NOTE TO USERS

Web Services: Framework and Technologies

Yonghan Zhu

School of Computer Science

McGill University, Montreal

January 2004

A Thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of

Master of Science

Your file Votre référence
ISBN: 0-612-98772-8
Our file Notre référence
ISBN: 0-612-98772-8

Acknowledgement

I would like to express my gratitude to Prof. Monty Newborn for his guidance and moral support along the course of the research, which made the completion of this thesis possible. His recommendation offered me an excellent opportunity to deepen the research at IBM Toronto Lab.

Many thanks go to Dr. Terry Lau, Senior Researcher at the Center of Advanced Study (CAS), IBM Toronto Lab. His mentoring and encouragement added notable value to this research.

My grateful appreciation goes to Jennifer Ellis. Her continuous encouragement and favorite helps on editing made this thesis a complete work.

I would also express my gratitude to Mr. Ed Mischkot, Don Bourne, Anthony Jong, Rick Strobel, Andrew Chen, Steve Sun, Justine Yoon, and other people at IBM Toronto Lab for their professional advices to the research.

And, last but not least, my love goes to Kathy Zhang for all her great selfless support and cares, for all the good and bad times we have shared.

TABLE OF CONTENTS

Abstract.....1

Introduction.....3

CHAPTER 1. Web Services Perspectives.....6

1-1. What Are Web Services.....6

1-2. Web Services Concepts And Working Flow.....8

1-3. A Brief History Of Web Services.....14

1-4. Current Web Services Architecture And Components.....17

CHAPTER 2. Web Services Technologies.....21

2-1. Transport Technologies.....21

2-1-1. HTTP.....22

2-1-2. FTP.....24

2-1-3. SMTP.....25

2-1-4. IIOP.....25

2-1-5. BEEP.....26

2-2. XML Technologies.....29

2-2-1. XML Namespace.....32

2-2-2. XML Schema.....33

2-2-3. XML Style Sheet.....34

2-2-4. XML Parser.....34

2-2-5. XML Addressing.....36

2-2-6. XML Query.....37

2-3. XML Web Services Interaction Technologies.....38

2-3-1. XML-RPC.....39
2-3-2. SOAP.....42
2-3-3. WSDL.....47
2-3-4. UDDI.....51
2-4. Web Services Business Specifications.....55
2-4-1. Business Process Execution Language For Web Services (BPEL4WS) 56
2-4-2. RosettaNet.....63
2-4-3. ebXML.....66
CHAPTER 3. Feature Considerations For Web Services.....73
3-1. Interoperability.....73
3-2. Security.....77
3-3. Choreography/Orchestration.....82
3-4. Reliability.....85
CHAPTER 4. Web Services Standards Organizations And Products.....88
4-1. Standards Organizations.....90
4-1-1. World Wide Web Consortium (W3C).....90
4-1-2. Organization for the Advancement of Structured Information Standards (OASIS).....92
4-1-3. Internet Engineering Task Force (IETF).....93
4-1-4. United Nations Center for Trade Facilitation Electronic Business (UN/CEFACT).....94
4-1-5. Web Services Interoperability Organization (WS-I).....95
4-1-6. RosettaNet And Other Business Specification Organizations.....96
4-2. Key Web Services Vendors and Products.....98
CHAPTER 5. Web Services: What It Brings.....100
5-1. Web Services Provides A Growing Publicly Available API Across The Web...102

Abstract

Immediately after its appearance in 1999, *Web Services* has become the hottest topic in the information technology industry. Web Services was primarily fostered by the exponentially growing demand for highly efficient Business-to-Business (B2B) solutions.

Web Services is a highly modularized application level framework. Its fundamental idea is to enable Web applications to interact with each other regardless of platforms, languages or network infrastructure used. This understanding has been accepted and shared by current Web Services vendors, users, professionals and standards organizations. Meanwhile, Web Services concepts, architecture, components, working models and direction are still under debate. This thesis provides an introduction to these topics based on a thorough research across existing materials.

Web Services consists of two groups of technologies. One of these two technology groups is represented by a layered stack on top of network transport layer. This stack contains a number of core Web Services technologies including eXtensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) and Universal Description, Discovery and Integration (UDDI), along with other additional or alternative technologies. The other technologies group is a set of feature specifications including interoperability, security, choreography/orchestration, reliability, etc.

Both of the two groups of Web Services technologies are quickly evolving by adopting new technologies and updating existing standards with newer versions. Currently, above the transport layer, Web Services is a framework fully based on XML technology. Therefore, Web Services is also referred to as *XML Web Services* in some documents.

The first part of this thesis clarifies the architecture and basic concepts of Web Services. The second part provides a complete introduction and analysis of the two groups of technologies stated above, and shows how these technologies work together. The third part of this thesis presents a discussion of the advantages and potential applications of Web Services. It is based on the perspectives of a variety of technical experts and solution designers.

The source materials used by this thesis include technical books, standards organizations documents, major vendors' technical papers and other articles collected from websites, e.g. webservices.org and coverpages.org.

In summary, this thesis provides a thorough discussion of the technologies currently used by Web Services, its development status, advantages and the potential applications. The discussion provides an overall independent vision of this fast evolving new framework by presenting its key technological points.

5-2. Web Services Introduces A Loosely Connected Application Architecture.....103
5-3. Web Services Provides A Solution For Integrated E-Commerce (IEC).....104
5-4. Web Services Offers A Complementary Means For Software Service.....105
5-5. Web Services Enables Grid Computation Across The Web.....107
5-6. Web Services Fosters Flexible Low-cost Worldwide Inter-business Solutions.....108
5-7. Promotes Possibility Towards Semantic Web.....109
CHAPTER 6. Conclusion.....110
REFERENCES.....112
APPENDICES.....123
Appendix A The Difference Between the Internet and the World Wide Web.....123
Appendix B Dave's History of SOAP.....125
Appendix C How ebXML Will Transform the Software Industry.....128
Appendix D Web Services Visionary (Part).....133

Introduction

Web Services, also known as XML Web Services, is a new Internet technology emerged in 1999. Although the basic idea behind it has existed for a long time, its emergence was triggered by and is currently based on the maturity of XML technology.

Today, almost all the major E-business relevant vendors and organizations are collaborating on Web Services development in various fields. The major E-business platforms, including IBM WebSphere, Microsoft .NET, SUN ONE, etc., have provokingly announced solid support to this new framework and its technologies. The major Internet application standards organizations, such as World Wide Web Consortium (W3C) and Organization for the Advancement of Structured Information Standards (OASIS) are actively spending time and effort defining technical standards and their implementation guidelines to fulfill the exponential growth in demand for implementation.

With tremendous technical advantages for implementing distributed applications systems, Web Services is becoming the next wave in the computer industry after the World Wide Web (WWW). The Founder and CEO of Forrester Research Inc., Mr. George Colony, said in the ICT World Forum 2003, "(A new) technology thunderstorm hits every five to nine years and we are due one (Web Services) now" [94]. Bill Gates, the Chairman and Chief Software Architect of Microsoft, also stated in his paper *Microsoft .NET Today* on June 14, 2001 that "XML Web services (are) gaining momentum among developers as the next

generation of Internet-based computing" [14]. The exciting future of Web Services looks irresistible.

By the middle of 2003, a stack of core Web Services standards has already been widely accepted by the industry and implemented into the major E-business products. This stack includes XML and XML Schema, which define the basic format and semantics of the data exchanged, SOAP, which defines the data exchange scheme between Web service provider and Web service requester, WSDL, which provides a technical description of the interaction pattern and message format requirements for each Web service, and UDDI, which specifies a universal means for publishing and discovering Web services.

These core stack technologies are developed and standardized by the most influential Internet standards organizations: OASIS, W3C and Internet Engineering Task Force (IETF). Each of these standards also has a number of other substitution technologies. In addition to these existing technologies, the major Web Services product vendors are either independently or collaboratively developing a variety of new specifications. These new specifications reflect the industry's approach to provide more interoperable, reliable, secure and collaboration-enabled Web services by concerning various features.

Web Services, as a new technology framework, is still in its infancy stage. The understandings of its concepts, structure, working mechanics and developing direction are inconsistent in the industry. This inconsistency remains a critical hurdle to be overcome before Web Services can truly "boom".

One reason for this inconsistency is that the recommendations upon which implementation of core standards are based are constantly being updated. For example, after SOAP v.1.1 had been recommended by W3C and implemented in most Web

Services products for under one year, SOAP v1.2 was released with a significant extension.

Another reason for inconsistency is the incompleteness of the whole Web Services technology stack. Since it is not complete, it is by definition unstable, so vendors choose to develop and adopt various proprietary technologies to implement their own products. Often, even when implementing an identical standard, different vendors take slightly different implementation approaches.

Furthermore, there are still a lot of sustaining technologies to be developed in order to fully explore the advantages of Web Services. But what these technologies should support and how they should realize it is still not clear for the industry.

The purpose of this thesis is to present an updated complete, clear and vendor-neutral understanding of the XML Web Services framework and its technologies. It is based on thorough research of the latest technical specifications, papers and books.

This thesis will present a complete neutral vision of current Web Services perspective, a complete understanding of the Web Services architecture and the latest version of dominant standards and specifications, and an analysis of present and potential application advantages of Web Services.

CHAPTER 1

Web Services Perspectives

1 - 1. What Are Web Services?

Web Services has recently become a frequently used term, representing a new breed of applications. Unfortunately, this term does not yet have a commonly agreed upon definition. Experts, vendors and standards organizations define Web Services by focusing on various aspects.

One of the leading World Wide Web (WWW or Web) technology standards organizations , the World Wide Web Consortium (W3C), defines a Web service as "...a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable 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" [1]. The www.webopedia.com (a popular website that functions as a dictionary for IT terminology) states: "The term *Web services* describes a standardized way of integrating Web-based applications using the XML, SOAP, WSDL and UDDI open standards over an Internet protocol backbone" [2]. And Aaron Skonnard, an instructor and researcher at DevelopMentor Inc., says, "Web Services

represents a new platform on which developers can build the same distributed applications they've always built, but this time with interoperability as the highest priority" [3].

Three Web Services-related terms are currently being used to refer to this new breed of applications by describing different aspects and, however, with some ambiguity: ***Web service***, ***Web services*** and ***Web Services***. "Web service" generally refers to a single application system that provides certain functionalities. "Web services" is a term with ambiguous meanings: sometimes it is used to refer to more than one "Web service", and sometimes it is used to refer to the abstract concept of this new breed of applications. Meanwhile, in some articles, "Web Services" is used to refer to the abstract concept instead of "Web services".

To avoid ambiguity, this thesis uses "Web Services" to refer to the abstract concept of this new breed of applications, uses "Web service" to refer to one single Web Services application system, and "Web services" to refer to more than one Web service.

It is difficult to give a concise definition of Web Services that can be universally accepted by the industry, especially when this technology is still in fast evolving. A conceptual description of Web Services is given below as the basis of further discussions in this thesis:

> Web Services is a highly modularized application level framework that enables distributed applications to publish self-descriptions, discover interaction manners of others and interact with the discovered applications. Within this framework, each application can be independently developed in any language, running on any platform and deployed in any network infrastructure. Currently, Web Services

consists of a core stack of several commonly agreed on standards including *eXtensible Markup Language* (XML), *Simple Object Access Protocol* (SOAP), *Web Services Description Language* (WSDL), *Universal Description, Discovery and Integration* (UDDI), etc., and other alternative and adds-on technologies.

A Web Services framework is typically implemented by employing a full stack of standards and following their implementation guidelines if available. Within such a framework, Web-based applications are able to intelligently coordinate and collaborate on various tasks. These tasks can be Business-to-Business (B2B) transactions, grid computations, or other tasks that need collaboration among multiple applications in a distributed environment.

Currently, a Web Services application can be written in any popular Object Oriented (OO) language including Java, C#, C++ or Visual Basic, some Procedure Oriented languages such as C [4], and even some script languages such as Perl [5]. Such an application hence can be deployed on any platform, such as IBM mainframe, AS/400, UNIX, Windows, Linux, etc. As long as a common standards stack is adopted, Web Services applications can either provide Web services to or utilize the Web services from others.

1 – 2, Web Services Concepts And Working Flow

Although many different, and sometimes inconsistent, definitions have been given to Web Services, the basic concepts and rules within this framework have been recently clarified and shared by the parties who are developing it. These concepts and rules are given below according to W3C's working draft of Web Services Architecture of August 8, 2003 [1].

A *Web service* is an abstract set of functionality provided by a *legal entity*, which may be an organization or an individual. Universally, each Web service is identified by a *Uniform Resource Identifier* (URI). A Web service conceptually consists of two sets of actions: message exchange and data manipulation. The *message exchange* refers to receiving and sending messages, while the *data manipulation* refers to the set of functionality the Web service provides. The data manipulation may contain a number of different functions towards the same data subject. These functions are generally referred to as *operations* of the Web service they belong to.

[**URI, URL** and **URN:**

The term *URI*, which stands for Uniform Resource Identifier, "is a compact string of characters for identifying an abstract or physical resource" on the Web [6]. Unlike other Web standards that generally have alternatives, URI is the only official technology for Web addressing/naming today. It makes the Web resources recognizable and accessible via various naming schemes and access methods.

The term *URL*, which stands for Uniform Resource Locator and was defined in RFC1808 [7], was the most widely adopted technology for locating Web resources. After the RFC2396 [6] was officially released, URL became informal but still a widely used standard to locate the Web resources with popular URI schemes, such as http, ftp, mailto, etc.

URN, stands for Uniform Resource Name, is another subset of URI. It has two distinct but related meanings: a persistently available URI with institutional commitments, and a persistent location-independent Web resource identifier scheme defined in RFC2141 [8].

The relationship among URI, URL and URN is illustrated in Figure 1-1.

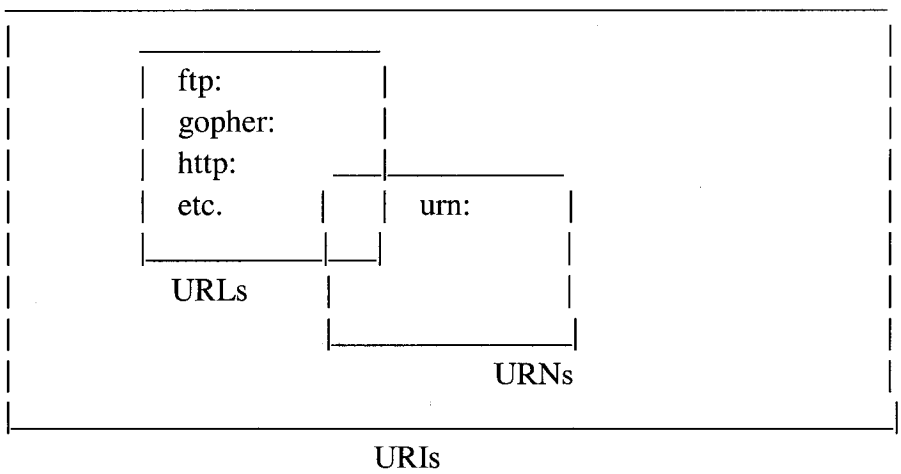

Figure 1-1 URI, URL and URN

(Source: *Naming and Addressing: URIs, URLs, ...* at http://www.w3.org/Addressing/ [9])

]

In a Web Services framework, a *legal entity* refers to an individual or an organization that has the right to form and execute interacting agreements with other legal entities. A legal entity that provides a Web service is called a service *provider entity*. A legal entity that wishes to utilize a Web service is called a service *requester entity*. The interacting agreements between two or more provider entities and requester entities are referred to as *contracts*.

A physical entity (generally a piece of running software) that implements the message exchange for a Web service is called an *agent*. For a Web service, both service provider entity and services requester entity have their own agents. A *service provider* is an agent that either implements the functionality by itself, or acquires the result by coordinating with other programs. A *service requester* is an agent that composes and sends out request messages to a specified Web service, and receives and passes up the result. The service

requester agent if required, also accomplishes searching available Web services across the Web, analyzing and selecting the best choice, etc.

One significant advantage of Web Services is that a service requester can intelligently search for available Web services across the Web, analyze the search result and make a choice. It presents significant progress towards highly automated B2B systems.

In order to realize this advantage, a service provider needs to provide a machine-processable document that describes the mechanics of using its Web service to potential service requesters. These mechanics refer to the Web service's message exchange pattern, message format, data types, communication protocols, etc.. This document is called a *Web service description* (WSD) and is normally described in WSDL. Besides WSD, a commonly understandable contract that specifies the effects and requirements of invoking a Web service may also be required. This contract is called the Web service's *semantics*. Web service semantics can be specified either in human languages or in machine-readable languages, either oral or written.

Typically, Web service agents are implemented in OO programming languages such as Java, C# and C++. A Web service provider normally runs as a public object (referred to as an **operation** in WSDL), which consists of a number of methods (each of which is referred to as a **method** in WSDL). Each method implements a manipulation function of the invoking data contained in the request message. The function may be processing the invoking data in the service provider's back-end system, or even organizing a series of other Web services to coordinate on certain tasks as requested.

Some Web services function independently. They simply receive requests, process the invoking data and respond with the results. Other Web services are designed to

collaborate on accomplishing certain tasks. Web service *choreography* refers to the defining of the sequence and conditions, under which multiple cooperating Web services interact with each other to achieve a certain function.

Currently, most Web services are only available to their pre-authorized service requesters, such as service requesters within the system or service requesters from business partners' systems. Some Web services are also publicly available to all users on the Internet, such as those free Web services provided by Google at http://www.google.com/apis/.

A simple Web Services working flow is illustrated in Figure 1-2:

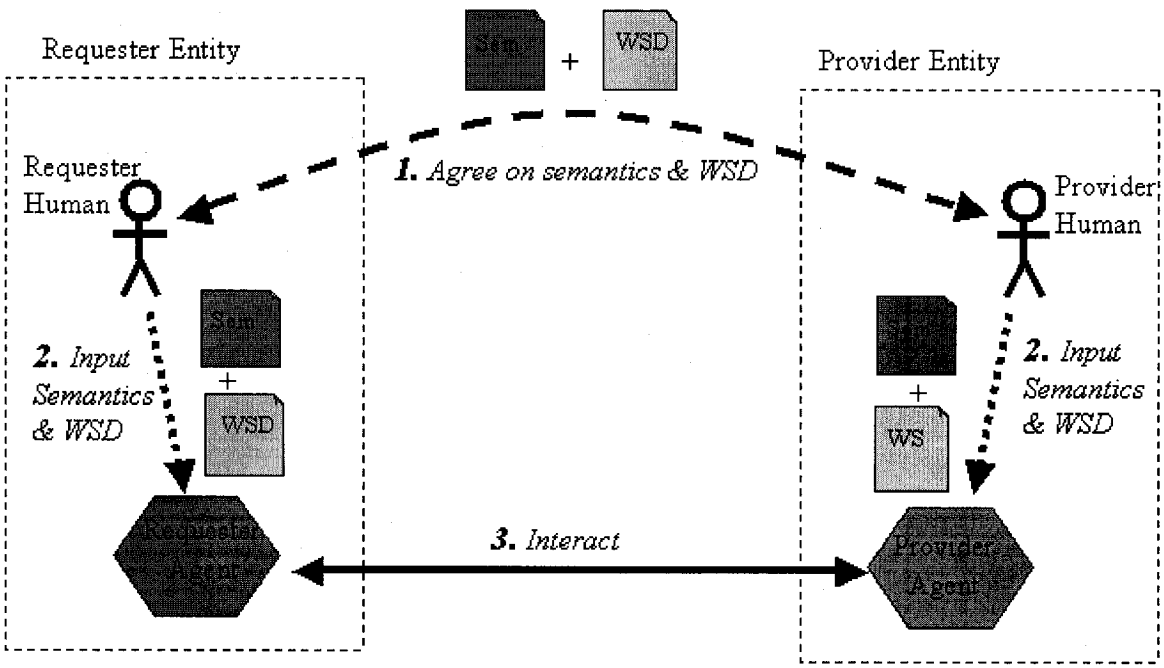

Figure 1-2 Web Services working flow

(Source: *Web Services Architecture, W3C Working Draft8 August2003* [1])

Before using a Web service, a service requester entity and the service provider entity need to negotiate and agree on a set of contracts defining the semantics and WSD of using the Web service, and store the description of these contracts at locations accessible for both parties. Today, these descriptions are generally written in WSDL and stored in UDDI registry servers.

To make use of a Web service, the service requester checks and confirms that the semantics of the Web service satisfy its requirements. It then retrieves the WSD description, figures out the message format and interaction pattern, and sends out a requesting XML message to invoke a specific method of the service provider. In some advanced scenarios, the service requester may even need to search across the Web for all available service providers and make an intelligent choice. After the request is processed, the service provider responds with a result, which may sometimes be a failure message, to the service requester in the format defined in WSD.

The most prominent advantage of the Web Services framework is its interoperability. That is, as long as the two agents of a Web service are able to exchange messages by following the industry standards and implementation guidelines, they can smoothly work together. The agents can be programmed in any languages, running on any platforms within any network infrastructure.

Currently, Hyper Text Transfer Protocol (HTTP) is the mostly employed protocol for transferring data between Web service agents. All Web service exchanging messages are presented as XML documents. Web service requesters and service providers generally adopt SOAP or eXtensible Markup Language based Remote Procedure Call (XML-RPC) to interact with each other. WSD is usually described in WSDL. The commonly agreed upon means to publish and discover valid Web services across the Internet is UDDI. In

some cases, if the service requester has full knowledge of the semantics and WSD of a service provider, UDDI and WSDL can be omitted.

These standards and their implementation guides are still under refinery by the most influential Internet standards organizations including W3C, Organization for the Advancement of Structured Information Standards (OASIS) and Web Services Interoperability Organization (WS-I). Most of the major Web Services vendors, such as IBM, Microsoft, Sun, HP and BEA, have announced their support for these standards as the Web Services *core stack.*

1 - 3. A Brief History Of Web Services

The origin of the Web Services idea can be traced back to some old science fiction stories. In these stories, the "future" world was described as so automatic that machines (programs) would be able to intelligently communicate and collaborate to accomplish sophisticated tasks without human intervention.

To achieve this goal, there are a few crucial technological requirements. First, a network must be set up to connect the collaborative machines. Second, an interoperable framework must be defined on top of the network to enable interactions among programs. Third, the collaborative programs must be able to intelligently understand the exchanged information and make correct decisions.

In the late 1980s, the Internet started to provide a super network connecting individual machines and Local Area Networks (LANs), which satisfied the first technological requirement described above.

A few years later, the Web was invented with two fundamental protocols - HTTP and HyperText Markup Language (HTML). The Web provides a convenient and user-friendly way for humans to interact with Web applications through browsers. It dramatically improved the usability of the Internet and stimulated the booming of the electronic economy [11]. It made progress towards the second requirement but is still far from satisfying it.

[Internet vs. World Wide Web:

Many people use the term *Internet* and *World Wide Web* (*Web*) interchangeably, which is not correct. According to Webopedia.com, "The *Internet* is a massive network of networks, a networking infrastructure" and the Web "...is a way of accessing information over the medium of the Internet" [12]. Defined by W3C, "The World Wide Web (known as 'WWW', 'Web' or 'W3') is the universe of network-accessible information, the embodiment of human knowledge..." built on top of the Internet [13].

]

When an increasing number of Internet-based business systems had been deployed, the demand of improving their interoperability grew. Normally, different Internet application systems are developed in different languages and run on different platforms within different network infrastructures. In this situation, too much incompatibility exists among the various Internet applications. People had to either get involved or employ very rigid application interfaces to enable data exchanges. This constraint frustrated the electronic economy's further evolution.

In the late 1990s, when the XML technology became mature, the industry was delighted by the agreement on a pragmatic new technology for data exchange that is based on the

XML technology - the XML-based Web Services framework. It enables Internet applications to exchange data and collaborate on sophisticated tasks regardless of what platforms they are running on or what language they were developed with.

Web Services has drawn the attention of most major E-business vendors and standards organizations since it first appeared. The most influential Internet standards organizations, including W3C, OASIS, Internet Engineering Task Force (IETF), etc., have started making standards, rules and implementation guides for Web Services. The key E-business platforms in the market, including IBM WebSphere, Microsoft .NET, HP OpenView, Sun One, BEA Web Logic, etc., have also announced their support to this XML-based Web Services framework. Furthermore, some vendors and organizations have co-founded new organizations specifically for Web Services. These newly founded organizations include WS-I and Webservices.org.

Bill Gates, Microsoft's chairman and chief software architect, stated in his article *Microsoft .NET Today* of June 14, 2001: "With XML Web services gaining momentum among developers as the next generation of Internet-based computing, it's time to deliver a platform that makes it simpler to build these solutions and provides a reliable framework for integration and interoperability... Microsoft's platform for building, deploying, operating and integrating XML Web services is .NET" [14].

Although many Web Services vendors have claimed their support for a few common technical and business standards, such as SOAP and WSDL, the Web Services implementations are still somehow vendor-specific and sometimes having trouble interoperating with others. The industry has already realized this problem and started to collaborate on developing standards implementation guidelines. [15] WS-I is the organization formed by most of current Web Services vendors to improve Web Services

interoperability among different platforms, languages, and applications. Nowadays, its effort is focusing on SOAP, WSDL, UDDI and Web Services security.

1 - 4. Current Web Services Architecture And Components

The Web Services architecture can be described in various dimensions. This section describes it in two typical dimensions: the layered communication model and the interactive application model.

Figure 1-3 illustrates the static communication model. It presents a layered stack of available Web Services interoperation protocols. From level 2 up, each protocol is supported by any one protocol in its immediately lower level. All protocols within a level provide similar interfaces and functions to their immediately upper level and are alternative to each other.

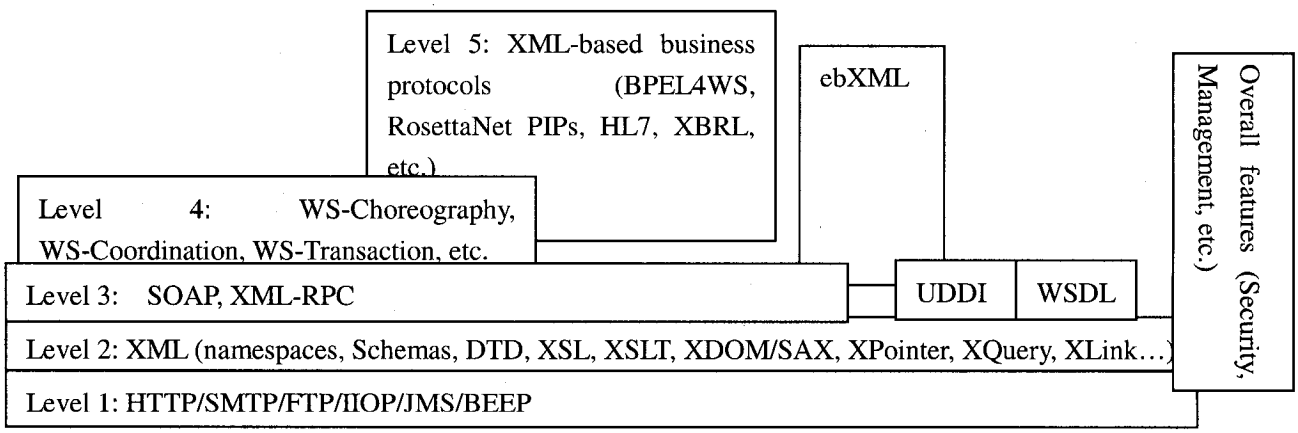

Figure 1-3 Web Services Architecture: layered communication model

To implement a Web service, one protocol of each level has to be employed from level 1 to level 3. The standards in level 4 can be used to coordinate a number of Web services to accomplish sophisticated tasks. Level 5 consists of a variety of business protocols applicable to implement B2B processes for various industries. Besides, there is an eXtensible Markup Language (ebXML), which intends to define a full package of specifications for developing B2B systems with the Web Services idea. These business-oriented specifications define counterparts of the function-oriented standards from level 3 to level 5. ebXML can optionally adopt SOAP and UDDI for business process interaction and service discovery. The standards from level 3 to level 5 in Figure 1-3 are mostly specified by W3C and OASIS while ebXML is solely developed by the United Nations Center for Trade Facilitation Electronic Business (UN/CEFACT). The overall features shown in Figure 1-3 represent the feature considerations that affect all levels in the stack, such as Web Services security and other management features.

The standards in level 1 to level 3 are used to build up the basic Web Services framework. WSDL and the UDDI can respectively be employed for describing and discovering Web services. In many articles, WSDL and UDDI are considered very useful (but not necessary) for improving Web services interoperability.

The bottom level of Web Services stack is the Transport level. It consists of the protocols to be adopted for transporting messages between Web Services agents. HTTP, which is the only data transfer protocol for the Web and the dominant data transfer protocol on the Internet, is currently the most widely supported Web Services transport protocol. A new alternative protocol of HTTP is Blocks Extensible Exchange Protocol (BEEP), which is an IETF recommendation (RFC3080) with a number of advantages. It "...is a new Internet standards-track protocol framework for new Internet applications" [16]. It requires only 30 bytes overhead while HTTP requires 100-300 bytes overhead for a

typical message transfer [10].

The interactive application model is illustrated in Figure 1-4. The protocols indicated in this figure are the most widely accepted XML-based Web Services standards. Till today, most Web services are still deployed for internal usage only, i.e., to be used inside a business's private system or within a number of business partners' systems. It is still in the experimental stage for Web services to be publicly available on the Internet. Hence, the Web Services registry and discovery service is not widely adopted yet. In Figure 1-4, step 1, 2 and 3 represent the processes related to Web services registration and discovery, while step 4 and 5 represent a simple interaction between a Web service provider and a Web service requester.

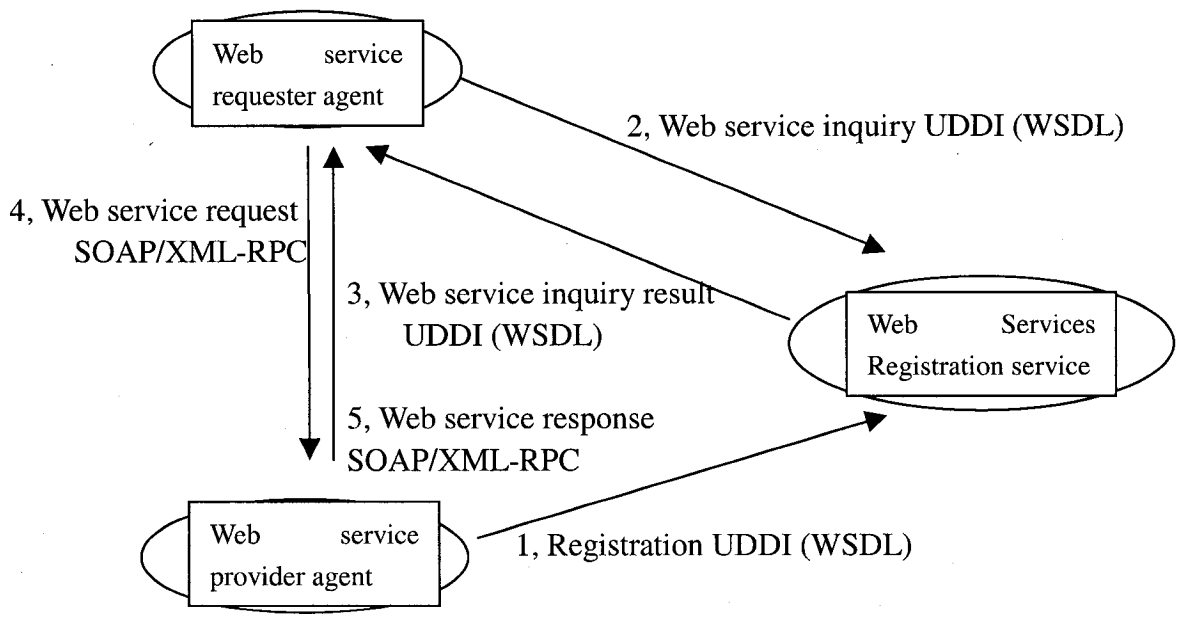

Figure 1-4 Web Services architecture: interactive application model (a simple example)

In a fully implemented Web Services environment, a Web service needs to be registered onto a UDDI (or other commonly agreed on registry protocol) server (step 1). The registry server stores the information of the Web service provider entity, the Web service's purpose, functionality and consequence, interacting mechanics, and all other useful information into its database. This information can be discovered by valid service requesters. When an authorized service requester needs to make use of a Web service, it first queries the reachable registry server's database for all the available Web services that meet its requirement (step 2). Each registry server then returns a result message containing detailed WSD and semantics of all the valid services to the service requester (step 3). After analyzing the acquired descriptions automatically or manually, the service requester selects one service provider and sends out a request message according to the WSD of this Web service (step 4). After the service provider completes the process, it returns the result to the service requester (step 5).

Today, Web Services choreography, security, reliability and other feature considerations are evolving quickly. Thus the actual process of steps 4 and 5 may become very complicated. These additional considerations allow Web services to be executed in collaboration and add on security, reliability and other features to guarantee higher service quality.

Web Services have attracted the attention of the E-business industry. A core stack consisting of a number of institutionally agreed upon standards has already been integrated into most Web Services products. Meanwhile, an increasing number of technologies, which are either necessary complements or nice-to-have add-ons to the Web Services core stack, are under collaborated development to meet a growing demand. The following chapters will introduce both the core stack standards and the new technologies of Web Services framework.

CHAPTER 2

Web Services Technologies

The discussion in this chapter is organized in a layered structure as shown in Figure 1-3.

2 - 1. Transport Technologies

The basic components of Web Services are the alternative transport technologies shown in level 1 in Figure 1-3. Today, the widely supported transport technologies for Web Services are HTTP, Simple Mail Transportation Protocol (SMTP), File Transfer Protocol (FTP) and Internet Inter-ORB Protocol (IIOP), which all support SOAP and XML-RPC. These standards are mature, whereas BEEP is a new transport technology for Web Services with a number of advantages. BEEP was released as a standard in 2001 and is expected to be more efficient for Web Services than the other transport standards.

HTTP is the only transport protocol for the Web and the dominant transport protocol for the Internet. "By 1998, HTTP accounted for over 75 percent of the traffic on Internet backbones dwarfing other protocols such as e-mail, file transfer, and remote login" [17]. Naturally, HTTP became the most widely adopted transport protocol for Web Services.

The interactions between Web service requesters and Web service providers can be

categorized into two types: RPC-Oriented and Document-Oriented [18]. RPC-Oriented interactions refer to real-time and synchronous Web service interactions that "...take the form of a method or a procedure call with associated input and output parameters" [18]. Document-Oriented interactions refer to Web service interactions behaving like a batch job. The service requester submits a request message to a queue of asynchronous processing and receives the result once the job is completed.

Naturally, HTTP became the major transport protocol for RPC-Oriented Web service interactions, which is joined by IIOP and BEEP. FTP and SMTP are the major transport protocols for Document-Oriented interaction.

2 – 1 - 1. HyperText Transfer Protocol (HTTP)

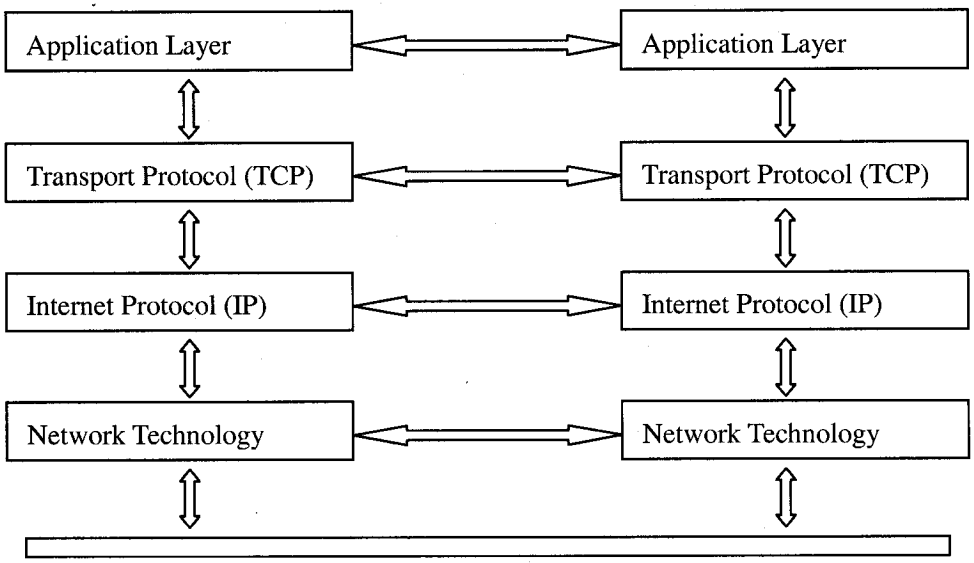

Figure 2-1 Internet communication system

[Source: Stephen A. Thomas, *HTTP Essentials* [17]]

In 1990, the father of the World Wide Web, Berners-Lee, along with Robert Cailliau, designed HTTP to enable communication between Web servers and browsers. The Internet is a multi-layered communication infrastructure illustrated in Figure 2-1. HTTP is the most popular communication protocol in the application layer of today's Internet, which is built on top of TCP.

"The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers" [19].

IETF has developed two versions of HTTP, which are both being used today: version 1.0 and version 1.1. They are defined in RFC1945 and RFC2616 respectively.

[Complementary to HTTP, a number of protocols have been developed recently: Draft Standard RFC2617 – HTTP Authentication: Basic and Digest Access Authentication; Information RFC2145 – Use and Interpretation of HTTP version numbers; Proposed standard RFC2109 – HTTP State Management Mechanism.]

Like many other communication protocols, HTTP defines two distinct parties – a *client* and a *server*. A Web-browsing PC can be seen as an HTTP *client* and a website hosting system is a typical HTTP *server*. Only a client can initiate a communication, by sending a request to the server. The server then sends back a response to the client with a result of the request.

HTTP has only defined a few frequently used standard request operations, which are

called *methods* in RFC2616 [19]:

GET: Retrieve data resource (e.g. Web page) from the requested URI

HEAD: Retrieve information about the specified data source URI

POST: Provide subordinate with the request to retrieve result from the Web server

PUT: Upload enclosed data entity to the requested URI

DELETE: Delete objects specified by the requested URI

TRACE: The server sends back the request message to the client as it received

OPTIONS: Obtain a description of what the request URI can provide

CONNECT: A reserved method for use with a proxy that can switch into a tunnel

HTTP is a stateless communication protocol in the application layer. Each request-response operation is independent. It saves a possibly tremendous cost for the server, which does not need to keep an ever-increasing resource for tracking the communications. Where it is sometimes necessary to keep track of a series of related requests from the same client, the *cookie* technology is adopted.

2 – 1 - 2. File Transfer Protocol (FTP)

FTP stands for File Transfer Protocol. This is a popular, reliable and efficient file transfer model over the Internet. Like HTTP, it also works in client-server structure. However, in FTP, the objects to be transferred are files to be stored in the target system's storage for indirect or implicit usages, while in HTTP, the objects to be transferred are primarily content to be directly used by the client's operations. An FTP server is generally referred to as a File server, while an HTTP server is generally referred to as a Web server.

The first proposed file transfer mechanism, which is considered the original version of FTP, was defined in IETF RFC 114 in 1971. The current version of FTP is defined in

IETF RFC 959 [50].

2 – 1 - 3. Simple Mail Transfer Protocol (SMTP)

SMTP, stands for Simple Mail Transfer Protocol, is the dominant electronic mail transfer protocol on the Internet. It enables mail relay across transport service environments. A transport service forms an Inter-Process Communication Environment (IPCE), which may contain one or multiple networks or a part of a network. Processes are able to communicate with each other through mutually agreed IPCEs. Electronic mail is one way to accomplish this kind of communications.

SMTP is defined in IETF RFC 821 [51]. Within this standard, it officially specifies supports to a number of popular transport protocols: TCP, NCP, NITS and X.25.

2 – 1 – 4. Internet Inter-ORB Protocol (IIOP) [52]

IIOP stands for Internet Inter-ORB Protocol. It is a communication protocol developed by the Object Management Group (OMG).

In OMG's Common Object Request Broker Architecture (CORBA), objects communicate with each other through Object Request Brokers (ORBs). IIOP is the common communication protocol in CORBA that enables ORBs, which may be developed with different vendors' products, to exchange messages with each other over TCP/IP connections such as the Internet. IIOP is the basic required transport specification for implementing CORBA systems.

IIOP is a protocol with client-server infrastructure. It can transport integers, arrays and

other more complicated objects in addition to text. It is defined in the basic CORBA specification, which is generally referred to as CORBA/IIOP. CORBA/IIOP specification can be found at http://www.omg.org/technology/documents/corba_spec_catalog.htm. Its latest version is 3.0.2.

2 – 1 - 5. Blocks Extensible Exchange Protocol (BEEP) [53][84]

BEEP stands for Blocks Extensible Exchange Protocol. It is a new transport technology on top of TCP/IP at the same level as HTTP, published by IETF in March 2001 in RFC3080: The Beep Core and RFC3081: Mapping The BEEP Core Onto TCP. Both of these two RFCs were solely authored by Dr. Marshall T. Rose, who has authored over 60 RFCs.

BEEP holds several advantages over HTTP for Web Services: while transport overload is becoming a serious consideration for HTTP, BEEP reduces its overload to a very low level; HTTP is a stateless protocol, whereas BEEP keeps the conversation states that makes it a better solution for state-oriented communication schemes; HTTP is a synchronous protocol, whereas BEEP supports both synchronous and asynchronous communications; while BEEP defines the transport infrastructure, it employs XML for the data structure it transports, which makes BEEP more natural to work with XML. Along with other advantages, BEEP is expected by many professionals to replace HTTP to be the major transport protocol for XML-based frameworks, such as Web Services. [53]

Unlike HTTP, which provides a client-server data transfer model, BEEP is peer-to-peer that allows simultaneous and independent message exchanges. BEEP messages are exchanged as Multipurpose Internet Mail Extensions (MIME) content.

Using BEEP, when a connection is initiated between two parties, a session is established. A session can launch a *channel* for each specific interaction. Each channel is identified by a digital identifier starting from "0" and owns a specific *profile* defining its context. Channel "0" is the reserved management channel launched while the session is first established. It is used to negotiate the setup and interactions rules of other channels.

The BEEP profiles can be categorized into two types: tuning profiles and data-exchange profiles. Tuning profiles are set up at the start time of the session and affect all other channels. It includes the channel management profile, which is used by channel 0, and the optional Transport Layer Security (TLS) transport security profile and Simple Authentication and Security Layer (SASL) family of profiles. Data-exchange profiles are used for establishing data transport channels. These profiles are defined by the application protocol designers, such as the designers who designed SOAP over BEEP.

Figure 2-2 shows the relations between session, channel and profiles.

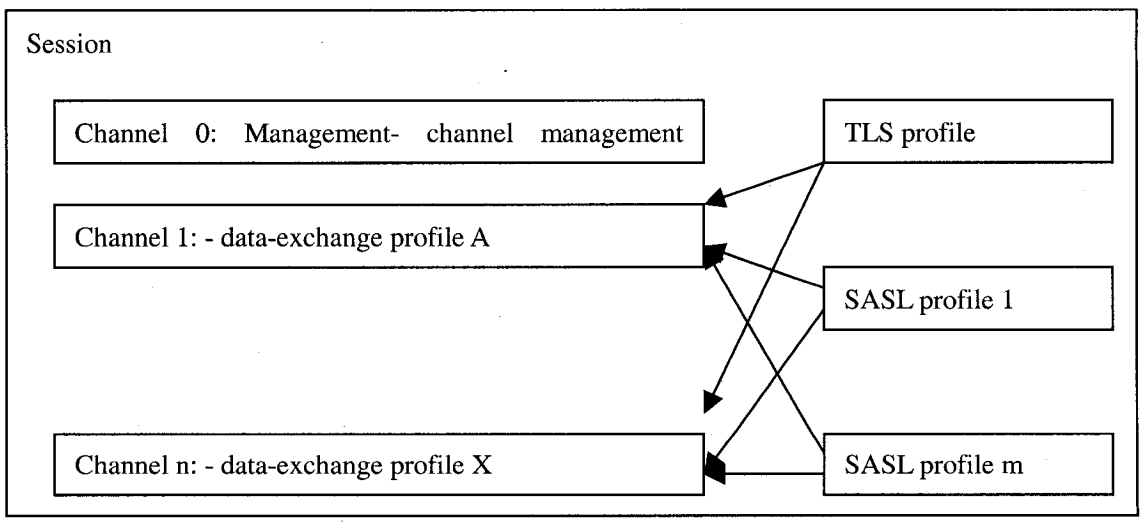

Figure 2-2 BEEP session, channel and profiles

Though BEEP is a peer-to-peer protocol, it would be convenient to identify the roles of the two parties in a BEEP communication. RFC 3080 specifies the roles of the peers at different stage of the session: at the initialization stage, the peer initializing the session is labeled *initiator* and the other is labeled *listener*; at the continuation stage, the peer starting the exchange is labeled *client* and the other is labeled *server*.

BEEP allows three styles of exchanges: MSG/RPY, in which the client sends a "MSG" to request the server to perform a certain task and the server sends back a "RPY" with the result while the task is accomplished; MSG/ERR, in which the client sends a "MSG" request to the server, the server replies an "ERR" message without performing the task; and MSG/ANS, in which the client sends a "MSG" to request the server to perform a certain task, and the server then sends back zero or a number of "ANS" messages containing the result with a "NUL" message to end. [54]

BEEP uses MIME to frame the content it transports, which enables its flexibility and capability to work with XML-based application protocols. Although not pointed out in formal specification documents, BEEP is an ideal transport protocol for ebXML since ebXML also employs MIME to construct its messages (see section 2-4-3 for details about ebXML).

BEEP gives a cautious security consideration in its core design, making it complementary with more consistent and integrated secure communications than other protocols.

While XML is playing a more and more important role in E-business and Web-based applications, BEEP is predicted to be a powerful competitor of HTTP in the near future. It is currently supported by Java, C++, C and Tcl.

2 - 2. XML technologies

"HTML – the HyperText Markup Language – made the Web the world's library. XML – the Extensible Markup Language – is its sibling, and it is making the Web the world's commercial and financial hub." – Charles F. Goldfarb, the father of Standard Generalized Markup Language (SGML) [20].

Charles F. Goldfarb invented SGML in 1974 while he was working for IBM. He led a team of hundreds of experts, spending twelve years finishing current SGML specification - ISO 8879. (The story is given in http://www.sgmlsource.com [21]) Since the Web appeared, SGML has brought out most of the Web language standards that represent structured data and documents. HTML and XML are the two most famous representatives of these languages. They are both subsets of SGML and fully compatible with it. While HTML tells how to display data on the Web, XML tells what the data is and what it means.

While creating SGML, the authors obeyed three basic principles:

- It must have a common data representation: markup
- The markup should be extensible
- There must be a mechanism for describing rules for document types

XML fully inherited these principles.

There are some differences between the characteristics of XML and HTML. First, XML is a "formal" language whose rules must be strictly followed. Otherwise the XML document cannot be parsed or represented at all, i.e., all right or none right. HTML is somewhat more "informal" in that its processors (browsers) will parse and represent whatever it can understand and ignore the rest. Second, XML is used to transfer data itself

and its structure, while HTML is transferring the information of how to display the data. Third, although "XML and HTML share a common tag-based structure, but HTML has a fixed vocabulary of tags with standardized meanings. XML tags, on the other hand, have no predefined meanings. You define your own syntax for describing data" [22].

Figure 2-3 illustrates the relationship between XML and HTML.

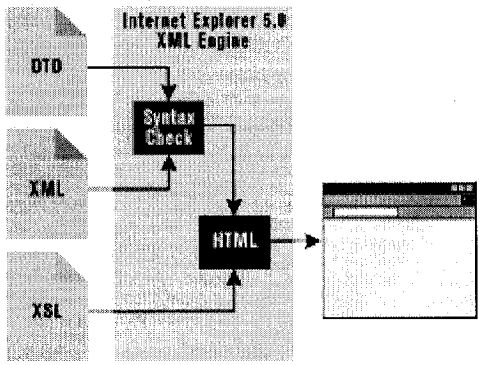

Figure 2-3 XML and HTML

(Source: Dino Esposito, *XML Language*, Microsoft Internet Developer [22])

Figure 2-4 shows W3C's vision of the "next generation of the Web".

As shown in Figure 2-4, XML is the foundation of Web Services technology. It not only defines the format of all Web Services messages (XML documents) transferred between service agents and registry servers, but also provides the language to describe Web service definitions, facilities and all other features. Furthermore, XML is becoming a fundamental technology for tomorrow's Web, like HTML for today's Web.

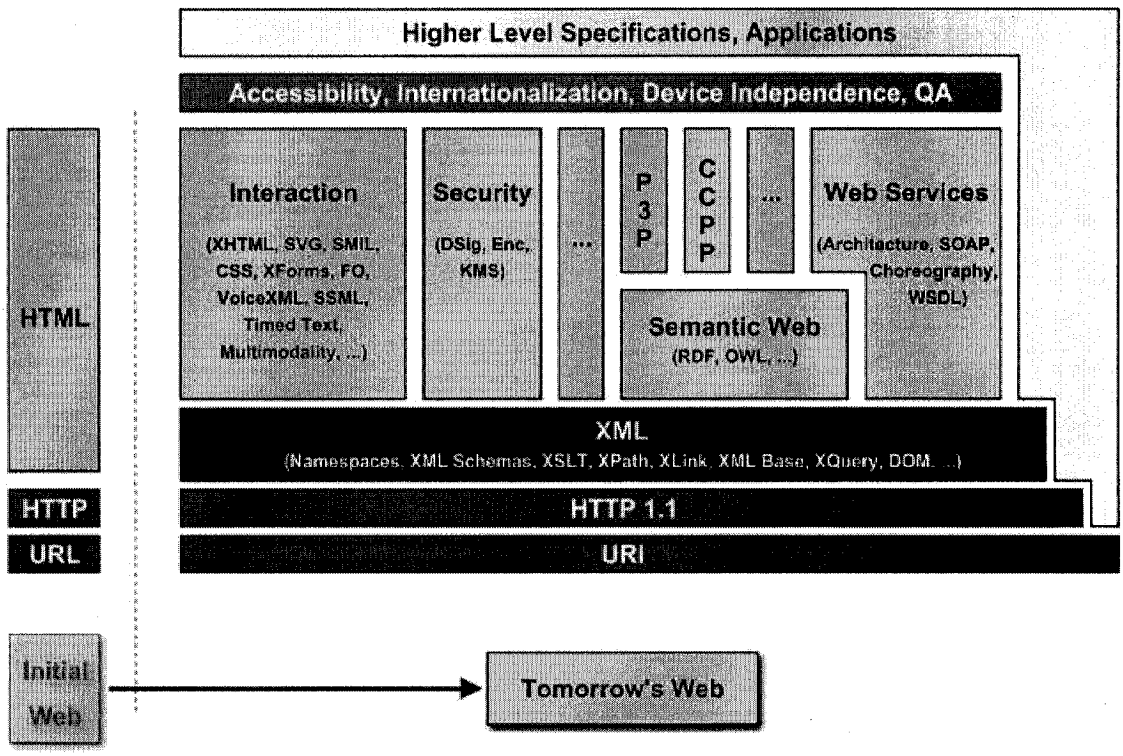

Figure 2-4 Web Architecture for today and tomorrow

(Source: *About the World Wide Web Consortium (W3C)* [23])

XML document structure can be understood in two dimensions: the *physical structure* (also known as *linear structure*) and *logical structure* (also known as *tree structure*).

In Object Oriented technology, an object can be treated as a set of well-organized components. While an XML document is seen as an object, the logical relationship of its components is viewed as its *logical structure*. The representations of the components in this logical structure are called *elements*. The elements construct a tree structure with only one root element.

On the other hand, physically an XML document is represented by a stream of characters and markups, which is called its *physical structure*. A program (*parser*) is required to

read through this stream, parse the markup and pass the data to applications. Such a stream is potentially organized in multiple "pieces" of text. These piece-of-text constructs are called *entities.*

XML is fully specified in Extensible Markup Language (XML) 1.0 (Second Edition) [24] as an official recommendation of W3C.

In order to fully exploit the XML advantages, a number of popular XML technologies and tools have already been or are being developed by various institutions. These technologies are becoming widely used across the Web.

2 – 2 - 1. XML Namespace

A very important scheme introduced by XML is *Namespace*. It brings many benefits to Web Services. "A namespace is a set of names in which all names are unique" [25].

While more and more application systems are deployed across the Internet, the amount of interactions among applications increases. Therefore it becomes very difficult to keep the uniqueness of variable names from different resources. Variables from different resources may possess the same name but carry completely different meanings.

It is confusing for interacting applications, e.g. Web service agents. XML experts figured out a proven efficient way to solve this problem – adding a prefix string, which indicates a context where the variable name belongs to, to the variable name. This method enables interactive applications to tell one variable from another even when they hold an identical name. This prefix string represents the namespace where this variable name is defined.

For Web Services, a name space is represented by a URI. The namespace URI is just a unique name that represents a specific namespace in the Web; it may not really relate to any physical Web resource. The XML namespace was announced by W3C as a recommendation on January 14, 1999 [26].

2 – 2 - 2. XML Schema

XML is a Meta language. It can define a document type (called a schema) for each XML document by declaring a specific document structure and the data types it uses. Currently there are two ways to define XML document types – *Document Type Definition* (DTD) and *XML Schema Definition Language* (XSDL). DTD is the default document type defining mechanism specified within the XML specification. XSDL is a newer add-on mechanism that supports more sophisticated definitions. XSDL was first officially recommended by W3C on May 2, 2001. Its current version is 1.1.

The concept *schema* here is like the schema to define a database. Logically, a database file can actually be treated as a well-structured document. In some circumstances, an XML document is also considered a database file.

The normative XML schema specification currently contains two parts: *XML schema part 1*: Structures [27] and *XML schema part 2*: Datatypes [28]. There is also a non-normative easily readable description of the XML schema, *XML schema part 0*: *Primer* [29]. A complete information collection about XML schema can be found at http://www.oasis-open.org/cover/schemas.html.

2 – 2 - 3. XML Style Sheet

A style sheet is a file describing how to display a certain data document. The popular style sheet specification before XML is Cascading Style Sheet (CSS), which "is a simple mechanism for adding style (e.g. fonts, colors, spacing) to Web documents" [30]. XML has developed a specific language to define XML style sheets, which is called *XML Stylesheet Language* (XSL). Each XSL style sheet is used to describe a certain type of XML document and each XML document type can have multiple style sheets for different purposes. XSL is fully compatible with CSS.

XSL is specified by W3C [31]. Its specification consists of three parts: XSL Transformations (XSLT), XML Path Language (XPath) and XSL Formatting Objects. XSLT is a subset of XSL that enables transformation from XML documents into any other document types. XPath is an expression language used by XSLT to access or refer to parts of an XML document. It is also used by XML Linking (XLink). XSL Formatting Objects is a vocabulary for specifying XML formatting semantics.

XSL is also an extensive language. All the syntaxes used by XSL belong to XML.

2 – 2 - 4. XML Parser

An XML message is also seen as an XML document. It is transferred on the Web as a well-formed character stream. When an XML message is received by an application, e.g. a Web service agent, it is physically just a stream of data and XML tags organized in a certain sequence. The recipient application thus needs to understand the logical document structure and the data contained in this physical character stream.

To achieve this goal, there are two approaches: either let the application read through the document and retrieve what it needs by itself, or employ a tool to parse the document and transfer the parsed message structure along with the data into the application. Apparently the first way may be too complicated and leaves too much tedious work to application development and maintenance. The second approach enables developers to concentrate on the application logic and makes it easier to modify message formats and application logics. Hence it is more attractive for today's XML application development. Those tools used to parse XML documents are called XML parsers.

There are a number of available XML parsers. Logically each of these parsers is a tool to parse XML documents in a specific approach. Physically they are a variety of APIs provided by various programming languages.

A popular XML parser API is *Document Object Model* (DOM). DOM "is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents" [32]. It is currently supported by Java, C#, C++, Visual Basic, Perl scripts, Python, and is a built-in part of Windows 2000. It contains three parts: Core, HTML and XML. The Core part defines general parsing rules for all structured documents. The HTML and XML parts respectively define additional parsing rules for HTML and XML documents.

Another popular XML parser is *Simple API for XML* (SAX). It was developed by an informal group of participants in the XML-DEV mailing list. The XML-DEV mailing list comprises an open group of people who voluntarily participate in code development, protocols and specifications creation, and other contributions to XML implementation and development [34]. SAX was initially the first XML parser in Java and used to be a Java-only API. Today, it starts to be implemented in other languages. The latest version of

SAX is SAX 2.0.1. The official web site of SAX is http://www.saxproject.org/ [33].

XML DOM is document-oriented while SAX is event-oriented. XML DOM reads in the whole XML document and outputs all the parsed data in one tree structure that allows random access. SAX treats the XML document as a stream of events, where each event represents a handler function call while a chunk of XML syntax is recognized. In many situations, DOM is a neat solution. However, in some circumstances, the target XML document may be too big or even on the other side of the Internet. It may take too long a time for the local application to get the parsed result. This time-consuming process hence may jam the process, especially when what the application requires is just a small part of the target XML document. In these cases, SAX is a preferable solution.

2 – 2 - 5. XML Addressing

In many cases, while an application is processing XML resources, it needs to use identifiers to address the fragmentations within the XML resources with URI references. Such an addressing system provides a framework to identify the internal structures of XML resources.

XML Pointer Language (XPointer) is such a framework. It is offically recommended by W3C. XPointer provides a resource fragmentation addressing framework for some other W3C XML specifications including XML Linking Language (XLink), XML Inclusions (Xinclude), Resource Description Framework (RDF), and SOAP 1.2. XPointer is the official basis of resource identification for resources whose media types are text/xml, application/xml, text/xml-external-parsed-entity and application/xml-external-parsed-entity. Other XML-based media types on the Internet are also encouraged to adopt XPointer to define their fragment identifier *languages* [35].

XPointer is an extensive framework based on *XML Path Language* (XPath). XPath is a W3C recommendation, which concisely defines the addressing mechanic inside an XML document. It is also the basic fragment identification specification for XSLT. XPath is complimentary to other specifications and can not be used alone [36].

XML Linking Language (XLink) is another W3C recommended specification that "...allows elements to be inserted into XML documents in order to create and describe links between resources" [37]. The "links" here are explicit elements within an XML document representing XPointer addresses.

2 – 2 - 6. XML Query

An XML document is effectively a well-structured tree of elements. By using parsers, the logical structure and data contents of an XML document can easily be retrieved and used by applications.

In many circumstances, XML documents are used as database files or database record sets. The leaf element names of an XML document can be seen as the column names of a database table and the data contained in the leaf elements can be treated as values of the table column. Similar to DataBase Management System (DBMS) that has SQL as its query language, W3C drafts *XML Query Language* (XQuery) to be the query language for XML.

XQuery is expected to be a widely adopted technology while the XML documents are more and more widely used as data containers. Currently W3C has released its draft version 1.0 [38].

"A Java Specification Request 'XQuery API for Java (XQJ)' submitted by IBM and Oracle Corporation has been published through the Java Community Process (JCP). The specification design goal is to develop a common API that allows an application to submit queries conforming to the W3C XQuery 1.0 specification to an XML data source and to process the results of such queries. XQJ relates to XQuery in the same way that JDBC relates to SQL" [39].

2 – 3. XML Web Services Interaction Technologies

To invoke a Web service, a service requester must be able to locate a service provider, discover its communication mechanism, expected interaction pattern and message format, and negotiate with the target service provider on interaction features including security, reliability, coordination, transaction control, etc. These activities are currently defined as Web Services-specific on top of available XML and transport technologies.

Internet standards organizations have developed a number of Web Services specifications and defined a core technology stack. The core stack includes XML/XML schema, SOAP/XML-RPC, WSDL and UDDI. Most of today's Web Services products have claimed their support to this stack.

In Figure 1-3, level 3 and up are Web Services specific protocols. SOAP and XML-RPC are the two major protocols defining Web service interactions between service requester and provider. They both were developed before the Web Services framework came into reality and became the basic technologies of this framework. UDDI is a powerful registry and directory service system, which enables Web service provider entities to publish their services and Web service requesters to discover valid Web services. The Web service

interaction mechanisms and formats can be formally described in WSDL.

However, Web Services is still a new framework with many unspecified and even unrecognized feature considerations. Various Web Services vendors are developing specifications for some of these feature considerations. A few of these specifications have been submitted to the standards organizations and further developed to become industry standards. Meanwhile, many other concerns of the Web Services framework are emerging into the industry's vision and stimulate vendors to develop new specifications. In order to provide a complete Web Services solution, major Web Services products are generally implemented with some vendor-specific specifications in addition to the widely accepted industry standards.

2 – 3 - 1. XML-based Remote Procedure Call (XML-RPC)

XML-RPC, which stands for XML-based Remote Procedure Call, is designed by Dave Winer. It incorporates the idea of Remote Procedure Call (RPC), which is also designed by Dave Winer himself. RPC is a framework that allows one program to call another over a network with the help of a registry server. XML-RPC can be seen a Web implementation of the RPC protocol. It was inspired by both RPC and, more importantly, the old draft of SOAP. In some opinions, current SOAP standard was inspired by XML-RPC [40].

XML-RPC represents a client-server service model on top of HTTP. To start a communication, the client program sends a request message in XML format to the server as a HTTP POST request. It invokes a specified service program. The invoked service program then processes the parameters included in the message and sends a result back to the client. The procedure is synchronous, which means the client program keeps active

while waiting for the response from the service program. When XML-RPC is employed by Web Services, a Web service requester and a Web service provider respectively represent the client program and the service program.

The XML-RPC specification consists of three parts: Data Model, Request Structures and Response Structures.

Data Model defines the data structures that can be used in XML-RPC messages. The basic types are <int> (or <i4>), <Boolean>, <string>, <double>, <dateTime.iso8601>, <base64> (base64-encoded binary number), <struct> and <array>. The default type is string if not specified.

An XML-RPC request contains two parts: a header and a payload. Below is an example of XML-RPC request that contains all the required parts:

```
POST /RPC2 HTTP/1.0
User-Agent: Frontier/5.1.2 (WinNT)
Host: betty.userland.com
Content-Type: text/xml
Content-length: 181

<?xml version="1.0"?>
<methodCall>
    <methodName>examples.getStateName</methodName>
    <params>
        <param>
            <value><i4>41</i4></value>
```

```
        </param>
    </params>
</methodCall>
```

The first five lines construct the header. All values of these parameters have to be exactly specified, except the first line that is omitted when the server is only handling XML-RPC calls.

The XML-RPC response also contains two parts: a header and a body. Below is a response example corresponding to the above request message example. The first six lines construct the header of the response.

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 158
Content-Type: text/xml
Date: Fri, 17 Jul 1998 19:55:08 GMT
Server: UserLand Frontier/5.1.2-WinNT

<?xml version="1.0"?>
<methodResponse>
    <params>
        <param>
            <value><string>South Dakota</string></value>
        </param>
    </params>
</methodResponse>
```

The latest version of XML-RPC specification is written by Dave Winer [41].

2 – 3 - 2. Simple Object Access Protocol (SOAP)

Simple Object Access Protocol (SOAP) is both the trigger and the core technology of Web Services. It supports a much wider range of message exchange patterns for Web Services than XML-RPC. Its emergence is an important event that brings the Web Services idea into reality.

SOAP can be seen as an extension of XML-RPC. It provides a simple and extensible message exchanging system based on XML, which enables Web applications to define business-specific message content, structures and process logics by strictly following its essential non-business-specific rules. SOAP does not define any specific application semantics but rather defines a modular packaging system and the corresponding encoding mechanism.

On June 24, 2003, the latest SOAP specification version, Version 1.2, was released as a W3C official recommendation. It consists of three parts: *SOAP Version 1.2 Part 0: Primer* [42], *SOAP Version 1.2 Part 1: Messaging Framework* [43], and *SOAP Version 1.2 Part 2: Adjuncts* [44]. The discussion in this section is based on this recommendation.

The Primer part of SOAP v1.2 is a non-normative document that provides an easy-to-understand tutorial of the features in this SOAP specification. The Messaging Framework part defines the core of SOAP, including the processing model, extensibility model, binding framework and message construct. The Adjunct part defines a set of adjuncts that can be used in connection with the SOAP messaging framework.

Unlike XML-RPC, SOAP defines a peer-to-peer stateless communication model in a decentralized distributed environment. It is fundamentally a stateless one-way message exchange paradigm. By combining such one-way exchanges, applications can create more complicated interaction patterns such as request/response (as XML-RPC does), request/multi-responses, etc., with features provided by underlying transport protocol and/or application-specific information.

SOAP interaction patterns can be categorized into two models: Remote Procedure Calls and Conversational Message Exchanges. The former model represents the RPC-like request/response interaction pattern defined in SOAP v1.1. The latter model provides flexible support to a wide arrange of interaction patterns for Web applications (Web service agents).

Considering the increasing demand of implementing B2B transactions that are combined with a series of sub-transactions processed by multiple parties, SOAP v1.2 brings in a mechanism that can route messages. In such a mechanism, each party involved in a message processing is called a *SOAP node*. The set of SOAP nodes, through which a single SOAP message passes, is called a *message path*. It includes the initial SOAP sender, zero or more SOAP intermediaries, and an ultimate SOAP receiver. By indicating a message path and its processing logic, a Web service can use SOAP to form a line of sub-transactions to accomplish a certain business process. The template that establishes a pattern for the message exchanges between SOAP nodes is called a Message Exchange Pattern (MEP).

Enabling a Web Services communication protocol to work with lower level transportation protocols, e.g. HTTP, is called *binding*. While XML-RPC is bound to HTTP only, SOAP is not restricted as to what transport protocols it can bind to. Though W3C's SOAP1.2

recommendation only gives the default specification of binding with HTTP [44], it may also be bound to SMTP, BEEP, FTP, Transport Control Protocol (TCP) or other available transportation protocols.

A SOAP message is an XML document following the W3C recommendation - XML Information Set (XML Infoset). XML Infoset is "an abstract data set", which provides "...a consistent set of definitions for use in other specifications that need to refer to the information in a well-formed XML document" [45]. An important feature of Infoset is that it requires each name to be declared within a qualified namespace. A SOAP message also must not contain Document Type Declaration (DTD) information items or Processing Instructions (PI) information items.

A SOAP message is constructed with three parts: Envelope, Header and Body.

Envelope is the root element of a SOAP XML message. It contains a Body and an optional Header as its child elements, along with a number of attributes including a name value **Envelope**, a namespace declaration **http://www.w3.org/2003/05/soap-envelope**, etc.

Header is an optional child element in a SOAP message. It contains all the extensible features of a SOAP message as its subelements or attributes. A Header can contain a number of blocks describing various features or data. Three attributes are critical while declaring Header blocks in a SOAP message: *role, mustUnderstand* and *relay*. In some circumstances, a message may need to pass through multiple nodes. The *role* attribute indicates to which SOAP node in the message path a Header block is targeted. The *mustUnderstand* attribute indicates whether a process on a certain Header block is mandatory or optional for the targeted node. The *relay* attribute indicate whether a Header

block targeting a node should be relayed if not processed. These three attributes form a realistic pattern of serialized SOAP transaction.

A SOAP message Body may consist of a various numbers of elements and attributes representing the information being sent to the ultimate receipient.

Figure 2-5 shows an example of conversational SOAP message.

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
                env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
                 env:mustUnderstand="true">
   <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
   <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
  </m:reservation>
  <n:passenger xmlns:n="http://mycompany.example.com/employees"
                env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
                 env:mustUnderstand="true">
   <n:name>Åke Jógvan Øyvind</n:name>
  </n:passenger>
 </env:Header>
 <env:Body>
  <p:itinerary
    xmlns:p="http://travelcompany.example.org/reservation/travel">
   <p:departure>
```

```
      <p:departing>New York</p:departing>
      <p:arriving>Los Angeles</p:arriving>
      <p:departureDate>2001-12-14</p:departureDate>
      <p:departureTime>late afternoon</p:departureTime>
      <p:seatPreference>aisle</p:seatPreference>
    </p:departure>
    <p:return>
      <p:departing>Los Angeles</p:departing>
      <p:arriving>New York</p:arriving>
      <p:departureDate>2001-12-20</p:departureDate>
      <p:departureTime>mid-morning</p:departureTime>
      <p:seatPreference/>
    </p:return>
  </p:itinerary>
  <q:lodging
   xmlns:q="http://travelcompany.example.org/reservation/hotels">
   <q:preference>none</q:preference>
  </q:lodging>
 </env:Body>
<env:Envelope>
```

Figure 2-5 an example of conversational SOAP message

(Source: SOAP Version 1.2 Part 0: Primer [42])

There are two optional standard namespaces that can be adopted by SOAP: http://schemas.xmlsoap.org/soap/envelope/ for Envelope and http://schemas.xmlsoap.org/soap/encoding/ for encoding. They can be substituted by

other namespaces if necessary.

Vendors of the Electronic Business industry have already released various development products supporting the SOAP v1.1 API in programming languages including Java, COM, Perl, C#, Python, etc. Binding specifications with transport protocols other than HTTP are either already implemented in a product (e.g. SMTP in .NET) or under development (e.g. SOAP over BEEP by IETF).

Without an institutional implementation guide, the message exchanges between different vendor platforms or APIs may encounter difficulty to interoperate even if they have employed the same SOAP v1.1 standard. This implementation incompatibility appeared among early Web Services platforms and still exists among some products today. It is one of the reasons that forced the creation of Web Services Interoperability Organization (WS-I) and its development of basic profiles to guide Web Services standard implementations including those for SOAP v1.1.

W3C released the official recommendation of SOAP v1.2 on 24 June 2003.

2 – 3 - 3. Web Services Description Language (WSDL)

Web Services Description Language (WSDL) provides a model to describe the mechanism of using a Web service with machine-processable XML documents. Microsoft, IBM and Ariba released the first version of WSDL in September 2000 and, along with some other companies, submitted WSDL Version 1.1 to W3C in March 2001. A W3C Working Group is currently developing WSDL Version 1.2. The discussion in this section is based on the latest WSDL v1.2 working draft of June 11, 2003.

WSDL Version 1.2 consists of three parts: *Part 1 - Core Language*, *Part 2 – Message Patterns* and *Part 3 – Bindings*. Part 1 defines a language to describe abstract Web service functionalities and a framework for describing the concrete Web service description details [46]. It defines the framework and the basic concepts of WSDL v1.2. *Part 2 – Message Patterns* defines the sequence, direction and cardinality of abstract messages sent and received by a Web service operation [47]. The supported interaction patterns by WSDL v1.2 are *In-Only*, *In-Out*, *Request-Response*, *In-Multi-Out*, *Out-Only*, *Out-In*, *Out-Multi-In* and *Multicast-Solicit-Response*. *Part 3 – Bindings* defines Web service binding extensions and corresponding message formats for SOAP Version 1.2, HTTP Version 1.1 GET/POST and MIME (IETF RFC2045) [48].

Figure 2-6 illustrates the understanding of Web services in the vision of a service describer. WSDL describes Web services starting from the *message* exchanged between service provider and requester. An exchange of the message between a service requester and a service provider is called an *operation*. A collection of operations forms an *interface*. An interface is bound to one or more transport protocols and corresponding message formats via *binding*. Each binding, and therefore the interface it binds to, is accessed by a number of *endpoints*, each of which is indicated by a URI. A Web *service* hence refers to a collection of endpoints bound to the same interface.

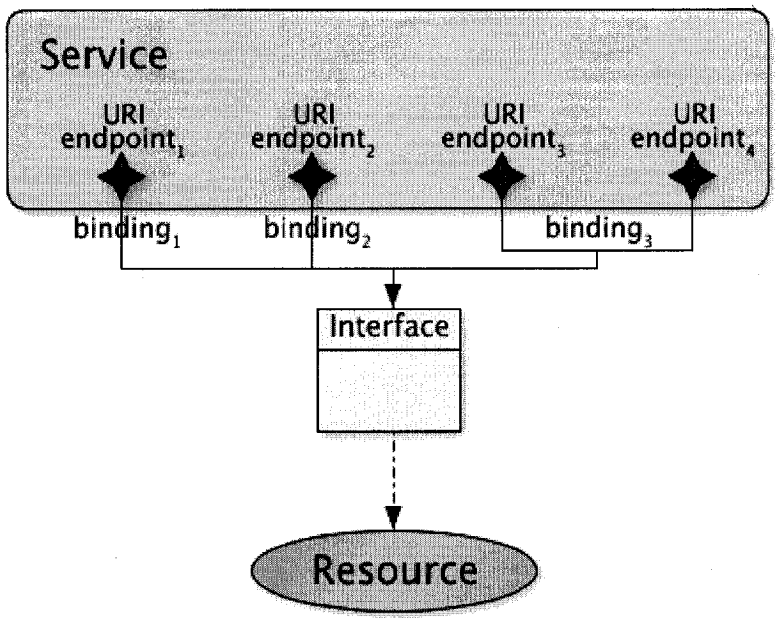

Figure 2-6 Understanding of Web services as a service describer

(Source: *WSDL v1.2 Part 1: Core - W3C Working Draft 11 June 2003* [46])

A WSDL document is an XML Infoset document with the extension name .xdsl. Its root element is <definitions>, which is actually a container of two categories of components: WSDL components and data type system components. The WSDL components are messages, interfaces, bindings and services. Data type system components are element declarations and data type definitions from other data type systems. Each component is an *Element information item* specifying certain properties.

The <definitions> component of a WSDL document defines the following properties: <messages>, <interfaces>, <bindings>, <services>, <type definitions> and <element declarations>. The structure of a WSDL document is defined as follows:

- A local name "<definitions>"

- A namespace name "http://www.w3.org/2003/06/wsdl"
- One or more *attribute information items* amongst its <attributes> as follows:
    - A *targetNamespace* attribute information item that affiliate to the top level components, including message, interface, bindings and service.
    - Zero or more namespace qualified *attribute* information items. The namespace names declared here MUST NOT be "http://www.w3.org/2003/06/wsdl".
- Zero or more *element* information items amongst its [children] in the order:
    - An optional *documentation* element information item, container of human-readable text.
    - Zero or more *element* information items from among the following in any order:
        - Zero or more *include* element information items, which allows for inclusion of definitions from relatively independent documents in the same namespace.
        - Zero or more *import* element information items, which allow for the import of components from documents in another namespace.
    - An optional *types* element information item, which contains imported and embedded schema compoents.
    - Zero or more *element* information items from among the following, in any order:
        - *message* element information items
        - *interface* element information items (includes operation definition with message pattern)
        - *binding* element information items
        - *service* element information items
    - Zero or more namespace qualified *element* information items amongst its [children]. Such items MUST be a member of one of the *element substitution groups* allowed at the top-level of a WSDL document.

Human developers or Web service requesters can read and analyze WSDL documents to

understand the interaction mechanisms and required message formats to interact with the described Web services. Certain feature descriptions, such as Web Services choreography, security and business processes, can be added on top of the basic WSDL descriptions to define more sophisticated tasks.

WSDL is currently the major method to fully describe Web service mechanisms. It is expected to become a more powerful description tool with a more extensible frame along with Web Services' evolving.

2 – 3 - 4. UDDI

In order to exploit the functionalities provided by Web services, users must be able to discover sufficient information to locate and execute the Web services. *Universal Description, Directory & Integration* (UDDI) is such a mechanism that is universally accepted.

UDDI is developed by UDDI.org, a member section of the *Organization of the Advancement of Structured Information Standards* (OASIS). It provides a set of descriptions and discovery services for: (1) businesses, organizations and Web service providers; (2) available Web services provided by these institutions; and (3) technical interfaces to make use of these Web services. UDDI provides interfaces for both publishing the information and discovering the published information. It can be employed both publicly on the Web and privately within an organization.

UDDI is a scheme on top of a number of Web standards including HTTP, XML, XML schema and SOAP. Because most of today's Web services are available only for internal usage, UDDI is the most sparsely adopted technology among the four Web Services core

stack technologies: XML, SOAP, WSDL and UDDI. .

UDDI has so far been developed in three versions. UDDI version 2 was ratified an official OASIS standard in May 2003. It was criticized for lack of sufficient security consideration. UDDI.org therefore released UDDI version 3, which addresses security concerns that were criticized missing in version 2 and adds an advanced access control. This latest version of UDDI also enables robust query against rich metadata, which envisions UDDI as a "meta service" for locating Web services.

UDDI v3.0 is widely expected to be a complete powerful solution for Web services description and discovery. The discussion in this section is based on UDDI version 3.

In UDDI, a structure of data representing a certain object is called an *entity*. Each UDDI information model is composed of a number of instances of six entity types:

businessEntity: Describes a business or organization that provides Web services;
businessService: Describes a collection of Web services provided by a businessEntity instance;
bindingTemplate: Describes the essential technical information to use a Web service;
tModel: Stands for *Technical Model*, which represents a reusable concept, such as a type of Web services, a category system, or a protocol used by Web services;
publisherAssertion: Describes a businessEntity's relationship with another one;
subscription: States a standing request to keep track of the changes of certain entities.

The businessEntity, businessService, bindingTemplate and tModel form the *core data structure* of UDDI. Figure 2-7 illustrates the core data types and structure.

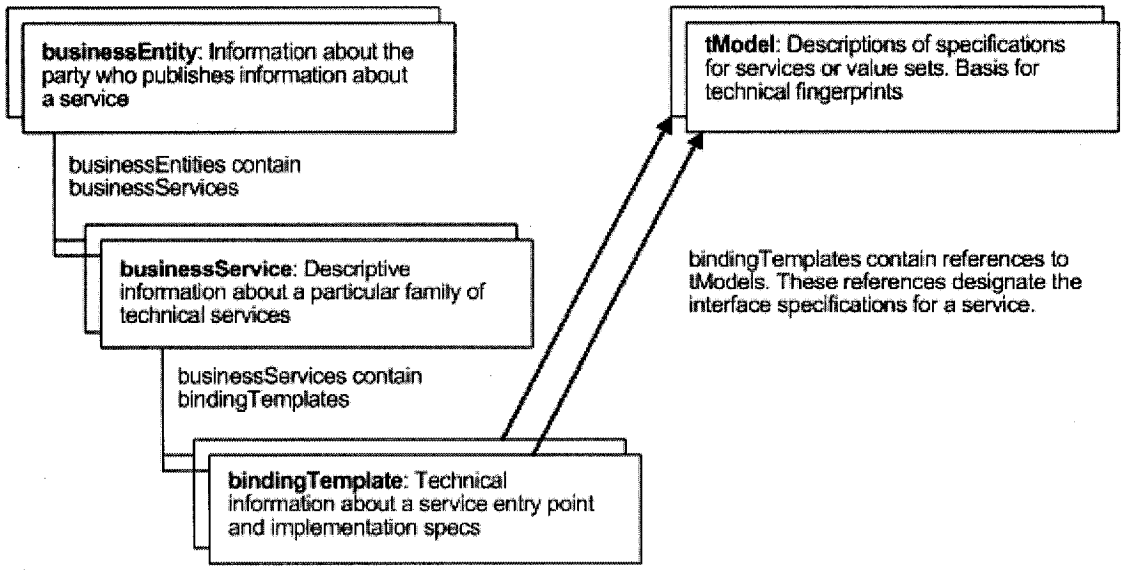

Figure 2-7 UDDI core data types and structure

(Source: *UDDI V3.0, UDDI Technical Committee Specification, 19 July 2003* [49])

UDDI data structures are formally defined with XML schema version 1.1.

UDDI defines two types of API sets to define and manipulate the entities: Node API sets and Client API sets. The Node API sets include UDDI inquiry, UDDI publication, UDDI security, UDDI custody transfer, UDDI subscription and UDDI replication. The Client API sets include UDDI subscription listener and UDDI value set. A set of Web services supporting at least one Node API set is referred to as a *UDDI node*. A combination of one or more UDDI nodes forms a UDDI *registry*. The businessEntity, businessService, bindingTemplate and tModel form the *core data structures* of UDDI. Within a UDDI registry, each core data structure instance holds a unique *UDDI key*. By following an appropriate policy, a number of UDDI registries can form a *UDDI affiliate* to share controlled copies of core data structure instances.

In order to use UDDI services, UDDI defines a *UDDI programmer API* and a *Replication API set*. Each UDDI registry must have at least one node that offers a Web service compliant with Inquiry API set. Normally, a UDDI registry should also have at least one node that provides a Web service implementing Publication, Security, and Custody and Ownership Transfer API sets. The Replication API should also be provided as Web services if a registry has more than one UDDI node. Generally, these Web services that implement UDDI APIs employ SOAP to define their interaction patterns. WSDL descriptions for a Web service are easily to be broken and converted into businessService, bindingTemplate and tModel entities in UDDI under the service provider entity's businessEntity.

Figure 2-8 illustrates a typical usage of UDDI registry for locating and invoking Web services.

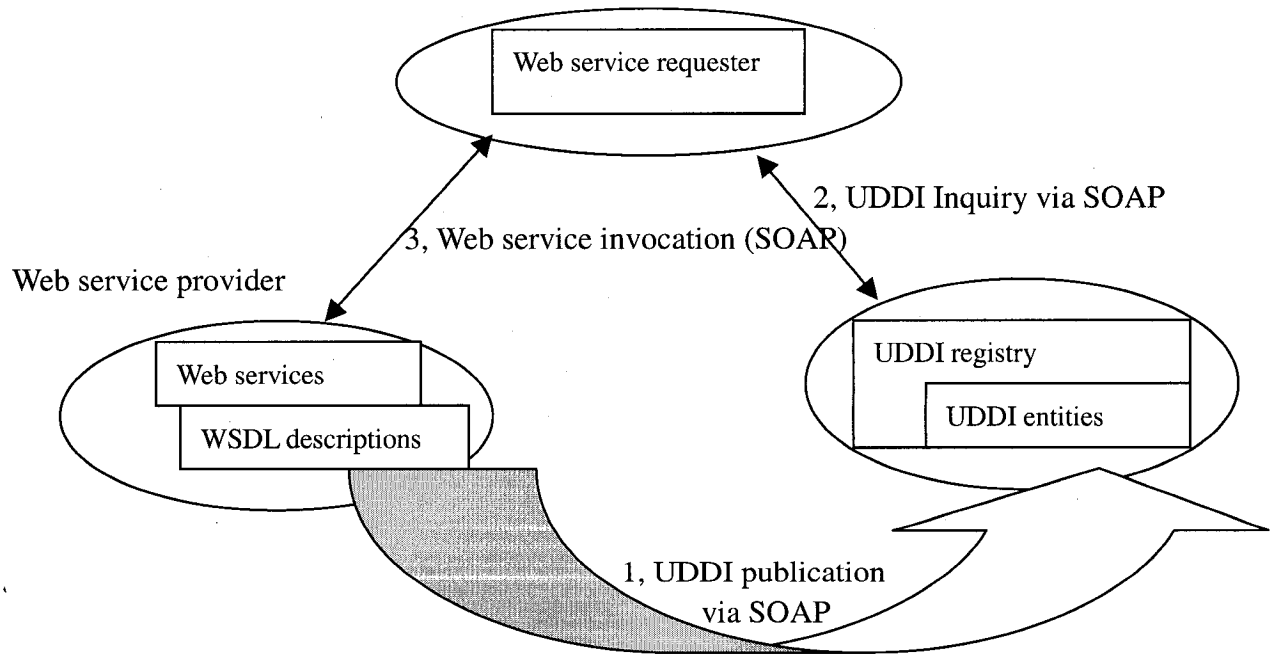

Figure 2-8 Web service using UDDI Web service APIs

2 - 4. Web Services Business Specifications

SOAP, WSDL, UDDI, XML, HTTP and their alternative technologies compose the fundamental Web Services technology stack. This stack enables distributed applications to freely exchange data and collaborate on accomplishing various tasks. Web Services brings many valuable benefits to business application development, one of which is enabling businesses to conduct B2B processes in an automated, highly efficient and low-cost manner.

In a B2B process, at least two parties are involved: a customer and a supplier. Sometimes even multiple suppliers might be involved. Each of these parties is represented by a set of applications that share an identical interface, which conducts trading activities in place of humans. Generally, these applications can discover, locate and interact with each other to collaborate on conducting business tasks on top of some message exchanging mechanisms such as the Web Services core technology stack.

While Web services are able to provide a highly interoperable mechanism between business systems, business applications must also be able to interact with each other via common business languages and logic. Since each industry has its own specific business terms and process definitions, it is natural to employ a specific business language to implement a specific type of transactions.

A business language for a specific industry normally needs to precisely define all the required terms, communication specifications and business patterns of this industry to implement business transactions in an extensible manner.

Many different business language specifications have been developed by various business-focused standards organizations. They are either for describing business processes in different industries or representing the activities in the same industry from different approaches. Most of these business languages are developed based on XML with XML Schema, which makes them easily be applicable to the Web Services framework with minimum modification.

A few of these business languages are more widely adopted by their target industries than others. These popular languages are mostly developed by business specific organizations including RosettaNet, HL7, XBRL, etc.

The Internet standards organizations and key Web Services vendors are collaborating to further improve the existing business languages and standardize the common areas between different business languages. IBM and Microsoft, along with a number of other vendors, have also developed a general business language standard *Business Process Execution Language for Web Services* (BPEL4WS).

2 – 4 - 1. Business Process Execution Language For Web Services (BPEL4WS)

BPEL4WS is a business language specification that can be used to describe general E-business process patterns with multiple Web services and standardizes the internal and between-partners message exchanges. It derives from two private business process language specifications: Web Services Flow Language (WSFL) and eXtensible LANGuage (XLANG.), which respectively belong to IBM and Microsoft.

IBM, Microsoft and a few other E-business giants first submitted BPEL4WS version 1.0 to OASIS on 31 July 2002 and then version 1.1 in May 2003. OASIS initiated a Web

Services Business Process Execution Language Technical Committee (BPEL TC) to further improve the BPEL4WS specification.

BPEL4WS collectively defines a mechanism to reliably and consistently define, create, and interact multiple business processes in a Web Services environment. It adopts the ideas from WS-coordination and WS-transaction, references various specifications for describing Web services coordination and collects the transactions belonging to IBM, Microsoft and BEA. [55]

In BPEL4WS, a *business process* is a complete business activity that might involve a sequence of inter-business interactions among multiple business partners to accomplish a certain task, which should have a start operation and an end operation.

Normally, an E-business process is described in two models: *executable* business process model and *business protocol* business process model. The *executable* model describes process behavior of each business participant, whereas the *business protocol* model focuses on message exchange behaviors between business participants. A process description in the *business protocol* model is called an *abstract process*. BPEL4WS specification focuses on the common core concepts for both executable and abstract processes. Meanwhile, it also provides modest extensions for both of the two patterns.

The discussion in this section is based on BPEL4WS v1.1. It is layered on top of a number of XML specifications: WSDL v1.1, XML schema v1.0 and XPath v1.0, among which WSDL v1.1 holds the most influence on BPEL4WS.

A business process normally consists of multiple interactions among a number of partners. Within a Web Services framework, each business interaction is implemented as a Web

service interaction, which is described in WSDL. BPEL4WS defines the whole business process by specifying peer-to-peer Web service interactions and their orchestration. While referring to a Web service, BPEL4WS uses its abstract description – portType (*interface* in WSDL), but not any deployment-specific description (e.g. binding). This approach keeps the BPEL4WS business process descriptions reusable and durable regardless of the specific Web service deployments being used.

In a business process, each participant business entity is called a business *partner*. A business process normally involves multiple partners. Partners are connected in a bilateral manner called *partner link type*, which specify two sets of all the Web services provided by both of the connected partners. Each partner can use BPEL4WS to define each of these Web services as a *role*, which indicates the role of this partner in the interaction while using this Web service. For each business process, a subset of roles can be selected to form a *partner link*. Within a partner link, the local partner's role is defined as the value of *myRole*, and the partner's role is specified as the value of *partnerRole*.

Implemented with Web services, the business process or part of the business process may be either *synchronous* or *asynchronous*.

Each business process is executed within a specified *business context*. A business context is composed by a collection of *container*s, each of which consists of critical data for correctly performing a business process. Normally business contexts are persistently stored, in order to preserve the consistency and reliability of business processes. For example, in case of system outrages, those running business processes can be appropriately resumed by recovering their business contexts from persistent storage.

A partner system might be involved in more than one business processes at the same time,

especially when it is participating asynchronous processes. In these situations, each message handled by the system needs to be recognized according to the business process it belongs to. This recognition information is referred to as *correlation*. In BPEL4WS, the data representing correlations is referred as *property*.

The major function of BPEL4WS is to describe business processes. A process can be defined as a structural composition of *activities*. BPEL4WS v1.1 specifies 15 types of activities that are categorized into two groups: basic activities and structure activities.

The basic activities include:

<receive>, waiting for an invocation request from a specific partner and link

<pick>, waiting for a message from any partner to continue the process

<reply>, replies a request, generally in an asynchronous model

<invoke>, invokes another operation in an asynchronous model

<assign>, copies container content to another container

<terminate>, indicates the business process should be immediately terminated

<throw>, signals an error occurrence

<wait>, causes the business process to wait for a specified period

<empty>, suggests doing nothing

<compensate>, undoes what has been accomplished within the business process.

The structure activities can be used to define structural business process logic. It is like using <if>...<then> to define structural blocks in programming. These structure activities include:

<flow>, indicates the activities within it can be executed concurrently

<sequence>, defines its sub activities to be executed in sequence

<switch> and <while>, define the structure of the processes like "switch" and "while" in

popular programming languages

<scope>, defines a group of activities sharing the same properties.

BPEL4WS also provides a description section to handle process faults. This section is defined in <faultHandler> block, which may include some BPEL4WS activities. A <faultHandler> block requests business process partners to provide "undo" Web services in order to recover from a fault while executing a business process. BPEL4WS also provides a flexible and extensible scheme to define fault-dealing procedures.

BPEL4WS is a language based on XML. It provides a flexible and extensible document format that fully meets XML requirements.

Figure 2-9 shows a typical BPEL4WS document structure.

```
<process name="ncname" targetNamespace="uri"
    queryLanguage="anyURI"?
    expressionLanguage="anyURI"?
    suppressJoinFailure="yes|no"?
    enableInstanceCompensation="yes|no"?
    abstractProcess="yes|no"?
    xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/">
    <partners>?
    <!-- Note: At least one role must be specified. -->
        <partner name="ncname" serviceLinkType="qname"
            myRole="ncname"? partnerRole="ncname"?>+
        </partner>
    </partners>
```

```
<variables>?
<!-- Note: The message type may be indicated with the messageType
    attribute or with an inlined <wsdl:message> element within. -->
    <variable name="ncname" messageType="qname"?>+
        <wsdl:message name="ncname">?
        ...
        </wsdl:message>
    </variable>
 </variables>
 <correlationSets>?
        <correlationSet name="ncname" properties="qname-list"/>+
 </correlationSets>
 <faultHandlers>?
 <!-- Note: There must be at least one fault handler or default. -->
        <catch faultName="qname"? faultVariable="ncname"?>*
            activity
        </catch>
        <catchAll>?
            activity
        </catchAll>
    </faultHandlers>
    <compensationHandler>?
        activity
    </compensationHandler>
    <eventHandlers>?
    <!-- Note: There must be at least one onMessage or onAlarm handler. -->
        <onMessage partner="ncname" portType="qname"
```

```
        operation="ncname" variable="ncname">
        <correlations>?
            <correlation set="ncname" initiate="yes|no"?>+
        <correlations>
        activity
    </onMessage>
    <onAlarm for="duration-expr"? until="deadline-expr"?>*
        activity
    </onAlarm>
</eventHandlers>
activity
</process>
```

Figure 2-9 BPEL4WS document structure standard

(Source: *Business Process Execution Language for Web Services Version 1.1* [56])

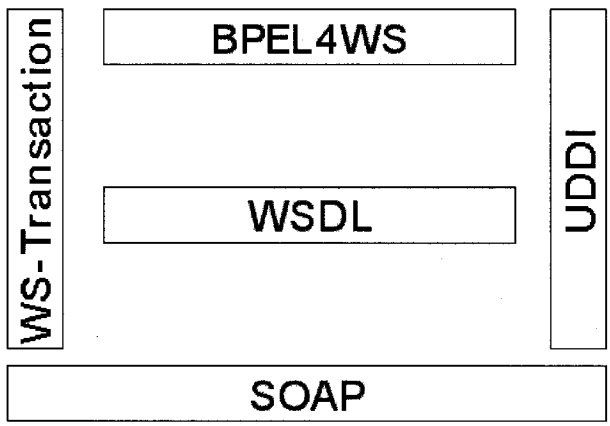

Figure 2-10 Web Services specification family with BPEL4WS

(Source: *Business Process Execution Language for Web Services Version 1.1* [56])

The position of BPEL4WS in current Web Services framework is shown in Figure 2-10.

2 – 4 - 2. RosettaNet Standards

RosettaNet is a self-funded global consortium of Electronic Components (EC), Information Technology (IT) and Semiconductor Manufacturing (SM) companies. It works to create, implement and promote E-business process standards for these three industries. The standards it works on are focused in the global B2B supply chain interactions in these three industries.

RosettaNet is composed of five boards: the EC Supply Chain Board, IT Supply Chain Board and SM Supply Chain Board that lead the standard development in each industry, Solution Provider Board that drives critical development and implementation strategies to support RosettaNet's key initiatives, and Executive Board that supervises the organizational direction and issues between different supply chains.

RosettaNet is one of the most influential E-business standard organizations in the world. It currently operates with the collaboration of more than 400 companies in EC, IT and SM industries representing more than US$1 trillion in total.

RosettaNet was formed in 1998. It was named after the Rosetta stone, the ancient tablet discovered in Egypt, which carries inscriptions of the same message in Greek and two ancient Egyptian languages. Scholars translated the two unknown Egyptian languages by using the Greek inscription.

RosettaNet is committed to driving "...collaborative development and rapid deployment of Internet-based business standards, creating a common language and open e-business

processes that provide measurable benefits and are vital to the evolution of the global, high-technology trading network" [58].

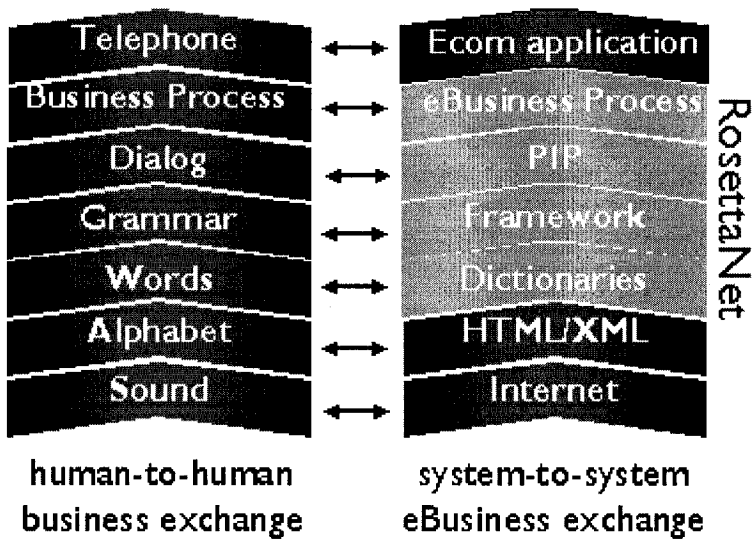

Figure 2-11 RosettaNet standard illustration

(Source: http://www.rosettanet.org)

The RosettaNet's standard structure is illustrated in Figure 2-11. This XML-based structure provides a framework of cross-platform, -application and -network interaction standards.

The RosettaNet standards consist of three parts: RosettaNet Dictionaries, RosettaNet Implementation Framework (RNIF) and Partner Interface Processes (PIP).

RosettaNet Dictionaries define a common semantic platform for conducting businesses within the supply chain. This semantic platform standardizes the terminologies for conducting businesses and eliminates process overlaps among E-business trading partners. It consists of a Business Dictionary and a Technical Dictionary. The RosettaNet Business

Dictionary defines transaction properties for business activities, whereas the Technical Dictionary defines technical properties for products and services.

RNIF Core Specifications define the XML-based exchange protocols for RosettaNet standards. These protocols define message transport, routing and packaging, security, signals, and trading partner agreements.

RosettaNet PIPs define business processes between trading partners. It is the core component of the RosettaNet standards family. These PIPs are divided into eight clusters. Each cluster is a collection of categorized core business processes that compose the business network backbone.

These clusters are indicated by numbers from 0 to 7:

0: RosettaNet Support, provides administrative functionalities

1: Partner Product and Service Review, allows various manipulations on trading-partner profiles and product-information

2: Product Information, enables distribution and periodic update of detailed product design information

3: Order Management, completely covers the order management business procedure from price quoting to payment and discrepancy notification

4: Inventory Management, supports all inventory related processes such as replenishment, allocation, etc.

5: Marketing Information Management, defines communication of marketing information including campaign plans, lead information and design registration

6: Service and Support, provides post-sales support, warranty and asset managements

7: Manufacturing, covers communication of design, configuration, and other information that support "Virtual Manufacturing" environment.

Each of these clusters is divided into a number of segmentations. Each of these segmentations defines a set of cross-enterprise business processes that are collaborated by multiple types of partners. PIPs are specifications of the business processes defined within these segmentations.

By June 23, 2003, totally 107 PIPs have been published in RosettaNet's website www.rosettanet.org/ [57] while more PIPs are under development. Each PIP has a unique identifier of a three-character string. In this string identifier, the first number indicates the cluster it belongs to, the second letter indicates its segmentation and the third number indicates the number of this PIP within the segmentation. For example, PIP 3A4 indicates the 4^{th} PIP within segmentation A of the 3^{rd} cluster.

Each PIP comes with a detailed PIP specification and a PIP Implementation Guide (RIG). A RIG consists of two parts: an Implementation Guide and a Mapping Tool. The Implementation Guide describes the business scenario, usage notes and lessons learned of the PIP, whereas the Mapping Tool assists it with information types and formats for future business partners.

RosettaNet PIP is one of the most widely employed business standards as well as a very good choice for implementing E-business with Web Services.

2 – 4 - 3. ebXML

The term ebXML stands for *Electronic Business using extensible Markup Language*. It "...is a modular suite of specifications that enables enterprises of any sizes and in any geographical location to conduct business over the Internet" [60]. OASIS and *United Nations Centre for Trade Facilitation and Electronic Business* (UN/CEFACT) jointly

sponsored the development of ebXML. UN/CEFACT, which has developed the widely adopted Electronic Data Interchange (EDI) standard, is a subsection of the United Nations.

The mission of ebXML is "To provide an open XML-based infrastructure enabling the global use of electronic business information in an interoperable, secure and consistent manner by all parties" [61]. Furthermore, the goal of ebXML is "...creating a single global electronic market" [65].

ebXML is continuously under development. This technical suite adopted experience and strengths from the existing Electronic Commerce/Electronic Data Interchange (EC/EDI) technology and use XML for its data presentation. Currently, it contains specifications for exchanging business messages, conducting trading relationships, communicating data in common terms and registering business processes.

SOAP, WSDL and UDDI form an approach towards Web Services from a different direction than ebXML. The former approach was initially to establish an RPC mechanism to enable free and efficient message exchanges based on XML. The latter approach was initially to improve the business document exchange based on the idea of the existing EDI technology and borrowed part of the EDI terminology. However, after a period of development, these two approaches are merging towards providing compatible solutions for business-to-business document exchange across the Web.

The latest ebXML suite was recently endorsed jointly by UN/CEFACT and OASIS on June 3, 2003 at Geneva, Switzerland. This latest suite is generally referred to as ebXML 2.0. It consists of seven components: ebXML Message Service Specification v2.0, ebXML Registry Information Model v2.0, ebXML Registry Services Specification v2.0,

ebXML Collaboration Protocol Profile and Agreement v2.0, ebXML Business Process Specification Schema v1.01, ebXML Technical Architecture v1.04 and the ebXML Requirements v1.06. All these specifications can be found and downloaded from http://www.ebxml.org/specs/index.htm.

The scope of ebXML is the business side in both B2B and Business to Consumer (B2C). Application to Application (A2A) within an enterprise can also adopt ebXML to implement. But this implementation should not be developed in the expense of B2B or B2C.

Like the Web Services mechanism are composed of SOAP, WSDL, UDDI and BPEL4WS, ebXML defines the architecture and a set of specifications for implementing electronic trading systems with similar functions and feature considerations. The technologies used in the former Web Services mechanism are relatively independent so that they can also be used in other purposes. This attribute makes this mechanism more flexible than the latter one. The latter mechanism presents a one-piece total solution for E-business, which makes it more integrated and more consistent.

To avoid overlap or ambiguity, ebXML recommends the UN/CEFACT Modeling Method (UMM) as its modeling methodology to be described in Unified Modeling Language (UML).

UMM modeling system can be broken into two parts: Business Operational View (BOV) and Functional Service View (FSV).

In ebXML, any business that participates into a certain *business process* is called a *trading partner*. Within BOV, a trading partner applies its *business collaboration*

knowledge, uses the existing ebXML *core library* and *business library*, and follows certain analysis and design rules to define its own *Business Process and Information Models*. The core library contains *core components*, which are the commonly basic components containing data and process definitions that can be adopted by various *business process*es. The business library contains *business processes and business information*, which in turn contain common or standard business process and information definitions that can be reused. BOV provides a three-phase method to design an ebXML system and allows defining the system by specifying Business Processes and Information Models. These Business Processes and Information Models are Meta models compliant to ebXML and ready to be used by trading partners.

With complete Business Processes and Information Models, FSV standards specify the supporting services for ebXML.

FSV requires an ebXML registry service, which stores the XML conversion of Business Processes and Information Models, along with other repositories including the core library. ebXML can employ UDDI to implement this registry service. Before performing a business process, a trading partner needs to access the registry to retrieve Business Processes and Information Models, core library and *Collaborative Protocol Profile*s (CPPs) of other trading partners with whom it intends to do business. A CPP is a document that describes the *Business Processes* and *Business Service Interface*s a trading partner can provide. Each trading partner should have its CPP published in the registry server. Based on all registered CPPs, trading partners need to negotiate and form a commonly agreed on agreement to conduct their business interactions. This agreement is called *Collaborative Process Agreement* (CPA). By strictly following a CPA, a group of trading partners can then perform business processes with *Message Service*s (message exchange mechanics).

ebXML defines three functional phases to accomplish business processes.

The first phase is Implementation Phase. In this phase, a trading partner retrieves the Business Processes and Information Meta Models, core library, business library and other information from the registry service, implements its own business processes, submits and updates corresponding information including CPP to the registry service.

The second phase is Discovery and Retrieval phase. In this phase, a trading partner discovers and retrieves all the information it needs to implement business interactions with others. The information may include: updated information acquired from the Implementation Phase, the other trading partners' CPPs, a list of scenarios, messaging patterns and security constraints. All trading partners involved are required to participate in the negotiation to work out a choreographed CPA to conduct future E-businesses. Each CPA then needs to be assigned a universally unique identification, such as a URI. In current ebXML specifications, the negotiation of a CPA has to be manually performed. This negotiation process is expected to be automated in the future when appropriate technology is developed.

The third phase is Runtime Phase. In this phase, the trading partners can do the actual transactions via ebXML Messaging Service by following those negotiated CPAs. ebXML Messaging Service does not specify any transport protocol but is open to any appropriate protocol. **SOAP with attachment** [66] is an optional standard for ebXML Messaging Service. The Messaging Service model is shown in Figure 2-12.

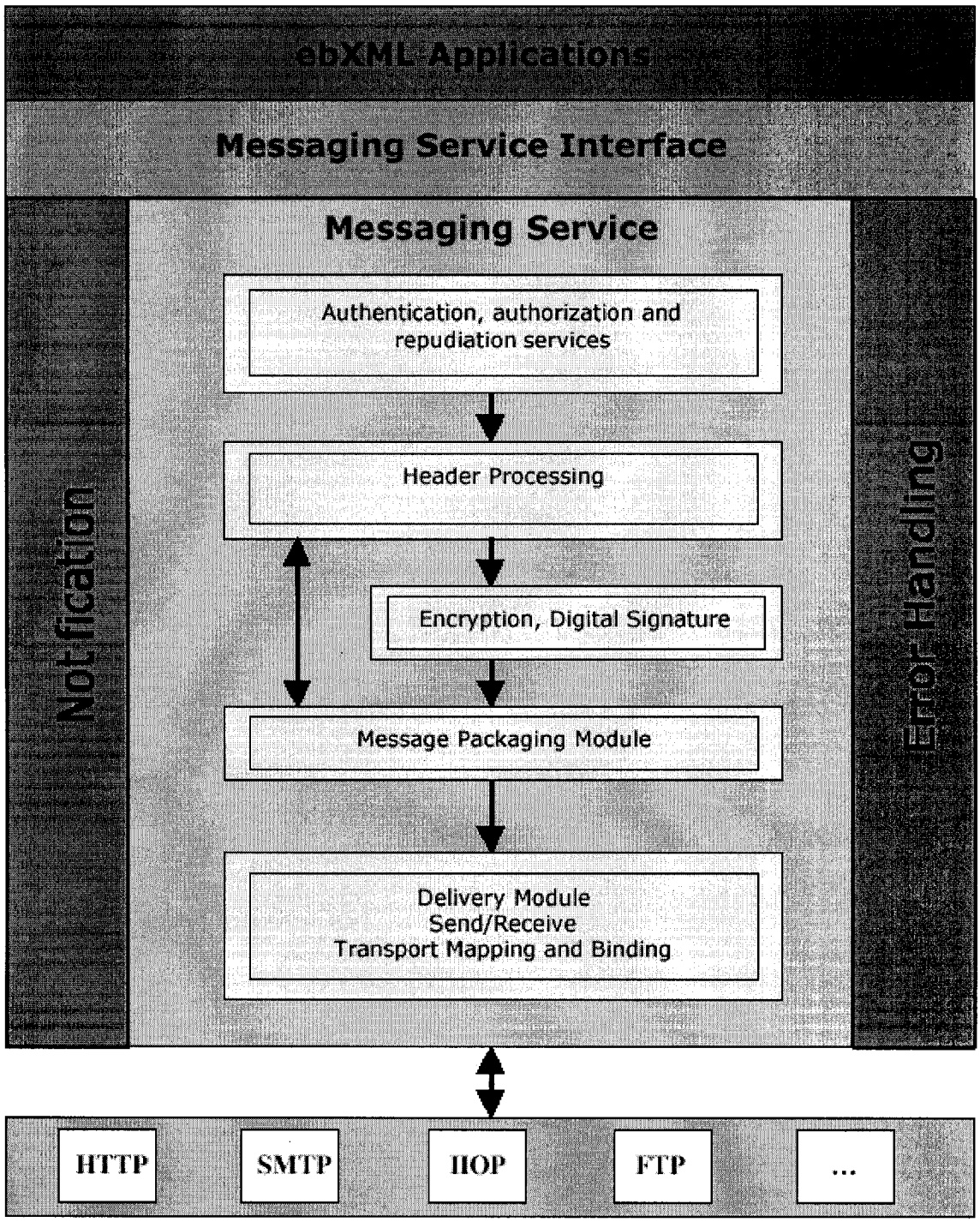

Figure 2-12 The Messaging Service Architecture

(Source: *ebXML Technical Architecture Specification v1.0.4* [63])

ebXML is still not a complete E-business standard suite yet. Apart from the seven endorsed specifications introduced earlier in this section, a few complementary technical reports are still under development. These reports include Core Component Dictionary, Catalog of Common Business Processes, etc. On the other hand, ebXML is open to adopt independent technologies to make its standard stack more functional. For example, SOAP can be adopted by ebXML for its Messaging Service and UDDI can be adopted by ebXML for its Registry Service.

Dieter Jenz from Jenz & Partner says, "At present, ebXML is the only non-proprietary horizontal business collaboration infrastructure architecture that provides integration at the business process level" [59].

CHAPTER 3

Feature Considerations For Web Services

Besides the Web Services core stack technologies described in Chapter 2, there are some other critical technical considerations before Web Services can be widely deployed. These considerations, similar to that of other interaction technologies, include Interoperability, Security, Reliability, Orchestration, Scalability, etc. The key Web Services vendors and standards organizations are collaborating on developing highly effective solutions for these features. Some progress has already been made.

3 - 1. Interoperability

Any technologies that involve interoperations between systems, including Web Services, need to consider their interoperability. Interoperability generally refers to the capability of conducting smooth interoperations between various deployments while employing the same technology, such as Web Services.

A Web service requester needs to be able to interoperate with the service providers and the Web Services registry server, exchange data and error messages in XML, acquire and

analyze Web service descriptions, negotiate and set up security protocols, choreograph collaboration activities, and conduct other activities. The interoperation may be within an enterprise, between business partners or across a heterogeneous set of applications, platforms and languages.

Currently, most Web Services solutions are developed for Enterprise Application Integrations (EAIs) within enterprise systems. Normally an enterprise will choose identical platforms, languages, Web Services technology stack and products to implement its Web Services EAI. This makes the implementation easy for Web services designers. A framework or platform would have no problem interoperating within itself.

Along with the evolution of Web Services technologies, products and EAIs, the demand to implement inter-business Web services is growing. This makes Web Services interoperability a critical issue: machine-to-machine interoperation requires that all participants completely understand each other's communication protocols, message structures, contents and semantics without any ambiguity.

In fact, however, participant business partners normally employ various types of Web Services platforms that may have some difficulty in interoperating with each other.

Many Web Services standards have already been released by standard organizations and vendors, which include W3C, OASIS, UN/CEFACT, IBM, Microsoft, etc. Sam Ruby, a key member of IBM Emerging Technologies Group, stated in an interview, "There's no question that (Web Service) standards have outpaced implementations. There are too many standards out there. There will still be more standards created." [67] For Web Services, the proliferation of standards may cause confusing while implementing products, hence discourage the evolvement or even adoption of this technology.

To achieve complete interoperability between Web services, some requirements must be fulfilled. First, the Web services must adopt the same protocol stack for implementation and deployment. Second, any of these standards must follow the same unambiguous and complete implementation guide, which clearly states mandatory and optional parts of the standards and how they should be implemented.

In order to work out an industry-wide solution for this critical issue, E-business vendors including IBM, Microsoft and BEA founded the Web Services Interoperability Organization (WS-I).

WS-I defines a set of *profiles, testing tools* and *use cases and usage scenarios*, and a number of *sample applications* to improve Web services implementation and interoperability. Each *profile* declares a set of optional specifications, which can be used to compose certain Web Services functionalities, and their correspondent implementation guidelines. The *testing tools* include a communication sniffer and an analyzer. These tools can be used to test whether an implementation precisely meets the requirements declared in the profiles. The *use cases and usage scenarios* respectively specify the indicated requirements that use Web services. The *sample applications* are implementations of the use cases and usage scenarios that can be used to help improving the testing tools. WS-I puts all its delivered documents on their website at http://www.ws-i.org/Documents.aspx.

Among these documents, *Basic Profile v1.0* [76] is currently a draft for public review. It specifies a Web Services standard stack including Messaging, Description and Service Publication and Discovery with optional Security concerns, and a standard implementation guide for these specifications.

The Messaging specifications include SOAP 1.1, XML 1.0 (Second Edition) with HTTP 1.1 and HTTP State Management Mechanism (RFC2965). The Description specifications include WSDL 1.1, XML Schema Part 1: Structures and XML Schema Part 2: Datatypes. The Service Publication and Discovery specifications include UDDI Version 2.04 API Specification - Dated 19 July 2002, UDDI Version 2.03 Data Structure Reference – Dated 19 July 2002, and UDDI Version 2 XML Schema. The optional Security specifications include RFC2818: HTTP Over TLS, RFC2246: The TLS Protocol Version 1.0, The SSL Protocol Version3.0, RFC2459: Internet X.509 Public Key Infrastructure Certificate and CRL Profile. Since this Web Services stack is specified for general usage purposes beyond businesses, no business specification (e.g. ebXML) is included.

The *Basic Profile* declares a number of must-follow implementation requirement statements for all these specifications in order to achieve interoperability. These statements are labeled with strings in the format **Rnnnn**, where each **nnnn** represents a unique string of four decimal digits. By following these statements, Web services can maximally achieve interoperability within the scope of the specified standard stack.

Beyond the Basic Profile, WS-I is also working on some other important issues to further improve Web Services interoperability. These issues include defining implementation requirement statements for other valid Web Services standards and for upper level standards, e.g. Web Services-based business standards.

Presently, there is no other significant dedicated effort towards Web Services Interoperability like WS-I. No other industry-wide accepted interoperability specification beyond the Basic Profile has been claimed yet.

Backed by its members including most giant Web Services vendors, WS-I's deliverables are expected to be the leading specifications to improve Web Services interoperability. Sun Microsystems claims its newly released J2EE1.4 is conformant to the WS-I Basic Profile.

3 - 2. Security

The Internet is a global infrastructure for sharing information that is accessible to all. For those Internet-based businesses, online data exchange and storage need to be kept private and secure. Normally, online information security refers to keeping data reliable, integrated and away from any illegal breach. Along with the evolving of Internet-based businesses, the volume of data exchanged and stored online is swiftly increasing. This increasing demand for data security is fostering the development of Internet and Web security technologies.

Web Services is a new Web-based mechanism, which is primarily adopted to conduct businesses processes on the Internet. Naturally, security became a critical issue for Web Services too. Web Services security is also continuously under development along with Web Services itself.

The Web Services security issue consists of both the Web security considerations and none-Web Internet security considerations. This is because the transport layer for Web Services includes not only HTTP, on which the Web is based, but also other none-Web protocols such as SMTP, FTP or BEEP. [Refer to section 1-3 for the difference between the Internet and the Web] Web Services security is still in its infancy. It primarily adopts

existing Internet and Web security technologies and adapts them into the Web Services framework.

In general, the term "security" includes Confidentiality, Authentication, Trust, Non-repudiation, Integrity, Authorization and Auditing. These features strengthen the Internet information security from different approaches. Confidentiality keeps the exchanged or stored data invisible to unauthorized entities. Authentication guarantees the parties that are exchanging data are the "true" entities as they claimed. Trust set up various trust levels for various parties involved in the data exchange, whereas Non-repudiation bars out those entities with bad records from any data sharing activities. Integrity keeps the data reaching its designated receiver unchanged. Authorization defines certain rights for each party to access various data resources. Auditing records all the activities of an entity after it entered a data resource and keep these records for security check.

In order to implement these security features, the industry has already collaborated and developed various technologies: key-based digital encryption and decryption to keep the Confidentiality; username/password, key-based digital signing and signature verification, challenge-response, biometrics, smart cards, etc. for Authentication; key-based digital signing and signature verification for Trust; key-based digital signing and signature verification, message reliability for Non-repudiation; message digest, itself authenticated with a digital signature for Integrity; application of policy, access control, digital rights management for Authorization; and various forms of logging, themselves secured to avoid tampering for Auditing. To implement secure Web Services solutions, all these features need to be carefully considered and these technologies need to be adopted and adapted.

Before June 2002, even the Web Services had already been in the stage for over two years and an institutional standard stack had already been accepted by the industry, there was no well-defined industry-accepted security standard for Web services yet. Many early Web services adopters simply use the existing Internet or Web security technologies to secure their systems.

Early adopted technologies include those of XML level: XML Digital Signature, XML Encryption and XML Key Management from W3C and IETF, Security Assertion Markup Language (SAML) from OASIS; and those of transport level: Socket Security Layer (SSL), Secure HTTP (S-HTTP), Public Key Interchange (PKI), Virtual Private Network (VPN), Kerberos, X.509, etc.

Based on these pioneering efforts, IBM, Microsoft and VeriSign developed a Web Services Security (WS-Security) Specification v1.0 and submitted it to OASIS on June 27, 2002 as a proposal. A number of other Web Services vendors, including Sun Microsystems, BEA, Intel, SAP, Cisco, etc., immediately announced their support to this specification. A newly formed Technical Committee (TC) - Web Services Security Technical Committee at OASIS is now working to improve WS-Security.

Experts from IBM and Microsoft developed a white paper - *Security in a Web Services World: A Proposed Architecture and Roadmap, Version 1.0* on 1 April 2002, which became the base of WS-Security [68]. This white paper specifies a comprehensive security model for Web service users to adopt both the existing and new security technologies. It provides an abstract model that can be employed by various Web service infrastructures. By following it, users can either easily build up secure interoperable solutions in single heterogeneous systems, or collaborate different identification

mechanics within a secure framework that meets various business requirements within various technological environments.

This white paper presents a whole view of the Web Services Security consideration beyond WS-Security. It discusses a group of Initial Specifications and a group of Follow-On Specifications.

The Initial Specifications include: *WS-Security*, which describes how to attach signature and encryption headers to SOAP messages and how to attach security tokens, including binary security tokens such as X.509 certificates and Kerberos tickets, to Web Services messages (*Security Tokens* is defined as "a representation of security-related information (e.g. X.509 certificate, Kerberos tickets and authenticators, mobile device security tokens from SIM cards, username, etc.)" in this white paper.); *WS-Policy*, which describes the capabilities and constraints of the security and other business policies on both intermediary nodes and endpoints, such as required security tokens, supported encryption algorithms, privacy rules, etc.; *WS-Trust*, which describes a framework for trust models that enables Web services to securely interoperate; and *WS-Privacy*, which describes a model for Web service providers and requesters to state subject privacy preferences and organizational privacy practice statements.

The Follow-On Specifications include: *WS-SecureConversation*, which describes how to manage and authenticate message exchanges between parties, including exchanging security context and establishing and deriving session keys; *WS-Federation*, which describes how to manage and broker the trust relationships in a heterogeneous federated environment, including support for federated identities; and *WS-Authorization*, which describes how to manage authorization data and authorization policies.

Figure 3-1 demonstrates the structure described in this Web Services Security roadmap white paper.

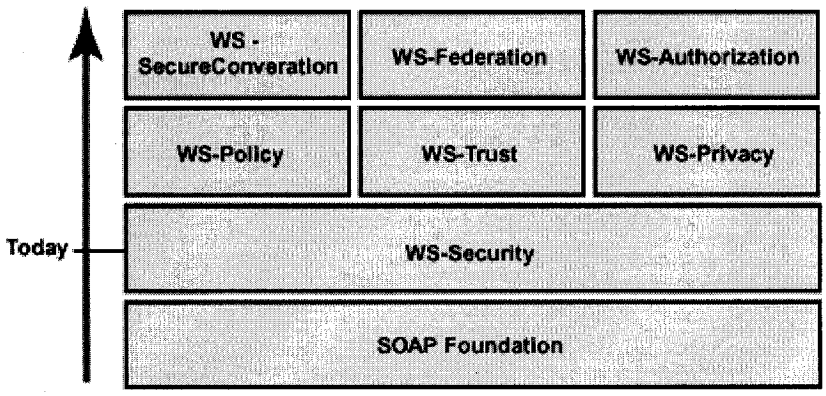

Figure 3-1 Web Services Security Specifications

(Source: *Security in a Web Services World: A proposed Architecture and Roadmap, v1.0* [68])

Without an institutionally agreed on security standard, it is hard to make Web services work together in a securely interoperative way. In January 2003, after IBM and Microsoft started collaboration on this white paper and released WS-Security 1.0, CBDI forum proved that a secure Web service implementation could be realized between a Microsoft .NET- and IBM WebSphere-based solution. This is considered a big step towards securely interoperability of Web services [72].

Security in a Web Services World: A Proposed Architecture and Roadmap, Version 1.0 is the first document that outlines the overall security mechanism for Web Services, which is composed and supported by leading E-business vendors including IBM and Microsoft. Its first initial specification – WS-Security – has already been officially developed by OASIS. Hence this roadmap is very likely to become the guide for Web Services security.

3 - 3. Choreography/Orchestration

While the basic stack of Web Services is becoming stable with XML, XML Schema, SOAP, WSDL and UDDI (considered optional at this stage), current Web Services development efforts are focused on its security and choreography. To fully explore Web Services advantages, patterns that would enable multiple Web services to interact and collaborate on accomplishing more sophisticated tasks are expected. Choreography is such a pattern that defines "how multiple web services are used together", specifies "the linkages and usage patterns involved" [1]. The term "linkages" here represents interactions between Web Services, which are implemented by sending messages between the paticipating parties such as invoking a Web service.

The Orchestration is easily to be confused with the Choreography. In many conditions these two terms are used interchangably. As one of the authors of BPEL4WS, Sanjiva Weerawarana, states: "During the lifetime of the BPEL4WS document, it was at one point called WS-Orchestration, then WS-Choreography, then WS-Business Process and and eventually BPEL4WS" [73], this two terms are historically interchangable. The chair of W3C's Web Services Choreography Working Group, Martin Chapman, believes there are some distinctions between them as Orchestration holds an managing fuction while executing the grouped Web services (such as in a transaction) and Choreography just gives the suggested execution plot without enforcing it. [73]

In W3C's Web Services Architecture Document draft published on the Web on July 1, 2003, Choreography is defined as the abstract concept for describing how Web services collaborate and exchange messages, while Orchestration is referred to as one of the techniques that realizes it [1]. In Chris Peltz's *Web Services Orchestration – a review of emerging technologies, tools and standards*, he states "Orchestration refers to an

executable business process that may interact with both internal and external web services. For orchestration, the process is always controlled from the perspective of one of the business parties. Choreography is more collaborative in nature, in which each party involved in the process describes the part they play in the interaction" [74]. This is a more commonly agreed on distinction between these two terms. However, recent development and standard convergence are blurring the distinction between them. Hence, the following discussion uses these two terms together.

There are three basic requirements for Web Services Orchestration/Choreography: asynchronous conversations, flexible and adaptable flow coordination, and exception and transaction integrity management.

A number of Choreography/Orchestration standards have already been released by either Web Services standards organizations or Web Services vendors. These standards include: eCo, Web Services Conversation Language (WSCL), XLANG and Web Services Flow Language (WSFL). eCo is developed by CommerceNet, which focuses on document exchange for B2B integrity. WSCL, however, is a simple conversation language focused on modeling the sequencing of Web Services interactions. XLANG was initially created by Microsoft and employed by Microsoft BizTalk Server. XLANG focuses on business processes creation and interactions between business providers. WSFL is an IBM proposal, which can be used to specify both public and private B2B process flows [89].

Based on these pioneering specifications, a few other specifications have been developed.

BPEL4WS is a specification that models Web services behavior for business interactions, superseding WSFL and XLANG. BPEL4WS is essentially layered on top of WSDL, whereas WSDL defines each of the operations allowed and BPEL4WS defines how these

operations collaborate together in a sequential logic. BPEL4WS is currently an OASIS standard draft under development (see Section 2-4-1).

Web Services Choreography Interface (WSCI) [90] is developed by Sun, BEA, SAP and Intalio for Web services collaboration. Unlike BPEL, WSCI does not define the whole map of how a group of Web services collaborate, but defines only the observable part in each Web service's point of view. A WSCI choreography includes a set of WSCI documents, each of which describe the should-do behaviors for each of the Web services in this group. WSCI has been submitted to W3C and becomes the basis of W3C's undertaken WS-Choreography specification.

Business Process Management Language (BPML) was developed by Business Process Management Initiative (BPMI.org), an independent organization chartered by Intalion, CSC, Sun and others. It borrows many ideas, views and syntax from WSCI so that a BPML document looks like a WSCI document. Meanwhile, BPML also provides some similar features like that in BPEL4WS, including similar process flow constructs and activities. BPML is a language designed for managing long-lived collaborated processes with persistent support.

Business Process Specification Schema (BPSS) from ebXML is another choice for Web Services Orchestration. However, since BPSS is coming as a tightly-connected component within the whole ebXML infrastructure, its relationship to other specifications is murky.

Figure 3-2 illustrates the relationship among BPEL4WS, WSCI and BPML.

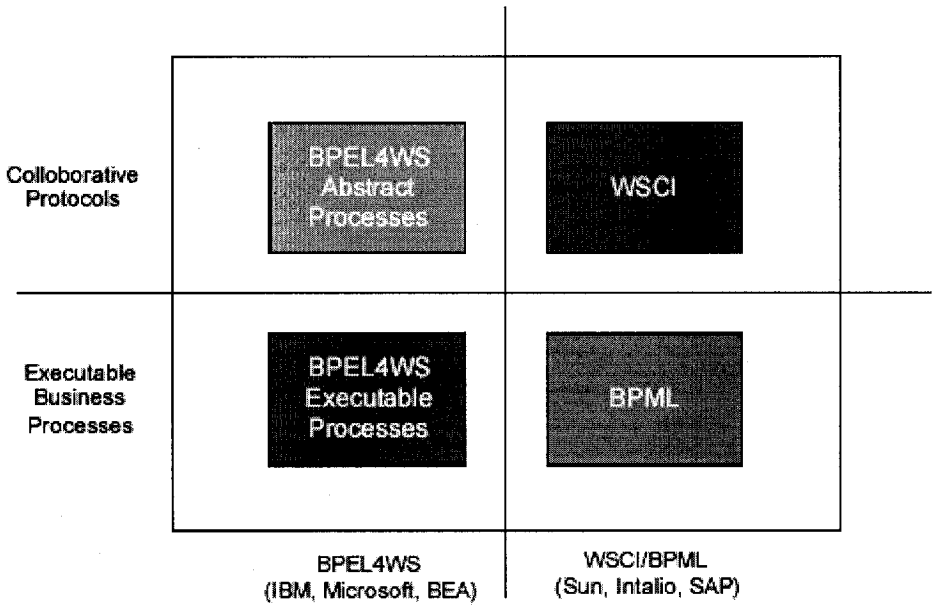

Figure 3-2 Relationships among BPEL4WS, WSCI and BPML

(Source: *Web Services Orchestration* [74])

3 - 4. Reliability

Reliability is always a critical consideration while developing a technology, especially one involving data exchanges between partner systems. One of the most significant value points of Web Services is it provides a programmable data exchanging mechanism among various application environments. Hence reliability is also a critical issue for Web Services, especially for Web Services asynchronous interactions.

Reliability for Web Services primarily means to guarantee the exchanged data to be delivered properly from its source to its destination.

Current Web Services reliability efforts are primarily focused on guaranteed message delivery, duplicate message elimination and message ordering-enabling while applying Web services.

Today, there are two approaches towards Reliability of Web Services: WS-Reliability and WS-Coordination/WS-Transaction.

WS-Reliability [77] is a royalty-free specification currently under development of OASIS Web Services Messaging Reliability Technical Committee. Fujitsu, Hitachi, NEC, Oracle, Sun and Sonic Software are the initiators of this specification.

WS-Reliability specifies a fundamentally reliable transport infrastructure in application level, which is based on SOAP but not restricted to any transport level specification. It achieves SOAP messaging reliability by defining a set of syntax and instructions that can be declared in the header and body of SOAP message Envelopes. The parties of an interaction guarantee the interaction reliability by following the instructions defined by these syntaxes to. In addition, WS-Reliability also contains an extended consideration, which makes asynchronous message exchanges as reliable as synchronous message exchanges. WS-Reliability is a complementary specification to ebXML and is proposed to be able to work with WS-Security once released.

WS-Coordination [78] and WS-Transaction are two specifications submitted to OASIS by IBM, Microsoft and BEA.

WS-Coordination defines a Web service coordination mechanism, which uses a specific Web service called *coordinator* to coordinate certain Web service activities. A Web

service *activity* refers to a set of Web services collaborating to accomplish a certain task. The task can be either simple or complicated.

Defined in WS-Coordination, a coordinator consists of three different types of components: an *Activation Service*, which is invoked by applications to create a coordination instance and the coordination context, a *Registration Service*, which enables an application to register for coordination protocols, and a set of *coordination protocols*.

While an application needs to start an activity, it first invokes the coordinator's Activation Service to create a coordinator instance and build up the environment (CoordinationContext) for this activity. Secondly, it sends the activity identifier with other context information to other participating applications. The other applications then register themselves to the Register Service of various coordinators and choose a universally accepted communication binding. Lastly, the coordinators interact with each other and their registered applications to accomplish the designated activities.

On top of WS-Coordination, WS-Transaction can be applied to ensure that a transaction is fully completed or otherwise fully roll-backed. Therefore the transaction activity can be kept consistent and integrated.

Although WS-Coordination and WS-Transaction were not developed to simply improve Web Services reliability, they can be used in this purpose when applied together.

WS-Coordination and WS-Transaction improve Web Services reliability on the transaction level when implementing business activities, whereas the WS-Reliability is located right on top of SOAP and focuses on improving common Web Services Reliability including business activities.

CHAPTER 4

Web Services Standards Organizations and Products

Each popular technology generally has a few organizations developing standards for it, a group of vendors implementing the standards by developing platforms and tools, and various users developing and deploying application solutions with these platforms and tools. Among them, standards organizations and vendors have direct influence on how the technology is evolving and where it goes.

Web Services derives from existing Web technologies. The major Web technology standards organizations then naturally play a major role in the evolution of Web Services. These organizations include World Wide Web Consortium (W3C), Organization for the Advancement of Structured Information Standards (OASIS), Internet Engineering Task Force (IETF) and United Nations Center for Trade Facilitation Electronic Business (UN/CEFACT). Some Web Services-specific standards organizations have also been formed to meet the various demands of this fast evolving technology. These new organizations include Web Services Interoperability (WS-I), Universal Description, Discovery and Integration of Web Services (UDDI), etc.

A number of E-business vendors play an important role in applying Web Services technologies. They continuously perceive demands from the market, analyze the demands and interpret them into technical ideas and designs, actively participate and influence standards organizations' work, and accelerate adoptions of standards.

Currently, the most influential computer giants and E-business pioneers, including Microsoft, IBM, Sun Microsoft, HP, BEA, Commerce One, etc., are still the most powerful force driving Web Services. They lead in the design of new specifications and infrastructures, submitting specifications to standards organizations and participating in their further development, and use their marketing power to influence users and competitors. Meanwhile, a number of small size vendors, who are strong at some specific areas in Web Services technology, are emerging.

Currently, the major vendors already have their own flagship products in the market. Most of these products are integrated into existing E-business platforms and tools as an additional part. The major products that provide complete Web Services support are IBM WebSphere, Microsoft .NET, Sun Microsoft Sun One, HP OpenView, BEA Web Logic, etc. They all have already acquired a certain proportion of the market.

Vendors generally implement a standard in various approaches, which might cause difficulties while interoperating with each other. It is more serious for implementing Web Services products since Web Services is primarily providing interoperations between systems. Hence besides the Web Services standards, vendors and organizations are also actively collaborating on implementations, e.g., developing implementation guides.

4 - 1. Standards Organizations

4 – 1 - 1. World Wide Web Consortium (W3C)

Web site: http://www.w3c.org/

Dr. Tim Berners-Lee, who invented the World Wide Web in 1989, founded W3C in October 1994 at MIT, Laboratory of Computer Science (MIT/LCS) in collaboration with European Organization for Nuclear Research (CERN). He is currently the director of W3C. W3C was founded "...to lead the World Wide Web to its full potential by developing common protocols that promote its evolution and ensure its interoperability." [23]

W3C currently has 13 offices around the world and 390 member organizations (by July 4, 2003) including three hosts: MIT/LCS of US, European Research Consortium in Informatics and Mathematics (ERCIM) of France and Keio University of Japan. Its members are primarily active vendors and organizations on Web technologies including IBM, Microsoft, Sun, BEA, etc. Each member organization has a seat in the W3C Advisory Committee (AC), which elects an Advisory Board that provides guidance to the W3C Team on strategy, management, legal matter, process, and conflict solution issues. W3C has a W3C Team that consists of more than sixty researchers and engineers, who work at the three host institutions, lead the technical activities and conduct the operations of W3C.

W3C Activities are conducted by three kinds of groups including Working Groups for technical developments, Interest Groups for other generic works, and Coordination Groups for coordinating multiple groups. These groups develop technical reports and open source software, as well as provide standard related services. Members of these

groups are individuals from W3C member organizations or the Team, and invited experts. These groups are divided into four categories: the Architecture Domain, which develops the underlying Web technology standards, Interaction Domain, which improves the interaction between Web users and content providers, Technology and Society Domain, which augments the existing Web to address social, legal and public issues, and Web Accessibility Initiative, which fully extends the Web availability.

W3C receives original proposals, which are referred to as Activity Proposals, and then distributes them to all its members for review. Once the proposal has reached a consensus, W3C forms an Activity to pursue further development of the proposal. The Activity first works out a working draft and publishes it on the Web for members and the public to review. Regarding to these reviews, the Activity groups develop the working draft into a candidate recommendation, which W3C believes satisfies the Activity requirement, and publishes it for implementation experience.

Once this candidate recommendation is mature after wide review, it is submitted to the AC and the director for final endorsement, and is referred to as a proposed recommendation. After the endorsement, it becomes W3C's official specification, and is now called a recommendation. It is like a standard in other organizations and is encouraged by W3C to be widely implemented.

By July 4, 2003, W3C has approximately 60 recommendations and proposed recommendations, 16 candidate recommendations, and 130 working drafts. All these recommendations are free of charge for people to use.

4 – 1 - 2. Organization for the Advancement of Structured Information Standards (OASIS)

Web site: http://www.oasis-open.org/

OASIS was originally formed in 1993 under the name SGML Open. It was formed by vendors and users to develop interoperability for using SGML. In 1998, it adopted its current name, OASIS, to reflect its expanded technical scope.

OASIS currently has over 600 corporate and individual members located in over 100 countries. As a non-profit organization, OASIS drives the "development, convergence and adoption of e-business standards."

OASIS currently works to produce global technical standards of security, Web Services, XML conformance, business transactions, electronic publishing, topic maps, and e-market interoperability. OASIS is active in collaborations creating global e-business standards. It jointly sponsored the ebXML project [61] with United Nations Center for Trade Facilitation and Electronic Business (UN/CEFACT). Meanwhile, it hosts rich technical information Websites including http://www.xml.org/, http://www.coverpages.org/, and supports a variety of networked sub organizations including http://www.uddi.org/, http://www.cgmopen.org/, http://legalxml.oasis-open.org/ and http://www.pkiforum.org/.

OASIS forms Technical Committees (TCs) for developing standards.

To initiate a TC, a proposal has to be submitted by at least three OASIS members. Within 15 days, the board of directors will notify the proposal submitters of its decision. Once it is approved, OASIS will broadcast a call for forming a TC among the members and also may invite some outside experts. A TC is formed and named with at least three members.

TC workouts may either TC Specifications or OASIS standards. A TC Specification is a completed work within the TC just before approval. After a TC specification is submitted, OASIS will call a review and vote on this specification among all its members. If it is approved by at least 2/3 of the members and objected to by less than 1/4 of the members, the TC specification can become the OASIS standard. OASIS encourages implementations of TC specifications even before they become standards since normally there would be no difference between a TC specification and a standard. OASIS registered a namespace to uniquely represent each document created by TCs. The namespace looks like this:

urn:oasis:names:tc:{tc name}:{type}{:subtype}?:{document-id}

While XML and other SGML-derived technologies are growing and becoming dominant in the Web, OASIS and W3C start to have more overlaps. Today, Web Services standards are partly developed by W3C and partly developed by OASIS. Praised for its high efficiency, OASIS is attracting more proposals form Web Services vendors.

4 – 1 - 3. Internet Engineering Task Force (IETF)

Web site: http://www.ietf.org/

The IETF is an open-to-all global organization that collaborate the efforts of interested network designers, operators, vendors and researchers to improve the architecture and smooth operations of the Internet. IETF activities are under the guidance of the Internet SOCiety (ISOC) and some of its subgroups.

ISOC is a professional global organization with over 150 organization members and 16,000 individual members in over 180 countries. It is the world's leading entity dealing with Internet-related issues, which is managed by a Board of Trustees elected by its

members. ISOC chartered the Internet Engineering Steering Group (IESG) and Internet Architecture Board (IAB) to guide IETF activities. The IESG consists of Area Directors that technically manage the IETF activities and supervise the Internet standard process. The IAB focuses on the architecture issues and long-range planning of the Internet, and coordination of the IETF activities.

IETF activities are conducted by working groups, each of which focuses on a specific area of Internet technologies. Working groups are directed by Area Directors (ADs) from IESG. The chair of IESG and IETF is the General Area Director (GAD), who used to be a member of IAB, and supervises the activities in IETF and IESG. IAB provides an architectural guidance to IETF works.

Anyone can propose an IETF standard. The proposal is sent to IAB for evaluation and adjustment. Once it is approved, the IAB refers it to the proper IETF working group. This proposed document thus becomes an Internet Draft (I-D). An I-D is published to receive comments from all interested parties and to be revised. Once it is well revised, the AD of the working group sends it to the IESG for any further necessary changes to be approved. The final version of the Request For Comment (RFC) is worked out and published by RFC editors.

4 - 1 - 4. United Nations Center for Trade Facilitation Electronic Business (UN/CEFACT)

Website: http://www.unece.org/cefact/

UN/CEFACT is an open organization of United Nations (UN) state members, intergovernmental organizations, and Economic and Social Council of the United Nations (ECOSOC) recognized private sector and industry associations. It was formed in 1996 to

better apply the newly developed technologies and to better make use of UN/ECE's abundant resources. ECOSOC is the highest UN body in the areas of economics, trade and development. The UN/CEFACT resides within the Economic Commission for Europe (UN/ECE), which directly reports to ECOSOC. This hierarchy structure makes UN/CEFACT the ideal body to collaborate the expert efforts around the world on making E-business standards that better meet various global business requirements.

UN/ECE was the creator of the first widely adopted E-business standard – Electronic Data Interchange (EDI). This accumulation of a variety of technical and business knowledge is applied in the collaborative work on ebXML between UN/CEFACT and OASIS. Moreover, the ebXML is developed to keep maximum consistency to the EDI standard.

4 – 1 - 5. Web Services Interoperability Organization (WS-I)

Web site: http://www.ws-i.org/

WS-I was created in February 2002 by a group of key players in the Web Services area, which include Accenture, BEA, HP, IBM, Intel, Microsoft and SAP. All WS-I members are organizations that are relevant to Web Services standards, especially implementations. It consisted of about 150 members by July 4, 2003, including almost all of the important Web Services vendors, developers and users.

While there are a lot of standards organizations defining the specifications of Web Services, users are also expecting implementation alignment and agreement of these specifications to provide interoperability and direction. The major goal of WS-I is to promote all-feature interoperability of Web services among different platforms, OS and languages provided by various vendors.

WS-I delivers *profiles* that guide the implementation of groups of Web Services specifications to guarantee their interoperability, *usage scenario* and *use cases* to reflect the real world business and technical requirements, *sample applications* to implement the scenarios and cases and to test and improve the profiles, and *testing tools* to test and ensure an implementation's conformance to the profiles.

All the work in WS-I is conducted by the experts of its member organizations under the supervision of a board consisting of about 20 of the most influential vendors and organizations in the Web Services industry.

Currently WS-I has delivered a Basic Profile that provides an implementation guide to a Web Services stack consisting of XML Schema 1.0, SOAP 1.1, WSDL 1.1, and UDDI 1.0 along with their corresponding testing tools. An increasing number of vendors have already announced their conformance to the WS-I Basic Profiles, including Sun J2EE 1.4, and used the testing tools to prove it.

While there are a couple of open global organizations making specifications for Web Services, WS-I has been chosen to guide the industry with its implementations. With WS-I's growing influence, a Basic Security Profile Working Group (BSPWG) was formed on April 1, 2003 led by Eve Maler from Sun Microsoft, and a Japan Special Interest Group was formed on May 30, 2003.

4 – 1 - 6. RosettaNet and other Business Specification Organizations

RosettaNet, described in 2-4-2, is one of the most influential and rapidly deployed business specification standards organizations today. While RosettaNet's accomplished standards are focused on the supply chain of four high tech sectors, other business

standards organizations are also gaining support in their specific business sectors. Today, the major concerns for the co-existence of these organizations are reducing the overlaps of their works, complementing each other in order to fully cover the business industry, and improving their interoperability.

A short list of representative business standards organizations are listed below:

RosettaNet:
Web site: http://www.rosettanet.org/

RosettaNet focuses on the supply chain standards for business sectors including Information Technology (IT), Electronic Components (EC), Semiconductor Manufacturing (SM) and Solution Provider (SP). It is a subsidiary of the Uniform Code Council Inc. (UCC), which is a leading U.S.-based organization that makes multi-industry standards for product identification and related electronic communications. More than 250,000 U.S.-based member companies are adopting UCC's standards in their supply chain control and management.

On June 3, 2003, RosettaNet and OASIS formalized a plan to collaborate their efforts on multi-industry business standards development and implementation. "Under this scenario, RosettaNet can leverage standards developed by OASIS, such as ebXML and the Universal Business Language (UBL), creating implementation-oriented solutions at a content level. OASIS, in turn, will look to RosettaNet for domain-specific input to ensure the applicability of universal standards within and between industries," said Patrick Gannon, president and CEO of OASIS [79].

eXtensible Business Report Language (XBRL):

Web site: http://www.xbrl.org/

XBRL is an XML-based, royalty-free open standards organization. It develops standards describing publication, exchange and analysis of complex financial information in corporate business reports. A consortium consisting of over 170 companies and agencies worldwide develops its standards.

Health Level Seven (HL7)

Website: http://www.hl7.org/

HL7 is an ANSI-accredited Standard Developing Organization (SDO) that works on developing application level data exchange, management and integration standards. These standards provide guidelines, methodologies and other services for health care information systems to exchange information in a flexible and cost-effective interoperable way.

HL7 was initiated by about 20 countries and regions in 1996. Its members include users, vendors and consultants.

4 - 2. Key Web Services Vendors And Products

Today, Web Services are being collaboratively developed by multiple standards organizations. No single organization is able to host the whole Web Services technology family or even just the basic stack. Hence, influential Web Services vendors such as IBM, Microsoft, Sun, BEA and others are taking part in almost all the collaborations on standards development by actively presenting themselves in these influential standards

organizations. These companies frequently submit their own technological workouts to these organizations to be further developed into new industry standards. These vendors profoundly influence the development of industry standards and technological direction. Therefore they are playing the most important role in the Web Services paradigm.

Web Services products are still fast evolving. While existing major E-business platform and tools still dominate the technology development and deployment, new products keep emerging to catch this new technical wave after the World Wide Web. IBM WebSphere, Microsoft .NET, Sun Microsoft Sun Open Net Environment (Sun ONE), HP OpenView and BEA Weblogic Platform are the Web Services platforms that provide the most functionalities. These platforms can all be used to develop, test, deploy, register and manipulate Web services.

CHAPTER 5

Web Services: What It Brings

Web Services emerged only four years ago, if we consider the appearance of XML-RPC and SOAP as its starting date. Web Services is now one of the most popular E-business technologies, and is evolving very quickly. Figure 5-1 illustrates how the E-business technologies evolved in the past 30 years.

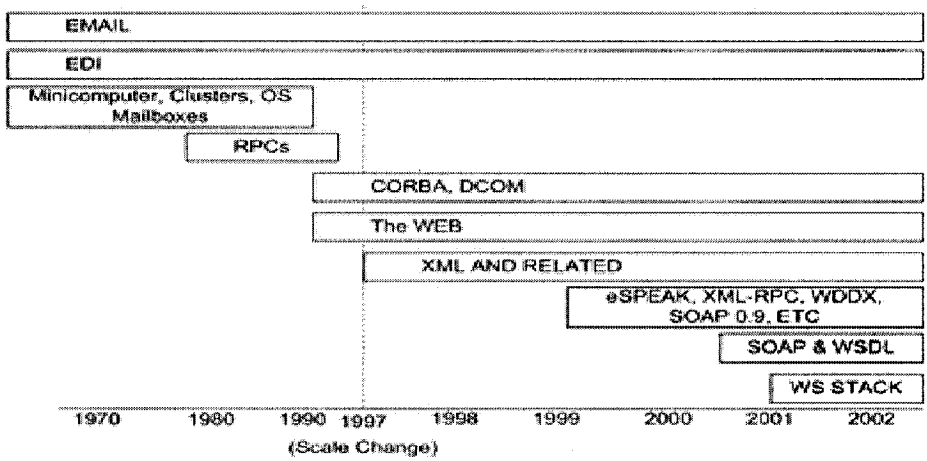

Figure 5-1 History of Distributed Computing

(Source: *The past, present and future of Web Services, Part I* [80])

The emergence of Web Services is driven by the increasing demand of machine-to-machine, primarily B2B, interactions and collaborations from the business industry.

In the last few years, Web page-based E-business has been a hot topic. It provides a networked, integrated and easy-to-use GUI interface to interact with human customers. It dramatically reduces human work to conduct business processes.

Today, Web Services based E-business is starting to take the stage center. Web Services further reduces human intervention, which is essential for Web page-based E-business, and enables machines (actually Web services) to automatically interact and collaborate to conduct business processes. Ideally, humans just need to design interoperation "agreements" among business partners and then "choreograph" the business processes. Web services will be able to conduct all of the executions intelligently without human intervention.

In the future, as Tim Berners-Lee described in his papers, Web Services will contribute and lead to the emergence of *Semantic Web*. "The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation." [82] In a Semantic Web, programs (called agents) would be able to intelligently interact and collaborate to accomplish various kinds of sophisticated tasks.

Web Services is tremendously improving the development and deployment of Web-based application systems. It can reduce development time, labor and cost, and dramatically improve application compatibility and reusability. Meanwhile, it enables highly interoperable application collaboration among different systems and infrastructures. This

section will present a perspective of the advantages Web Services brings.

5 - 1. Web Services Provides A Growing Publicly Available API Across The Web

Web services are represented by the Web service providers, which are normally objects that provide certain functionalities, across the Internet. Those functionalities provided by Web service providers include manipulations and computations of the data passed over in the requests, choreography of a series of other applications across the Web as requested, etc.

Generally, Web service providers and requesters both are implemented as OO programs. A Web service requester can invoke a Web service provider by simply sending a request message. From the service requester side, invoking a Web service is like calling a function in its API library. (Although in the implementation underneath, Web services are invoked through a special interface.) However, Web services are more flexible: first, Web service providers can be objects on another system across the network; second, Web service providers can be written in any language and running on any platform, which may be different from those of Web service requesters.

A popular example of free Web services provided on the Internet is temperature report. Many of the websites that provide trial Web services offer such a service, which can return current temperature of the area specified in the service request. For example, if an application needs to acquire current temperature in Boston, it can simply invoke either a Web service provider on IBM's website written in Java and running on AIX, or another Web service provider on Microsoft's website written in C++ and running on Windows

Server. As long as the message exchanges follow the same protocol stack binding, the service requester does not need to care what language or platform the service provider uses. The service requester can use such a Web service just like calling a local function through a local API.

Today, Web services are primarily deployed internally within business systems or between business partners. They provide a manner that enables programs developed with different languages to freely interoperate with each other across different platforms through a common API. With more and more chargeable and free Web services being publicly available over the Internet, this API may be enriched without a limit and all applications in the Internet would be able to benefit from it.

5 - 2. Web Services Introduces A Loosely Connected Application Architecture

After a number of years' efforts, enterprises have built up powerful and complicated computer systems. These systems may include self-developed applications and purchased software including those of Enterprise Resource Planning (ERP), Customer Relation Management (CRM), Manufacturing Resource Planning (MRP), Human Resource (HR), Management Information System (MIS), transaction systems among business partners, etc. These systems are normally developed in different languages, running on various platforms, and sometimes even with a different network structure. To fully and efficiently utilize these systems' capability, they need to be integrated through Enterprise Application Integration (EAI).

Before Web Services, developers had to define strict and rigid interfaces between invoked applications and the predetermined calling applications. The development had to be carefully scheduled to make sure each invoking application and invoked application

would work properly before going to the next developing stage. Picking up the appropriate technology to enable the interoperability among various application systems was a more serious challenge.

Web Services introduces a common solution for EAI in a flexible way. First, it provides a common programming and interaction interface among various application systems. Second, it does not require application systems to be tightly combined together with specific send-receive patterns or formats, but rather provides flexible patterns defined in SOAP with self-described XML messages. Third, with a flexible description and discovery mechanism, application systems can be easily redeveloped with other languages and platforms, and redeployed without affecting other functionalities.

With the development of common provisions to improve Web Services interoperability, application collaboration among business partners is becoming more real. Meanwhile, more and more Web Services organizations and vendors start to provide Web services publicly available on the Internet. With these efforts, the advantages of Web Services to provide loosely connected application architecture will be further improved.

5 - 3. Web Services Provides A Solution For *Integrated E-commerce* (IEC)

IEC represents the idea behind B2B, which means to eliminate manual processes while trading by allowing trading partners' business systems to directly exchange data.

Businesses have developed various storefronts, which provide Web page interaction to improve the automation, to reduce customers' manual paperwork.. However, if a customer needs to browse a large number of suppliers' website before accomplishing a

deal, e.g. 300 websites, this browser-server pattern will still require too many manual steps therefore diminish the benefit of IEC.

Recently, the E-commerce portal became a better solution than the storefront. It is like an ASP that collects the business categories of all the trading partners and puts them together into the portal Customers can obtain all required information by simply searching through such a portal server. Still, this solution may encounter some potential difficulties. For example, if both trading partners need to keep updating their ERP systems while interacting with the portal server, it will raise a data security concern for these business systems.

Web Services could be a powerful and convenient way to realize the IEC for business trading partners. It enables a customer system to communicate with multiple vendor systems automatically only by adopting the same interface and exchange information both with XML. Web service requesters do not directly touch the data in the service provider systems. Instead, the service provider will accomplish the task locally, reducing the menace to local system security.

5 - 4. Web Services Offers A Complementary Means For Software Service

Generally, a software package is sold in the form of Compact Disk (CD) or other media. It may raise a lot of copyright concerns, such as illegal copies, illegal installations, etc. Also, a software installation media may need to be kept for a period in order to be reinstalled or additionally installed. But the media may be broken or damaged after stored for a while. For a software producer, making, storing and delivering copies also add more cost to distributing the software. Furthermore, when a piece of software requires a more

powerful or a new platform to run on, the user has to purchase such a platform and may have to hire a technician to install it. In a case when the user only needs this software occasionally, the cost might prove prohibitive.

Before Web Services, these problems were solved by providing various Application Service Providers (ASPs). ASPs are third-party entities that manage and distribute software-based services and solutions to customers across a wide area network from a data center.

However, ASPs still have a number of disadvantages. Firstly, they cannot provide application services for all available software because of legal or other limitations of the software itself. Secondly, how to utilize the application services varies from one software product to another, thus the aggravating its complexity. Thirdly, users need to log in to acquire the services, which makes the security and reliability management of the ASP site a significant consideration. Furthermore, an ASP generally is not the owner of the software, so that its ability to improve the service is limited.

Adopting Web Services infrastructure can eliminate these disadvantages of ASPs. Each component function of a piece of application software can be presented as a Web service or a group of Web services. By this means, the copyright control, usability, system security and reliability, etc., all can be actualized by providing a universal invoking interface to the users.

Figure 5-2 illustrates an example that can apply Web services to provide software services.

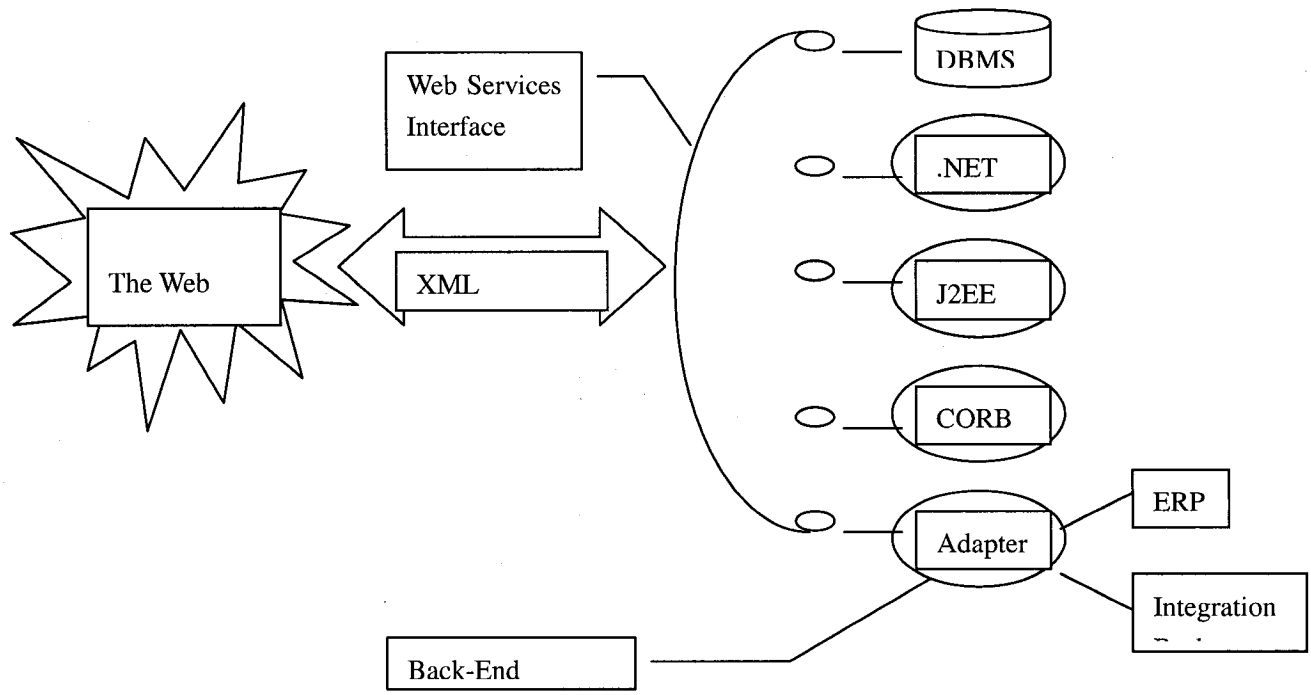

Figure 5-2 Software Service Examples

5 - 5. Web Services Enables Grid Computation Across The Web

The Web is moving into an era of computation collaboration. The idea of computation collaboration has already expedited a few vendor-specific grid computation technologies. "Under the coaxing of IBM, Sun Microsystems, Hewlett-Packard and others, grid computing has been moving into the commercial realm." [83] IBM has already announced its grid computation service to the Internet.

Although there are a number of more efficient technologies than Web Services available for grid computation, these technologies are still vendor-specific, and often are not Web-based. These two attributes limit grid computation to be adopted only by a small

group of users across the Web.

Web services are performed completely locally on the service providers' systems. It is therefore possible to organize grid computations by appropriately provide a series of Web services. Web Services-based grid computation can be implemented across a variety of vendor-specific platforms and provide service to all users across the Internet. While more and more Web services are becoming available on the Web, the Web Services-based grid computation may become more powerful.

5 - 6. Web Services Fosters Flexible Low-cost Worldwide Inter-business Solutions

Without Web Services, when two business parties want to set up a B2B connection, they must either rely on Web pages, with which human intervention is necessary, or carefully design and develop a stringent interface between their two business systems. For any negotiated application that needs to communicate with the other side, a corresponding application that performs the data exchange strictly following the predefined steps must exist on the other side. Furthermore, which language and platform to be used and which network infrastructure to be adopted are also critical restrictions while considering compatibility. These constraints not only affect the system development, but also restrict its further maintenance and improvement. Businesses are frustrated by the complexity and stringent constraints of implementing Inter-business solutions.

With Web Services, the whole process becomes much easier. First, the parties can apply the loosely connected architecture introduced by Web Services. Secondly, each party can independently choose its own platforms and network infrastructure. Thirdly, each party

can flexibly draw its own development and deployment schedule under the overall negotiated frame. Fourthly, once the Web services are accomplished, other Inter-business systems can simply employ it without any modification. Furthermore, Web services can be available across the Internet, which makes the cost of communications low for Inter-business solutions.

Along with other advantages, Web Services is widely expected to foster global inter-business solutions across the Internet.

5 - 7. Web Services Promotes Possibility Towards Semantic Web [82]

The inventor of the World Wide Web, Tim Berners-Lee, who is currently the director of W3C, has proposed a new concept – The Semantic Web. It describes an intelligent Web on which applications can understand each other's terms and logic, which are not stringently standardized.

In a Semantic Web, programs are able to invoke and collaborate with each other. A human can give an abstract order to an agent program in the Semantic Web. This agent is able to intelligently analyze and understand the semantic meaning of the abstract order. It hence selects, organizes, executes and communicates with a series of other programs to accomplish the task defined by the original order. Finally, the agent returns the required result to the user.

In principle, the Web Services mechanism contributes a common communication mechanism for agents to collaborate tasks and exchange messages, which takes a further step towards the ultimate goal of Semantic Web.

CHAPTER 6

Conclusion

Still in its infancy, the Web Services mechanism holds tremendous potential to be continuously developed and applied. The advantages it brings in, especially the common interaction interface, would help the Internet utilization to enter a new paradigm.

Today, the Web Services infrastructure is primarily adopted for improving Enterprise Application Integration (EAI), such as in the systems of Amazon and eBay. Web Services breaks the interoperability barrier between various application systems.

The rising application of Web Services is to implement application collaboration among a group of business partners, each of which is trusted by the group and agrees to participate. These groups may grow into big industrial alliances. They will still be considered as proprietary systems since trust is granted by human to intended business partners only.

In the future, Web Services may lead the Web to provide open services across the Internet with a highly secure and automated infrastructure, which is likely a part of the Semantic Web. IBM, Microsoft, Google, XMLMethods, and other institutions have already attempted to provide publicly available Web services on their websites. The Web Services

mechanism is expected to be as popular as today's browser-server model for the Internet.

However, besides the obvious advantages it holds, the Web Services may also facing potential technological obstacles for its development. For example, while too many standards are developed, people have a lot of choices to build up their own Web Services technology stack. This may easily decrease the interoperability between Web services systems, and then discourage further development of this young technology. Furthermore, lack of a universally agreed on standards for security and other feature considerations is also a disadvantage that holds Web Services from further development and deployment.

While E-business is the industry that would benefit the most from Web Services, the advantages of this technology are yet far from fully realized by the E-business society, especially those application users who really have the power to decide whether or not investing on this new technology.

The industry has already noticed these potential obstacles and started collaborating on eliminating the negative points. It might take time. But according to the active work that is being undertaken, many people are still optimistic about the future of Web Services.

REFERENCES

[1] David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris and David Orchard, *Web Services Architecture, W3C Working Draft 8 August 2003*, W3C, at http://www.w3.org/TR/ws-arch/.

[2] http://www.webopedia.com/.

[3] Aaron Skonnard, *The Birth of Web Services*, the October 2002 issue of MSDN Magazine, at http://msdn.microsoft.com/msdnmag/issues/02/10/xmlfiles/default.aspx.

[4] *Web Services Toolkit for Mobile Devices,* 25 July 2003, IBM, at http://www.alphaworks.ibm.com/tech/wstkmd.

[5] http://www.soaplite.com.

[6] T. Berners-Lee, R. Fielding and L. Masinter, *Uniform Resources Identifiers (URI): Generic Syntax*, IETF, August 1998, at http://www.ietf.org/rfc/rfc2396.

[7] R. Fielding, *Relative Uniform Resource Locators*, IETF, June 1995, at http://www.ietf.org/rfc/rfc1808.

[8] R. Moats, *URN Syntax*, IETF, May 1997, at http://www.ietf.org/rfc/rfc2141.

[9] *Naming and Addressing: URIs, URLs, ...*, W3C, at http://www.w3.org/Addressing/.

[10]Ethan Cerami, *Web Services Essentials, First Edition*, 2002, O'Reilly & Associates, Inc.

[11]Robin Bloor, *Governments Don't Understand The Electronic Economy*, Bloor Research, 26 September 2000, at http://www.it-director.com/article.php?articleid=1283.

[12]*The Difference Between the Internet and the World Wide Web*, Webopedia.com, at http://www.webopedia.com/DidYouKnow/Internet/2002/Web_vs_Internet.asp, Appendix A.

[13]*About The World Wide Web*, W3C, at http://www.w3c.org/WWW/.

[14]Bill Gates, *Microsoft .NET Today*, 14 June 2001

[15]http://www.ws-i.org/.

[16]BeepCore.org, at http://www.beepcore.org/beepcore/home.jsp.

[17]Stephen A. Thomas, *HTTP Essentials*, 2001, Wiley Computer Publishing

[18]Eric Newcomer, *Understanding Web Services: XML, SOAP, WSDL and UDDI*, 2002, Addison-Wesley.

[19]R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach and T. Berners-Lee, *Hypertext Transfer Protocol -- HTTP/1.1, RFC2616*, June 1999, IETF.

[20]Charles F. Goldfarb, *Charles F. Goldfarb's XML Handbook, 4th Edition*, 2002, Prentice Hall.

[21]http://www.sgmlsource.com/.

[22]Dino Esposito, *XML Language*, in the June 1999 issue of *Microsoft Internet Developer.*

[23]*About the World Wide Web Consortium (W3C)*, W3C, at http://www.w3c.org/Consortium/.

[24]Tim Bray, Jean Paoli, C. M. Sperberg McQueen and Eve Maler, *Extensible Markup Language 1.0 (Second Edition), W3C Recommendation 6 October 2000*, W3C at http://www.w3.org/TR/REC-xml.

[25]Aaron Skonard, *Understanding XML Namespaces*, July 2002, Microsoft Corp. and CMP Media LLC.

[26]Tim Bray, Dave Hollander and Andrew Layman, *Namespace In XML, World Wide Web Consortium 14-Janauary-1999*, at http://www.w3.org/TR/REC-xml-names/.

[27]Henry S. Thompson, David Beech, Murray Maloney, and Noah Mendelshon, *XML schema part 1: Structures, W3C Recommendation 2 May 2001*, W3C at http://www.w3c.org/TR/xmlschema-1/.

[28]Paul V. Biron and Ashok Malhotra, *XML schema part 2: Datatypes, W3C Recommendation 2 May 2001*, W3C at http://www.w3c.org/TR/xmlschema-2/.

[29] David C. Fallside, *XML schema part 0: Primer, W3C Recommendation, 2 May 2001*, W3C at http://www.w3c.org/TR/xmlschema-0/.

[30]*Cascading Style Sheet*, W3C, at http://www.w3.org/Style/CSS/.

[31]*The Extensible Stylesheet Language Family (XSL)*, W3C, at http://www.w3.org/Style/XSL/.

[32]*Document Object Model (DOM)*, W3C, at http://www.w3c.org/DOM/.

[33]*About SAX*, at http://www.saxproject.org/.

[34]*About XML-DEV*, XML.org, at http://www.xml.org/xml/xmldev.shtml.

[35]Paul Grosso, Eve Maler, Jonathan Marsh, Norman Walsh, *XPointer Framework, W3C Recommendation 25 March 2003*, W3C, at http://www.w3.org/TR/xptr-framework/.

[36]James Clark, Steve DeRose, *XML Path Language (XPath) Version 1.0*, W3C, at http://www.w3.org/TR/xpath.html

[37]Steve DeRose, Eve Maler, David Orchard, *XML Linking Languae (XLink) Version 1.0*, W3C, at http://www.w3.org/TR/xlink/.

[38]Scott Boag, Don Chamberlin, Mary F. Fernandez, Daniela Florescu, Jonathan Robie

and Jerome Simeon, *XQuery 1.0: An XML Query Language, W3C working draft 22 August 2003*, W3C, at http://www.w3.org/TR/xquery/.

[39] *Cover Stories*, OASIS, at http://xml.coverpages.org/ni2003-06-12-b.html.

[40]Dave Winer, *Dave's History of SOAP, Sat, Sep 25, 1999*, at http://www.xmlrpc.com/stories/storyReader$555, Appendix B.

[41]Dave Winer, *XML-RPC Specification, 15 June 1999*, Userland Software Inc. at http://www.xmlrpc.com/spec.

[42]Nilo Miltra, *SOAP Version 1.2 Part 0: Primer, W3C Recommendation 24 June 2003*, W3C at http://www.w3.org/TR/soap12-part0/.

[43]Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, and Henrik Frystyk Nielsen, *SOAP Version 1.2 Part 1: Messaging Framework, W3C Recommendation 24 June 2003*, W3C at http://www.w3.org/TR/soap12-part1/.

[44]Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Henrik Frystyk Nielsen, *SOAP Version 1.2 Part 2: Adjuncts, W3C Recommendation 24 June 2003*, W3C at http://www.w3.org/TR/soap12-part2/.

[45]John Cowan and Richard Tobin, *XML Information Set, W3C Recommendation 24 October 2001,* at http://www.w3.org/TR/xml-infoset/.

[46]Roberto Chinnici, Martin Gudgin, Jean-Jacques Moreau, and Sanjiva Weerawarana, *Web Services Description Language (WSDL) Version 1.2 Part 1: Core Language, W3C*

Working Draft 11 June 2003, W3C at http://www.w3.org/TR/wsdl12/.

[47]Martin Gudgin, Amy Lewis, and Jeffrey Schlimmer, *Web Services Description Language (WSDL) Version 1.2 Part 2: Message Patterns, W3C Working Draft 11 June 2003*, W3C at http://www.w3.org/TR/wsdl12-patterns/.

[48]Jean-Jacques Moreau and Jeffrey Schlimmer, *Web Services Description Language (WSDL) Version 1.2 Part 3: Bindings, W3C Working Draft 11 June 2003*, W3C at http://www.w3.org/TR/wsdl12-bindings/.

[49] Tom Bellwood, Luc Clément, David Ehnebuske, Andrew Hately, Maryann Hondo, Yin Leng Husband, Karsten Januszewski, Sam Lee, Barbara McKee, Joel Munter, and Claus von Riegen, *UDDI Version 3.0 - UDDI Spec Technical Committee Specification, 19 July 2002*, OASIS at http://uddi.org/pubs/uddi-v3.00-published-20020719.htm.

[50]J. Postel, J. Reynolds, *File Transfer Protocol, RFC 959*, October 1985, IETF.

[51]J. Postel, *Simple Mail Transport Protocol, RFC821*, August 1982, IETF.

[52]Gabriel Minton, *IIOP Specification: A Closer Look*, Unix Review, 1997.

[53]Ed Dumbill, *XML Watch: Bird's Eye BEEP*, December 2001, at http://www-106.ibm.com/developerworks/xml/library/x-beep/.

[54]Ed Dumbill, *XML Watch: Worm's Eye BEEP, March 2002*, at http://www-106.ibm.com/developerworks/library/x-beep2.html.

[55]Frank Leymann and Dieter Roller, *Business processes in a Web services world - A quick overview of BPEL4WS*, August 2002, IBM at http://www-106.ibm.com/developerworks/webservices/library/ws-bpelwp/.

[56]Tony Andrews, Fancisco Curbera, Hitesh Dholakia, Yaron Goland, Johannes Klein, Frank Leymann, Kevin Liu, Dieter Roller, Doug Smith, Satish Thatte, Ivana Trickovic, and Sanjiva Weerawarana, *Business Process Execution Language for Web Services – Version 1.1*, 5 May 2003, Copyright BEA, IBM, Microsoft, SAP, Siebel.

[57]http://www.RosettaNet.org/.

[58]*RosettaNet Background Information*, RosettaNet, at http://www.rosettanet.org/background.

[59]*How ebXML Will Transform the Software Industry* at http://www.webservices.org/index.php/article/articleview/399/1/22/, Appendix C.

[60]*About ebXML*, OASIS, at http://www.ebxml.org/geninfo.htm.

[61]http://www.ebxml.org/.

[62] Benoît Marchal, *ebXML: Introducing The Vision*, at http://www.developer.com/xml/article.php/2204681.

[63]ebXML Technical Architecture Project Team, *ebXML Technical Architecture Specification v1.0.4*, ebXML, 16 February 2001.

[64]OASIS ebXML Messaging Services Technical Committee, *Message Service Specification version 2.0*, OASIS, 1 April 2002.

[65]Mike Rawlins, Mark Crawford, Don Rudie, Thomas Warner, Kenji Itoh, Jean Kubler, Kathleen Tyson-Quah, David R.R. Webber, Garrett Minakawa, Turochas Fuad, Dr. Marcia McLure, Norbert Mikula, Christopher Lueder, Scott Hinkelman, Ravi Kackar, Doug Hopeman, Gaile L. Spadin, Sangwon Lim, *ebXML Requirement Specificationv1.06*, UN/CEFACT and OASIS 2001.

[66] Henrik Frystyk Nielsen and Hervé Ruellan, *SOAP 1.2 Attachment Feature*, W3C, at http://www.w3.org/TR/2002/WD-soap12-af-20020924/

[67]Robert McMillan, *Web Services Visionary*, 17 June 2003, IBM, at http://www-106.ibm.com/developerworks/webservices/library/ws-samruby.html, Appendix D.

[68]*Security in a Web Services World: A Proposed Architecture and Roadmap - A joint security whitepaper from IBM Corporation and Microsoft Corporation Version 1.0*, 7 April 2002, IBM & Microsoft at http://www-106.ibm.com/developerworks/webservices/library/ws-secmap/.

[69]Berni Dwan, *Web Services – Not Quite There Yet*, *Computer Fraud & Security* October 2002, Page 14-16.

[70]Barbara Gengler, *Web Services Push*, *Computer Fraud & Security*, April 2002, Page 4.

[71]Janice J. Heiss, *The Future of Web Services Security: A Conversation with Eve Maler*, March 2003, Sun at http://java.sun.com/features/2003/03/webservices-qa.html.

[72]Elspeth Wales, *Web Services Security, Computer Fraud & Security* Feb. 2003 issue.

[73]Roger Cutler, *RE: "Orchestration" and "Choreography"*, W3C, at http://lists.w3.org/Archives/Public/www-ws-arch/2002Aug/0224.html

[74]Chris Peltz, *Web Services Orchestration, a review of emerging technologies, tools and standards*, January 2003, Hewlett Parkard.

[75]Web Services Interoperability Organization, *Web Service Profiles – An Introduction version 1.0*, 6 February 2002, IBM & Microsoft.

[76]Keith Ballinger, David Ehnebuske, Martin Gudgin, Mark Nottingham, and Prasad Yendluri, *Basic Profile Version 1.0 Working Group Approval Draft Date: 2003/05/20*, WS-I.

[77]Colleen Evans, Dave Chappell, Doug Bunting, George Tharakan, Hisashi Shimamura, Jacques Durand, Jeff Mischkinsky, Katsutoshi Nihei, Kazunori Iwasa, Martin Chapman, Masayoshi Shimamura, Nicholas Kassem, Nobuyuki Yamamoto, Sunil Kunisetty, Tetsuya Hashimoto, Tom Rutt, and Yoshihide Nomura, *Web Services Reliability (WS-Reliability) version 1.0*, 8 January 2003, Fujitsu, Hitachi, NEC, Oracle, Sonic, Sun.

[78]Felipe Cabrera, George Copeland, Tom Freund, Johannes Klein, David Langworthy, David Orchard, John Shewchuk, and Tony Storey, *Web Services Coordination (WS-Coordination)*, BEA, IBM and Microsoft.

[79]*OASIS and RosettaNet Form Standards Development to Implementation Alliance*, at http://www.webservices.org/index.php/article/articleview/1035/1/3/.

[80]Uche Ogbuji, *The Past, Present And Future Of Web Services, Part 1*, 28 September 2002, at http://www.webservices.org/index.php/article/articleview/663/1/61/.

[81]Uche Ogbuji, *The Past, Present And Future Of Web Services, Part 2*, 7 October 2002, at http://www.webservices.org/index.php/article/articleview/679/1/61/.

[82]Tim Berners-Lee, James Hendler, Ora Lassila, *The Semantic Web, Scientific American, May 2001.*

[83] Stephen Shankland, *IBM stretches grid business*, 27 April 2003, CNET News.com, at http://news.com.com/2100-1010-998429.html.

[84]M. Rose, *The Blocks Extensible Exchange Protocol Core, RFC3080*, March 2001, IETF.

[85]Ramesh Nagappan, Robert Skoczylas and Rima Patel Srignaesh, *Developing Java Web Services – Architecting and Developing Secure Web Services Using Java*, 2003, Wiley Publishing Inc.

[86]Frank P. Coyle, *XML, Web Services and the data Revolution*, 2002, Addison-Wesley

[87]Harvey M. Deitel, *Web Services: a Technical Introduction*, 2003, Prentice Hall

[88]Doron Sherman, *BPEL: Make Your Services Flow – Composing Web Services Into Business Flow*, Web Services Journal, at http://www.sys-con.com/webservices/articleprint.cfm?id=589.

[89]Jon Udell, *Orchestrate Services*, 5 July 2002, at http://www.infoworld.com/article/02/07/05/020708plweborch_1.html.

[90] Assaf Arkin, Sid Askary, Scott Fordin, Wolfgang Jekeli, Kohsuke Kawaguchi, David Orchard, Stefano Pogliani, Karsten Riemer, Susan Struble, Pal Takacsi-Nagy, Ivana Trickovic, and Sinisa Zimek, *Web Services Choreography Interface (WSCI) 1.0, W3C Note 8 August 2002*, BEA, Intalio, SAP and Sun at http://www.w3.org/TR/wsci/.

[91]Beth Blakely, *The Future Of Web Services*, 30 August 2002, At http://www.zdnet.com.au/itmanager/trends/story/0,2000029592,20267816,00.htm.

[92]Dan Gisolfi, *The Web Services Architect: Catalysts For Fee Based Web Services*, 1 November 2001, at http://www-106.ibm.com/developerworks/webservices/library/ws-arc6/.

[93]http://www.coverpages.org/.

[94]*Web services are next IT storm, Forrester CEO says*, Webservices.org, at http://www.webservices.org/index.php/article/articleview/951/1/7/.

APPENDICES

Appendix A

The Difference Between the Internet and the World Wide Web

- Webopedia.com,

at http://www.webopedia.com/DidYouKnow/Internet/2002/Web_vs_Internet.asp

Many people use the terms Internet and World Wide Web (a.k.a. the Web) interchangeably, but in fact the two terms are not synonymous. The Internet and the Web are two separate but related things.

The Internet is a massive network of networks, a networking infrastructure. It connects millions of computers together globally, forming a network in which any computer can communicate with any other computer as long as they are both connected to the Internet. Information that travels over the Internet does so via a variety of languages known as protocols.

The World Wide Web, or simply Web, is a way of accessing information over the medium of the Internet. It is an information-sharing model that is built on top of the Internet. The Web uses the HTTP protocol, only one of the languages spoken over the Internet, to transmit data. Web services, which use HTTP to allow applications to communicate in order to exchange business logic, use the Web to share information. The Web also utilizes browsers, such as Internet Explorer or Netscape, to access Web documents called Web

pages that are linked to each other via hyperlinks. Web documents also contain graphics, sounds, text and video.

The Web is just one of the ways that information can be disseminated over the Internet. The Internet, not the Web, is also used for e-mail, which relies on SMTP, Usenet news groups, instant messaging and FTP. So the Web is just a portion of the Internet, albeit a large portion, but the two terms are not synonymous and should not be confused.

Appendix B

Dave's History of SOAP

- Sep 25, 1999 by Dave Winer

at http://www.xmlrpc.com/stories/storyReader$555

SOAP worked with Dave as they knew XML RPC was the existing protocol. It annoys me that everyone wants to be a superstar and invent new protocols without consulting people who are already doing it.

That's not exactly true. Before folklore becomes reality, XML-RPC was originally, privately, called SOAP, when Don Box and I were working with Bob Atkinson and Mohsen Al-Ghosein at Microsoft, in early 1998.

UserLand had a protocol before that called "RPC", I announced it in DaveNet, and they asked if I'd like to work with them on this.

I put a hold on our work and posted a heads-up to Frontier developers that the spec might be changing, based on the first meeting we had with the Microsoft folks.

We quickly implemented a client and server for what was then called SOAP, and Mohsen wrote a client and server too, in JavaScript I believe, and we got them working together.

Then a lot of other MS people got in the loop, and the arguing began, and it dragged on for weeks.

I talked privately with some of my friends, "What do I do?". We analyzed the choices. I could stay with the program with MS, and now we know where that would have led. We would have waited over a year, with our users in limbo.

We could go back to the "RPC" format, but we had already implemented the "SOAP" server/client, and it was much better than the old one (the old one didn't have <struct>s or <array>s).

Or we could change the name to something else and release it publicly. I asked the MS guys how they felt about this, and they said nothing. So I sucked in my breath, released the spec as XML-RPC, and waited for them to squash me (I was trained by Apple, who definitely would have crossed me off their list for not being their slave). They never squashed, and in fact, they kept inviting me to meetings to discuss this or that about their spec, which was evolving into something that would be hard to see as originating from the XML-RPC spec.

To me, it was most important to get Microsoft out publicly promoting the idea of low-tech wire protocols based on the standards of the Internet. I would have been just as happy with support from IBM, Oracle, Sun, Apple, Netscape, whatever, because I know that the value of a big name is essential in making something like this stick. Once we had a basic agreement with MS on what became XML-RPC I went on a private tour of execs in the industry, but none of them (I think) had a clue what I was talking about. Microsoft, on the other hand, as a group, got it immediately, all the way to the top, I emailed with Gates on this several times.

Internally, they acted like a standards body, with people from IETF and W3C arguing over all kinds of things. I had no time for the arguments. I basically said yes to everyone. Go for it. Let's do it, I kept saying. But it kept dragging on.

Now we're implementing our SOAP stuff, and we'll plug it in behind all the interfaces we're doing. When I say XML-RPC now, I mean SOAP *and* XML-RPC. Any interface I define now will also be a SOAP interface when we get our server running. Bierman is working on that, for now, but I expect that Andre and I will take it over soon.

Anyway, I'm rambling. The bottom line is that people *will* reinvent the wheel. It always works that way. Instead of hating them for doing it, love them for it and make sure that we can hide their differences behind scripting APIs. Then everyone can win, and no one will be threatened by repeating innovation loops.

Appendix C

How ebXML Will Transform the Software Industry

- WebServices.org

at http://www.webservices.org/index.php/article/articleview/399/1/22/

We talk to Dieter Jenz about his latest report on ebXML and its consequences for the software industry

Most people will agree that there is a definite need for a universal, standards-based business collaboration infrastructure that supports inter-enterprise as well as intra-enterprise integration, provides out-of-the-box interoperability and is available at low cost. Part of the ebXML initiative is to offer such a solution. ebXML is an initiative jointly sponsored by UN/CEFACT and OASIS and has developed a set of specifications endorsed by major industry consortia. Already there are several implementations of core ebXML specifications and organizations have started pilot projects. The most notable is the Automobile Industry in the US (www.aiag.org) who use ebXML in practice to collaborate and trade.

Dieter Jenz of Jenz & Partner has recently written a paper on the impact ebXML will have on the software industry. He shared with us some of his key findings.

He findings indicate there is a wide market for ebXML, if it can solve real problems. He states that "There is virtually no organization that is not striving for optimal leverage of IT resources, how to better automate business processes and to connect business processes with business partners. In pursuing the highly strategic objective of aligning IT with business, IT managers want interoperability, being tired of all the hassles that are incurred with plumbing together a plethora of application systems."

Until now EDI has had a high entry barrier in terms of cost and effort. This has hindered its progress in first-generation B2B, and is yet to become a widely used business collaboration technology. Dieter explains "As a consequence, enterprises had to retain their paper-based business processes to do business with business partners that have not committed to EDI."

The ebXML future

After the advent of XML, several initiatives have started to pave the way for the establishment of an XML-oriented, standards-based business collaboration infrastructure, which allows business partners to connect their business processes. Today, three major business collaboration infrastructure architectures exist: ebXML, RosettaNet and Microsoft's BizTalk.

Dieter in particular favours ebXML to lead the way forward. He says "At present, ebXML is the only non-proprietary horizontal business collaboration infrastructure architecture that provides integration at the business process level."

The question asked by many is, will ebXML work in practice? It is acknowledged that a common business collaboration infrastructure yields tremendous synergy effects and will

lead to significantly reduced costs for inter-enterprise as well as intra-enterprise business integration. Dieter is one of many who believe that the business value goes far beyond the level of current business process integration solutions.

The move to BPM

It has been widely suggested that the advent of ebXML marks an important turning point in the system integration industry towards Business Process Integration broker technology. Dieter says that the move towards application architectures "will exploit the power of multiple servers (application server, database server, presentation server) and the separation of application logic, data access logic, presentation logic, and process control logic facilitates this transition". In essence he means that the focus will shift from integration at the data level to integration at the process flow level.

Where do Web Services play a role

Web Services and ebXML will complement each other. For example a Business Process Management System (BPMS) will assume the role of a broker, participating in inter-enterprise business collaborations (public processes) and connecting them with internal business processes (private processes). The role of Application integration will fall to Web Services.

Dieter sees Web Services as the glue in this system. "Web Services provide technical interoperability in that the technology allows a service requester (the BPMS) to interact with a service provider via messages of definite language-neutral and platform-neutral format" he states.

What work remains to be done

While it is "so far so good", there is still work to be done. The fact remains that ebXML is content-agnostic, meaning that it restricts itself to providing a technical collaboration infrastructure. In effect this means it only defines the envelope used to transport content from sender to recipient and some of the processes and structures of how business could take place. The actual business documents such as an application form for an insurance product, is not specified.

Dieter told us, "The content in the form of business documents is defined by industry bodies, such as the Open Applications Group (OAG). Likewise, ebXML only defines the Business Process Specification Schema, but does not define concrete business processes".

Still much work remains to be completed on such business documents, and this is a possible hindrance to the future take up of ebXML systems. Dieter looks to the work of the UBL to provide the answers. He says "The Universal Business Language (UBL) Technical Committee is working under the auspices of OASIS. It is chartered with the development of a standard library of XML business documents. UBL is intended to become an international standard for electronic commerce. UBL is still at an early stage in development. Although UBL and OAG's Business Object Document (BOD) definitions overlap since they define content for the same things, companies should not shy away from committing to a widely recognized industry standard, which OAGIS is today."

Dieter concludes, "In our research report, we have analyzed who are the driving forces pushing towards the adoption of a standards-based business collaboration platform, who will benefit from this sweeping move and who will lose. One of our key findings it that Software vendors will OEM core infrastructure components, enabling systems integrators

to offer best-of-breed solutions at as yet unprecedented price points. Solutions will become available starting from $5,000, which makes them affordable for smaller companies."

"EAI software vendors will be affected most, forcing them to review and modify their current business strategies. EAI is a tactical issue while BPM is considered strategic. Hence, EAI vendors need to move up the "food chain", meaning that focusing on data integration is no longer a viable strategy for EAI vendors."

Appendix D

Web services visionary (Part)

Sam Ruby's job is to see into the future of Web services

- June 17, 2003 by Robert McMillan

at http://www-106.ibm.com/developerworks/webservices/library/ws-samruby.html

Sam Ruby, a member of the IBM Emerging Technologies Group, has become a key part of several Web services-related open source projects over the last three years, including Tomcat and the IBM SOAP stack. He's still contributing both his code and his insight to the community. He spoke with Bob McMillan on a number of topics, including the appeal of open source, the future of Web services, and the power of Web logs

[*developerWorks* represents the interviewer, *Ruby* represents Sam Ruby]

… …

developerWorks: There are so many emerging Web services standards, and groups that these standards can go through, and then there's this de facto open source way of developing standards. It seems very easy to get confused about what standards are emerging where, and what standards efforts are important to watch. Does this all make sense to you?

Ruby: I think the short answer to that is that I don't think anybody knows which way it's going. What we've got is lots of people with competing interests. No one person knows exactly the right answer for the future. But instead of going like we have done in the past and say, "Let's build this humongous standard like CORBA," and you've got to agree to every single aspect of it in order to have an implementation.... What we're doing collectively -- not something that IBM's doing or Microsoft's doing, but what the whole industry is doing -- is defining this, in steps. And what you get is a bit of confusion. You get a bunch of people with opposing standards -- some of which live, some of which die.

In the process, the industry -- I'm not saying IBM or anyone in specifics -- is trying lots of venues, whether it's W3C or OASIS or just simply publishing a URL out in the Web and not going through any standards body.

There have been a number of noticeable failures. Microsoft originally put out SOAP With Attachments, then later said that was a failure, and then they put out this other thing called DIME, and now they're saying that was a failure, and the best thing to do is go back with SOAP With Attachments when you've got to, but actually put the data in the XML infoset when you can.

So what you're seeing there are people trying this, seeing if it works, seeing if other people rally around it. We don't know yet which of these standards will be used five years from now. However, the basics, like SOAP and WSDL, seem to have gotten a lot of traction and don't seem to be going away.

... ...