SOFTWARE DEVELOPMENT AND INTEGRATION USING WEB SERVICES

Chakib Chraibi, Barry University, cchraibi@mail.barry.edu

ABSTRACT

The processing, exchange, and integration of the enormous amount of information on the World Wide Web, on one hand, and the need to rapidly develop dynamic and agile enterprise applications, on the other hand, have led to the development of powerful Web applications, referred to as Web services. Web services are at the foundation of a swift change of paradigm in software development and deployment. Web services are XML-based components that are language, platform and location independent. They are self-contained, self-describing, modular applications that can be published, located and invoked across the Web. They are becoming the major components for the creation and integration of large-scale distributed systems. An overview of web services and their effect on distributed computing is provided. Associated enabling technologies and frameworks are discussed and analyzed.

Keywords: Web applications and services, enterprise integration, XML, J2EE, .NET.

INTRODUCTION

The remarkable growth of the Internet has given way to an ever-expanding network infrastructure in terms of connection and bandwidth. This has motivated the creation of an enormous amount of information on the World Wide Web that needs to be processed, exchanged, and integrated. On the other hand, in the Internet age, there is a need for highly responsive, constantly improving software applications with shorter application life cycles. This has led to the development of powerful Web applications and services that are at the foundation of a change of paradigm in software development and deployment.

Web services are XML-based components that can be called from any other application, regardless of their language, platform, and location. They are self-contained, self-describing, modular applications that can be published, located and invoked across the Web. In other words, web services are basically server functions that have published the interface mechanisms needed to access their capabilities over the Internet. They communicate using platform-independent XML over standard Internet protocols, such as HTTP and SMTP. They are becoming the major components for the creation and integration of large-scale distributed systems.

Web services accept requests and provide responses across the Internet, using standardized XML-based messaging system and protocols. Since they are not tied to any system platform or programming language, they are transforming the Web into a repository of interoperable components that can be used to create or execute simple or large-scale applications over the Internet, moving towards the goal of seamless integration and rapid application development.

According to recent studies, an overwhelming number of IT organizations (up to 91%) have a backlog of application development projects. More than half of these backlogs are significant. The key is to complete enterprise applications more swiftly and reliably. Furthermore, there is

an urgent need to create an integrated information technology infrastructure that would meet customer and business expectations for real-time information and business-to-business collaboration. In this paper, we provide an overview of web services and their effect on distributed computing and software development. In particular, we discuss the enabling technologies and the fundamental frameworks used for their development and deployment in order to provide rapid application development and application integration and interoperability within the enterprise and across enterprises.

To remain competitive, enterprises must develop integrated information technology systems that allow instant and easy interoperability among the different units within the organization, as well as outside the organization with business partners, suppliers and customers. Web services provide a loosely-coupled and platform-independent infrastructure for connecting people, processes, and applications behind or beyond the firewall. This new model of integration is more flexible, responsive, and based on open XML-based standards that are widely supported and deployed. Figure 1 shows an example of integration using Web services.

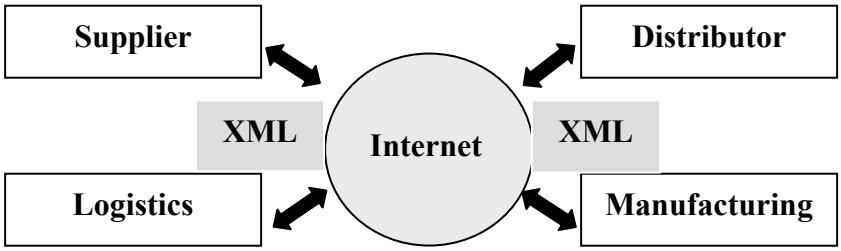

Figure 1. Enterprise Integration with Web Services

According to a recent report from infoline.com, 88% of CTOs are looking into web services to increase the bottom-line effectiveness of their company. The rental car company Dollar has claimed that its web services-based integrated service with Southwest Airlines has reduced by 4 US dollars each single transaction and generated an incremental 10 million dollars in revenues. It seems then crucial to look into Web services and how they provide enterprise integration through a service-oriented architecture. In this paper, we depict the fundamental technologies and methods behind web services.

WEB SERVICES ENABLING STANDARDS

The XML-based standards make web services interoperable across heterogeneous platform and languages and accessible through protocols such as HTTP. Web services are described using WSDL (Web Services Description Language) and registered in an XML-based registry such as UDDI (Universal Description, Discovery, and Integration). These services may be accessed using SOAP (Simple Object Access Protocol) messages.

The life-cycle of a Web service is depicted in Figure 2.

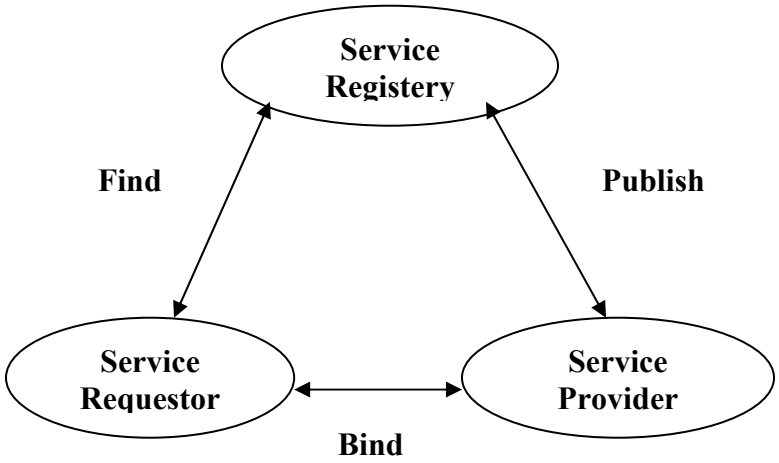

Figure 2. The life-cycle of a Web Service

The service provider develops a Web component with a useful functionality, as well as a SOAP service wrapper. Then, a WSDL file is created to define web services interfaces, data and message types, and protocol mappings. Then, the service should be deployed and published in a UDDI registry. The requestor to find services uses UDDI. A SOAP client application should be developed to invoke the Web service.

UDDI provides a registry of web services. It is used for describing, discovering, and integrating web services. The UDDI registry includes WSDL documents that provide information about web services, including the names, categories, and access points. UDDI utilizes SOAP messaging for publishing, editing, browsing, and searching for information in a registry.

WSDL allows the specification of a SOAP Web service as an XML file. WSDL provides the template for describing and publishing web services. It also allows clients how to interact with a Web service and invoke its public interface. WSDL standardized remote invocations by describing the function's structure, the input/output parameters, and the transport protocol binding, using an XML grammar.

SOAP is an XML-based, platform independent protocol for information exchange between applications. It was originally designed to transport remote procedure calls via HTTP. But now, it provides a simple and standard packaging structure for transporting XML documents over different standard Internet technologies, including SMTP, HTTP, and FTP. A SOAP message is an XML document that contains a SOAP envelope, which includes information such as the method name and the method parameters, as well as some encoding and binding mechanisms. Table 1 provides an example of a SOAP request and a SOAP response from a fictional weather service [1].

<?xml version='1.0'?> <SOAP-ENV:Envelope xmlns:SOAP-ENV ="http://www.w3.org/2001/09/soap-envelope/" xmlns:xsi =http://www.w3.org/2001/XMLSchema-instance xmlns:xsd ="http://www.w3.org/2001/XMLSchema"> <SOAP-ENV:Body> <ns1:getWeather xmlns:ns1="urn:examples:weatherservice" SOAP-ENV:encodingStyle= "http://www.w3.org/2001/09/soap-encoding/"> <zipcode xsi:type="xsd:string">33027 </zipcode> </ns1:getWeather> </SOAP-ENV:Body> </SOAP-ENV:Envelope>	<?xml version='1.0'?> <SOAP-ENV:Envelope xmlns:SOAP-ENV ="http://www.w3.org/2001/09/soap-envelope/" xmlns:xsi =http://www.w3.org/2001/XMLSchema-instance xmlns:xsd ="http://www.w3.org/2001/XMLSchema"> <SOAP-ENV:Body> <ns1:getWeatherResponse xmlns:ns1="urn:examples:weatherservice" SOAP-ENV:encodingStyle= "http://www.w3.org/2001/09/soap-encoding/"> <return xsi:type="xsd:integer">90 </return> </ns1:getWeatherResponse> </SOAP-ENV:Body> </SOAP-ENV:Envelope>

Table 1. Example of a SOAP Request/Response

Figure 3 shows an example of invocation of the temperature Web service through a generic soap client and the response in HTML.

Figure 3. Example of an interactive use of a Web Service

WEB SERVICES PLATFORMS

A web services platform is an environment used to host one or many web services, which typically includes UDDI business registries, SOAP servers, and additional security and transaction capabilities. The main Web service frameworks include Sun Open Net Environment, IBM Web Sphere, BEA's Web Logic, Oracle 9i Application Server, which are all based on the J2EE specification, and Microsoft's .NET, which uses the .NET platform.

The J2EE (Java2 Platform, Enterprise Edition) consists of a set of specifications and a platform for building enterprise-level applications. This technology provides a component-based approach to the design, development, assembly, and deployment of enterprise applications. J2EE components are written in Java, compiled into Java Byte Code, assembled into a J2EE application, and deployed on the J2EE server after verifying they are in compliance with the J2EE specification. The J2EE platform uses a multi-tiered distributed architecture and XML-based data interchange. Application clients directly access enterprise beans running in the business tier. The middle tier infrastructure is Enterprise Java Beans (EJB). Enterprise beans are larger, coarse-grained applications that are ready to be deployed. The database tier handles enterprise information system software, and includes enterprise infrastructure systems such as JDBC compatible databases. Figure 4 gives an overview of the general J2EE architecture.

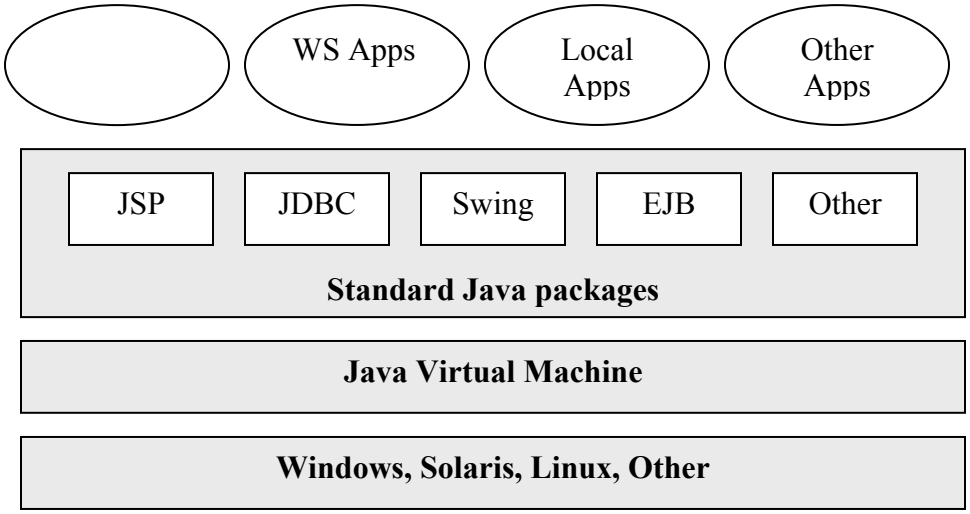

Figure 4. The J2EE Architecture

As the J2EE architecture, the .NET platform architecture is a collection of tools and technologies that help build and deploy robust enterprise applications. However, .NET allows several languages to be used, including Visual Basic.Net and C#. At the base of the .NET architecture is the Common Language Runtime (CLR). The CLR is the engine that manages the execution of the code. The .NET Enterprise Servers, such as Microsoft's SQL server, are add-on server products designed to provide specialized services. The .NET platform enables software integration through the use of XML web services. The .NET vision is that in the future all devices will be connected to the Internet and use standardized languages such as XML and SOAP to communicate.

Figure 5 shows a global view of the .NET architecture.

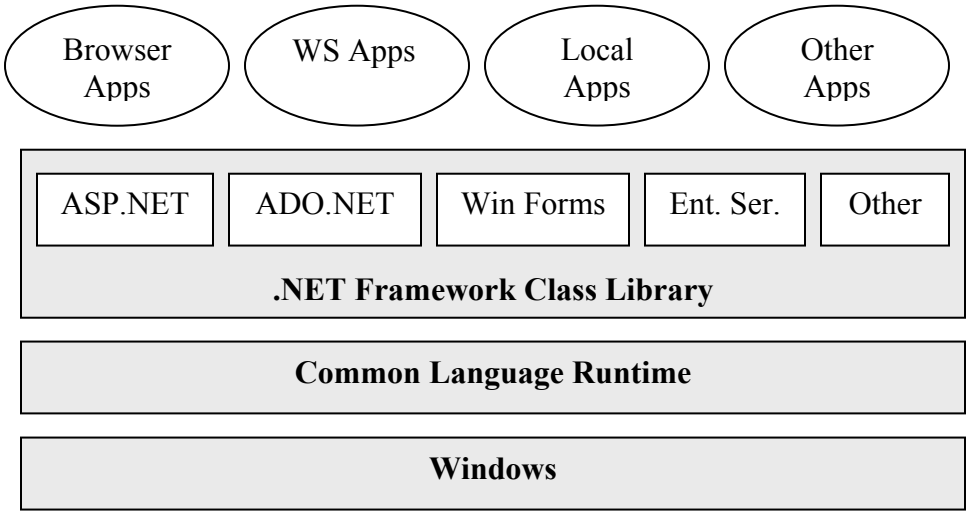

Figure 5. The .NET Architecture

Although there is a tremendous overlap between the J2EE and the .Net platform technologies, there are certainly significant differences. The most obvious one is that J2EE is a single language multi-platform system, while .Net is a multi-language single platform system [3, 4, 5, 6].

WEB SERVICES SECURITY

Security is a major issue when deploying a Web service. First, the messages exchanged are in XML, which is in a human-readable text form. Second, these messages usually travel through HTTP, which is an established and known protocol. Third, Web services aim to integrate with trading partners beyond the firewall. Finally, Web services business transactions frequently involve a multi-hop topology with intermediate actors, crossing multiple trust domains and encryption keys.

Fundamental security requirements include authentication, authorization, integrity, confidentiality, and non-repudiation. Although, these different security mechanisms have other underlying business reasons, their main purpose is to ensure the trust in carrying out business transactions on computer and network systems. This attribute is essential for Web services whose main purpose is to provide access to business functions over the Web through open and well-known standards.

Traditional security technologies used on the Internet include firewalls and HTTPS (HTTP over TLS/SSL). Web services requests are designed to tunnel through firewalls. However, firewalls are designed to restrict access to hosts and ports, but they do not leverage different levels of access control from the same source. HTTPS is transport-dependent and point-to-point. Although it provides authentication, encryption and data integrity, it has no mechanisms for securing transactions beyond the HTTP server.

Thus, current security mechanisms are insufficient by themselves to address the security needs of Web services. They have to be incorporated and enhanced in a comprehensive security strategy that ensures end-to-end security, strong authentication and authorization, flexible access control, sophisticated security policy, auditing, data integrity and confidentiality, and last but not least, interoperability based on various security mechanisms.

To address this challenge, a unifying approach drawn from different security technologies is required. Several models and technologies are under study. They are based on existing security technologies, such as SSL/TLS, Kerberos, PKI, and XML-related security mechanisms, such as XML encryption, XML signature, XML canonicalization and SAML (Security Assertions Markup Language). The aim is an end-to-end security framework for exchanging security information between business partners over the Internet.

CONCLUSION

Web services are software components providing services across the Internet via an XML-based messaging system. The purpose is to provide seamless interoperability, communication, and cooperation across heterogeneous systems, platforms, and programming languages. This would allow the Internet to be used as a backbone for a repository of Web components that can be directly invoked or used to build large-scale applications within and between organizations. This should facilitate application integration in an enterprise or across enterprises, e-business and B2B applications. Web services would also permit rapid application development so much needed in this Internet age, which requires short application cycles and distributed, extremely agile, and dynamic systems.

However, issues of interoperability and scalability are not settled yet. Some XML-based standards used in the web services framework have not been officially endorsed by the W3C. Furthermore, not all specifications are backward compatible. On the other hand, web services create a new level of openness that raises the security vulnerabilities that need to be addressed. This should require a combination of security mechanisms to enforce the appropriate level of security needed.

REFERENCES

1. Cerami, E. (2002). Web Services Essentials. Sebastopol, California: O'REILLY.
2. Chappel, D. and Jewell T. (2002). Java Web Services. Sebastopol, California: O'REILLY
3. Curbera, F. et al. (2002). Unraveling the Web Services Web, IEEE Internet Computing, vol. 6, no. 2, March/April 2002, pp. 86-93.
4. Evjen, B. (2002). XML Web Services for ASP.NET. New York, New York: Wiley.
5. Naggapan, R. et al. (2002). Developing Java Web Services. New York, New York: Wiley.
6. Nghiem, A. (2003). IT Web Services. Upper Saddle River, New Jersey: Prentice-Hall.