A Proposal for A Comparison of Web Service Development Methodologies

[redacted]

Abstract

Web services are considered the future of the Internet. Since Web services are a relatively new technology there are evolving standards and methodologies to developing them. I propose to research and compare the existing Web service development methodologies.

1 Introduction

The definition of a Web service is a software system that can be connected to over the internet which is defined and described using eXtensible Markup Language (XML). Web services, in general, have become a powerful way for information to be distributed and accessed from multiple platforms anywhere on the Internet quickly. Since these services are built on a single Web server instead of personal computers, they are readily available practically from any device that has an Internet connection. This device could be a cell phone, personal digital assistant or a personal computer. The greatest advantage to

Web services is their ability to interact with each other through the Internet. This communication ability is made possible through a series of protocols all based on XML.

Consider an example of a Web service for a company with multiple departments which need to communicate. Each of these departments has their own software systems that run independently. Implementing a Web service would allow all of these departments to communicate freely. This communication is accomplished using XML and The Simple Object Access Protocol (SOAP).

SOAP is a protocol designed for use exchanging information in a decentralized and distributed environment [1]. The SOAP protocol is based on XML and can be broken down into three parts: a set of encoding rules, a to represent remote procedure calls and an envelope that contains what is contained in the message and how to process it. Simply stated, the SOAP protocol can be considered a grammar for XML.

Now that the company's multiple software systems are able to communicate freely with each other consider that the company wants its Web service to be able to communicate with another companies service. The Web Service Description Language (WSDL) enables this communication. WSDL is also based upon XML and describes how to access a Web service and what services it provides. With SOAP and WSDL multiple company's Web services can freely communicate and transfer data.

Finally that these businesses' Web services are connected, how do these businesses find the services they are looking for? This is done through the Universal Description Discovery and Integration (UDDL) directory. It simply serves as a directory for businesses and organizations to list what services they provide.

This example uses the traditional style of developing a web service. The most commonly used protocols are SOAP, WSDL and UDDI. There are also other communication protocols developing. For example, the Web Service Transaction Language (WSTL) is built on top of WSDL [12]. WSDL provides a remote service the information needed to interact and WSTL allows the remote service to understand the transaction support of the local Web service. Another communication protocol called the Web Service Offerings Language (WSOL) contains formal descriptions of various constraints and classes of service for Web services. [16]. Yet another developing standard is the XML Service Request Language (XSRL). XSRL is based off of XML and AI planning techniques. XSRL has the ability to express requests and constraints over as well as generating a schedule for interacting with the UDDI resident services [11].

As a result there are a number of emerging standards, protocols and methodologies to consider when developing a Web service. I propose to research and compare these existing Web service development methodologies.

2 Prior Work

Coupling and cohesion are two well-known software design principles which are essential when designing a Web service. Coupling is known as the inter-dependence between two processes. The goal is to reduce coupling as much as possible. This would thus promote independence of processes. This in turn reduces problems of duplication and redundancy.The second software design principle is cohesion. Cohesion is simply the degree of strength of functional relatedness of operations within a service [10].

Figure 1 shows the architecture of a typical Web service. There are three sections in the stack: the wire, description, and discovery agency. At the bottom is the wire section. The transport layer handles the network connectivity over the TCP-IP protocol. The packaging layer handles how the data is encoded before being transported. The extensions layer defines the headers on the data. The Simple Object Access Protocol (SOAP) and Hypertext Transfer Protocol HTTP are most commonly used for these layers.

All of the type descriptions used in the next section, the description layer, must be expressed using the XML Schema language. The first three, XML Schema, interface description and implementation description are used to pass the messages on to the wire section. Next the policy description layer is used to describe information specific to the service. For example, this may be costs, security requirements, timeouts, etc. The presentation layer where the user interface is managed. Composition and Orchestration describe

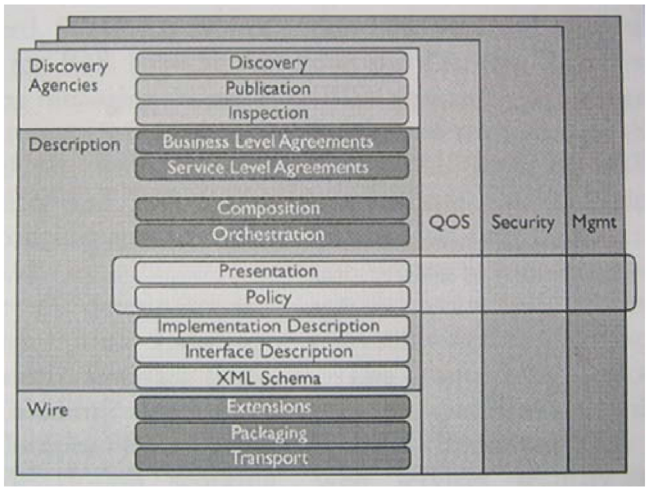

Figure 1: Layers

the interactions between services. For example the composition layer may contain the parent-child relationships between services and the orchestration layer may contain the business processes. The top two layers contain the agreements made between the client and server. Service level agreements layer may contain, for example, the appropriate costs and performance levels that the service should be obtaining. The business level agreements layer holds the agreement between the two partners who will be completing a transaction.

The top section, discovery agencies, is comprised of the technologies that allow the service to be available to a client. The three layers on the right of the stack are the issues that apply to all the sections and layers of the stack. These are quality of service, security and management.

3 Thesis Statement

I will research and compare existing Web service development methodologies.

4 Implementation and Methodology

Comparison of these methodologies will be made mainly through research. Simple working examples and experiments will be used in expressing how these technologies work and which is best to use in certain situations.

5 Research and Writing Timetable

Researching these evolving protocols, development tools and methodologies will being immediately. The first two chapters will be completed by the semester break. Simple examples of these languages and protocols will be created during research. Research and writing will continue through the semester break. This work will be completed by March first with the oral defense to follow.

6 Conclusion

This work on Web service development methodologies will hopefully be a valuable resource for those seeking to develop a Web service. It will help a developer decide which languages, protocols and services are best for their particular situation. It will also hopefully be a resource for those attempting to understand these complex evolving standards.

References

[1] Simple object access protocol (soap) 1.1. *http://www.w3.org/TR/SOAP/*, 2003.

[2] M.-C. Fauvet F.A. Rabhi B. Bentallah, M. Dumas and Quan Z. Sheng. Overview of Some Patters for Architecting and Managing Composite Web Sites. *ACM SIGecom Exchanges*, 3:9–16, June 2002.

[3] Michelle Casagni and Margaret Lyell. Comparison of Two Component Framaeworks: The FIPA-Compliant Multi-Agent System and The Web-Centric J2EE Platform. *Proceedings of the 25th International Conference on Software Engineering*, pages 341–351, 2003.

[4] Dieter Fensel Christoph Bussler and Alexander Maedche. A Conceptual Architecture for Semantic Web Enabled Web Services. *ACM SIGMOD Record*, 31:24–29, December 2002.

[5] Neil Davidson. Testing Web Services. *http://www.webservices.org/index.php/article/articleprint/179/-1/42/*, 2003.

[6] Heather Kreger. Fulfilling the Web Services Promise. *Communications of the ACM*, 46:29–34, June 2003.

[7] Kai R.T. Larsen and Peter A. Bloniarz. A Cost and Performance Model for Web Service Investment. *Communications of the ACM*, 43:109–116, February 2000.

[8] Joline Morrison Mike Morrison and Anthony Keys. Integrating Web Sites and Databases. *Communications of the ACM*, 45:81–86, September 2002.

[9] Gerry Miller. The Web Services Debate: .NET vs. J2EE. *Communications of the ACM*, 46:64–67, June 2003.

[10] M. Papazoglou and J. Yang. Design Methodology for Web Services and Business Processes, 2002.

[11] Mike Papazoglou, Marco Aiello, Marco Pistore, and Jian Yang. XSRL: A request language for Web services.

[12] Paulo F. Pires, Mario Benevides, and Marta Mattoso. Building Reliable Web Services Compositions. *Net.Object Days - WS-RSD'02*, pages 551–562, 2002.

[13] Schahram Dustdar Rainer Anzbock and Harald Gall. Software Configuration, Distribution, and Deployment of Web-Services. *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering*, pages 649–656, 2002.

[14] Aaron Weiss. Microsoft's .NET: Platform in the Clouds. *netWorker*, 5:27–31, December 2001.

[15] Joesph Williams. The Web Services Debate. *Communications of the ACM*, 46:59–67, June 2003.

[16] Composition Management Wscm. Web Service Offerings Language (WSOL) and Web Service.