Context and Concept of Web Services

Vijay Kasi
Georgia State University, USA

Brett Young
Georgia State University, USA

INTRODUCTION

Web Services Defined

The term Web services has as many definitions as there are people who have worked on it. The different definitions, in general, stress various aspects of Web services. The diverse nature of these definitions confirms the diverse interpretations of Web services ("Evolution of Integration Functionality," 2001; Freger, 2001; Infravio, 2002; Ogbuji, 2002; Stal, 2002; Wilkes, 2002). The big computer giants such as Microsoft and IBM promote Web services, and the definitions offered by them are as follows.

- **IBM:** "A Web Service is a collection of functions that are packaged as a single entity and published to the network for use by other programs...[They are] self-describing, self-contained, modular applications..." (Glass, 2000).
- **Microsoft:** "Web Services are a very general model for building applications and can be implemented for any operating system that supports communication over the Internet and represent black-box functionality that can be reused without worrying about how the service is implemented...[They use] building blocks for constructing distributed Web applications..." (Kirtland, 2001).
- **World Wide Web Consortium (W3C):** "A software system identified by a URI [uniform resource indicator], whose public interfaces and bindings are defined and described using XML [extensible markup language]. Its definition can be discovered by other software systems. These systems may then interact in a manner prescribed by its definition, using XML based messages conveyed by internet protocols" (W3C, 2002).

This article attempts to clarify these generic definitions into language that is tangible and meaningful to the reader. To do so, background is given on the systems, applications, and architecture that led to the need and development of Web services.

BACKGROUND

The advancement of technology in computer systems and business applications in the current era are unprecedented. The rapid pace of application development has led to the development of disparate applications and services. In an era of mergers, acquisitions, and virtual integration, there is an increasing need to link disparate applications and services. The information needs to be shared not only within a business, but between businesses. We are moving from an era of working within applications intra-enterprise to working between applications interenterprise. Employees, customers, and partners require easy access to information and services whenever they need it.

Considering the risks involved, technology integration is one of the principal challenges in enterprises (Raczkowski, 2002). Web services promise a new level of interoperability between applications and enable the integration of enterprise applications. Web services are expected to get a major share of business-integration expenditures and are expected to grow in the next 5 years (Raczkowski). Web services, being a subset of the application-integration market, will experience high growth and, according to *IDC* ("Cautious Web Services Software Adoption," 2004), spending on Web-services projects will reach $11 billion by 2008.

MAIN THRUST OF THE ARTICLE

Web services, while still in their infancy, show potential. However, they are not a panacea to all the business problems today, and some issues remain needing resolution. In this article, the historical context of Web services positions them with previous technologies. Web services are not the first approach to streamlining enterprise application integration. There is a history of numerous approaches to integrating enterprise applications. After positioning Web services in their proper context, the article discusses the concepts that underlie Web services and how they are applicable. The architecture and protocols used in Web services are discussed next. Finally, future Web-services issues are identified.

Object-Oriented Paradigm

The procedural programming paradigm emerged in the 1970s because of intricacies in understanding assembly language programs. Though procedural programming proved useful, it fell short of providing the necessary tools needed to model real-world entities; thus, the software-development process became enormously cumbersome, which led to the development of object-oriented programming. Programmers could now reuse the software units by implementing a packaging scheme. This scheme focused on application areas, facilitated the reuse of software components, and led to the development of distributed systems and distributed object models.

Distributed Computing Systems

The technological advances in the computer engineering discipline resulted in the emergence of powerful computers and computer networks. Computing was distributed over the networks instead of being performed on a single, centralized computer. This led to the development of the two-tier architecture, more commonly known as the client-server architecture. In the two-tier architecture, the operations were decomposed into two parts, with one part, the client, initiating a distributed activity, and the other part, the server, carrying out the activity.

The centralization of activities presented some problems in terms of scalability and flexibility. Thus, the three-tier architecture was proposed by adding an additional tier to separate the application tier into the presentation and data tiers (Gunzer, 2002). The three-tier architecture imparted unprecedented flexibility and provided a feasible alternative to deal with the issue of scalability. This formed the basis for distributed application development and distributed computing. Some of the popular middleware technologies are discussed below.

Common Object Request Broker Architecture (CORBA)

CORBA is an open-standards-based solution to distributed computing. The Object Management Group, an industry consortium, developed the specifications for CORBA. The primary advantage of CORBA is that clients and servers can be written in any programming language. That is possible because the objects are defined with a high level of abstraction provided by the interface definition language (IDL). After defining an IDL file, a compiler takes care of the mapping of the IDL file to a specific programming language. Though advantageous, compatible object request brokers are needed on both of the connections to make the client and the server objects communicate (Gunzer, 2002; Ogbuji, 2002).

Distributed Component Object Model (DCOM)

Microsoft's DCOM has a layer that sits on the top of a remote procedure-calling (RPC) mechanism and allows calls to remote objects that interact with the COM run-time services. The DCOM server publishes its methods to the clients by supporting multiple interfaces. These are written in the interface definition language, which is similar to C++ (Gunzer, 2002; Ogbuji, 2002).

Remote Method Invocation (RMI)

RMI enables one to create Java-to-Java applications, in which the methods of remote Java objects can be invoked from other Java virtual machines. A Java program can make a call on the remote object once it obtains a reference to the remote object. Programming with RMI is straightforward once the programmer has gained some experience with the Java and distributed applications. On the other hand, RMI may only be used when Java exists on both sides of the connection (Gunzer, 2002; Infravio, 2002; Ogbuji, 2002).

World Wide Web

The advancements in the distributed object models proceeded in parallel with the development and establishment of the World Wide Web architecture. This gave rise to some standard protocols that were widely accepted. The transmission control protocol/ Internet protocol (TCP/IP) formed the common base protocol for connectivity. This was followed by the advent of various protocols like the hypertext transfer protocol (HTTP), file transfer protocol (FTP), and Gopher for specific needs.

HTTP became the common communication protocol. The emergence of the World Wide Web brought about the acceptance of a standard Web architecture. Thus, a platform was set in terms of standards, and the advancements in distributed object models saw the surfacing of Web services.

In terms of presentation, the hypertext markup language (HTML) established itself as the standard. The capabilities of HTML were very limited and did not provide any structure or programming ability. To counter these deficiencies, XML was created. XML provided the ability to program over the Web as an addition to presentation.