basic introduction to XML is sufficient to understand how web services are put together and why they have such potential to change business processes fundamentally (XML puts information in the hands of those who need it). Don't be fooled into thinking XML is simple—it is not. Its flexibility to represent complex structured data creates complexity. Fortunately, this complexity is hidden from the users of web services and is not necessary to understand web services at a business level.

SOAP—Simple Object Access Protocol

SOAP is the protocol used to exchange XML documents over the Internet. It provides the definition of XML-based encoded data used for exchanging structured and typed information between peers in a decentralized, distributed environment (SOAP 2002). Using XML, SOAP defines the messages passed back and forth between the web service client and the server. SOAP is flexible in how it defines the container for the XML data. This container is the SOAP envelope, which contains headers and a body. This SOAP message is not tied to any particular transport protocol and can be transported with HTTP, SMTP, or even FTP. SOAP is platform independent because any two operating systems with a SOAP stack can communicate via SOAP messages, allowing applications running on heterogeneous systems to speak the same language. SOAP is part of a distributed computing environment that is mostly invisible to users. SOAP is the first such distributed messaging protocol accepted by most major vendors including Microsoft, IBM, and Sun.

SOAP provides support for many aspects of distributed computing. Some functions it performs are:

- Defines the unit of communication—the SOAP envelope that contains all message headers and body
- Allows handling of errors by the SOAP fault mechanism
- Provides a scheme allowing extensibility, which implies that SOAP's evolution is not constrained
- Allows several paradigms of communications: the RPC, direct document-centric, and indirect document-centric approaches to pass SOAP messages between users
- Determines the protocol used to transport SOAP messages (HTTP, SMTP, or FTP)

WSDL—Web Service Description Language

We know how web services communicate with XML-encoded data within SOAP envelopes, but how do these clients know how to use these web services? That is the job of WSDL. WSDL is just another Interface Definition Language (IDL), with one big difference from the IDLs of CORBA, DCOM, and others—everybody agrees that WSDL is the standard to describe web services. As expected, WSDL is defined with XML. WSDL describes everything needed to use the web service, as well as what to expect back from the service. This information includes what protocol is used to transport the SOAP message (HTTP, SMTP, or FTP), how the data passed between client and server is structured, what it means, and the URI (Uniform Resource Identifier) by which the service is accessed. Web services can exist without a WSDL description, but without this description, clients can not figure out, in an unambiguous manner, how to access the desired web service.

Describing the interface to a particular web service is important to its successful adoption. The details of how this description is specified are not important, but understanding the type of information required to use the web service successfully is. Following is a list of some important information that needs to be documented:

- For each message between the client and the server, the structure of the data must be described. This function associates a data type with each item of data passed in a message between client and server. The types element is used for this function.
- Each message passed between the client and the server must be specified. This function names each message that has a types definition. The message element is used for this function.
- For each web service, the interaction between the client and the server needs defining. Does the service expect a request-response, or is the interaction different? The portType element explains details of this interaction between client and server.
- The client must know what protocol to use to connect to the server. This protocol might be HTTP, SMTP, or FTP. The binding element performs this function.
- The client must know the location of the service so that it can be called. This location is specified with the service element

WSDL is an important part of web services because it allows potential users of a service to evaluate the service and learn how to use it. Without a WSDL description of a web service, the user must depend on verbose descriptions of the web service, which make precise understanding difficult. Without this detailed and precise definition of web services, it is difficult for programmers to invoke them correctly because misinterpretation is so easy with informal descriptions. In essence, WSDL transforms a web service into a commodity that any client can easily use.

UDDI—Universal Description, Discovery, and Integration

We know how to define what a web service does, but how do users discover what web services are available to them? That is the job of UDDI—it allows a web service to be registered so that users can find the WSDL of the web service. UDDI allows the provider of a web service to advertise it by providing a central directory service for publishing technical information about web services. This central database of web services is called a registry. UDDI is implemented as a web service that allows manipulation of this registry containing definitions of web services. Users find great value when they have many choices for a web service—it is the value of the best of many (Gaynor 2003). Service providers find value in UDDI because it allows them to become one of the choices users have. When market uncertainty is high, this association with UDDI is potentially worth a lot of money to the service provider with the best service. UDDI lowers the entry barriers for new providers wishing to enter the market because its service enables users to know what their choices are. Similar to other web service protocols, the agreement between all the vendors is its most valuable attribute. The idea behind UDDI is to provide a public place for web services to be cataloged.

Putting the Web Service Puzzle Together

In summary, we presented the framework defining what a web service is and how to use one. We started at the bottom by defining XML, which is the language used to describe technical aspects of web services in a precise way. Then we discussed how this XML information is packaged into a SOAP message that is independent of any vendor's proprietary architecture. WSDL is discussed as the standard language that describes how to use a particular web service. Finally, UDDI defines the infrastructure to build registries of technical information about web services. Together these standards allow a commoditization of web services. These main puzzle pieces define how web services work, how they provide a universal standard for sharing information from heterogeneous systems, and ultimately why they will provide such value to the networked economy.[2]

References

Alexander, C. *Notes on the Synthesis of Form*, Cambridge, MA: Harvard University Press, 1964.

Baldwin, C. Y., and Clark, K. B. "Managing in an Age of Modularity," *Harvard Business Review* (68), 1997, pp 73-109.

Gaynor, M. *Network Services Investment Guide: Maximizing ROI in Uncertain Times*, New York: Wiley, 2003.

Graham, S., et al. *Building Web Services with Java: Making Sense of XML, SOAP, WSDL and UDDI*, Indianapolis, IN: Sams, 2002, p. 581.

[2]Significant portions of this tutorial draw on a previous paper by the authors (Iyer, *et al.* 2003) and a book by the first author (Gaynor 2003).

Iyer, B., et al. "Enabling Dynamic Business Networks using Web Services," *Communications of the Association for Information Systems* (11: 30) 2003.

Parnas, D. L. "On the Criteria To Be Used in Decomposing Systems Into Modules," *Communications of the ACM* (15:12) December 1972, pp. 1053-1058.

Schilling, M. A. "Toward a General Modular Systems Theory and Its Application to Inter-Firm Product Modularity," *Academy of Management Review* (25), 2000, pp. 312-334.

Simon, H.A. *The Sciences of the Artificial*, Cambridge, MA: The MIT Press, 1996, p. 231.

Sleeper, B. "Defining Web Services," San Francisco: The Stencil Group, 2001.

SOAP. **http://www.w3.org/TR/2002/WD-soap12-part0-20020626/**, 2002 (accessed July 23, 2002).