An Overview of Web Services

Based on literary review

Socrates Krishnamurthy, Charles Stevens, Ramnath Nair

MIS 513

Instructor: Xiaoqing Li

University of Illinois – Springfield

July 21st, 2011

Executive Summary

As the Internet becomes an increasingly dominant communication and work technology, the need for full connectivity and interoperability follows. Web services can provide a competitive advantage and have almost unlimited applications.

Web services provide a means of interaction among machines through a network, while fundamentally remaining independent of the operating system in place.

Function is made possible by various network technologies and communication protocols, which collaborate to build a seamless, integrated application.

Web service design has many elementary principles, but the architectural details vary according to the service's goals and intended inputs, and outputs. Once designed, a Web service is often published, so as to expose it for potential client's to find, and ultimately use.

An extremely important aspect to developing a robust and uniformly-applicable Web service is to follow strict data description, transformation, and interpretation procedures. This is known as Web service *semantics*, and is a growing area of research.

As with any application involving network communication, security is of great concern. Information can be intercepted and/or manipulated if not properly protected through a multitude of appropriate tactics.

Since Web services are popular within the business world, constructing a reliable service is a key. Service quality, timeliness, and resiliency can make or break a corporate bond.

Web services are in use by a large number of corporations, both big and small, such as Google, Amazon, and UPS.

1.1 Why Web Service?

For Organizations operating in the digital world, data sharing and online collaboration with Suppliers, Business Partners and Customers are crucial and important. This provides Competitive edge for an organization to gain customers and protect them from other viable forces. To share the data and make external entities collaborate seamlessly, it is imperative for an organization to provide access to their data without any interruption. However, the information systems in each entity may have different operating systems. They may be located anywhere across the Globe. These entities could not use the data from a customer's database without prior knowledge of physical location, security and configuration parameters of the remote computer on the network. Web Services, aka *Remote computing Services,* are used to address these issues. "The increasing adoption of Web services is one of the most important technological trends in contemporary business organizations. This

trend is motivated by claims about the ability of Web services to facilitate information technology (IT) flexibility, improve information management, and even lead to a competitive advantage" (Lior Fink et al.).

Platform independent, network independent, device independent Web services enable entities from anywhere in the world to access and use these services through the Internet. This heterogeneous nature of web services enables seamless collaboration and data sharing between an organization and its entities. In fact, large organizations that spread across the globe started using these reusable application components across the organization increasingly for suppliers and business partners. This enables common services developed by one business unit to be used across the corporation. "Web services have a massive impact on the way businesses think about designing, developing, and deploying Web based applications. This study draws on these characterizations and views Web services applications (WSAs) as Web-based applications that are designed to facilitate the seamless integration of business software using open standards." (Ratnasingam, pp.382)

This powerful benefit of web service was recognized by many organizations, which started offering such service components to their suppliers and business partners enabling them to access the necessary part of their database in a controlled manner. This allows organizations to build and strengthen their customer relationship, streamline information flow, maintain data accuracy, eliminate data inconsistency and accelerate availability of data.

1.2 What is Web Service?

In order to know what a web service is, it is important to first know SOA techniques. SOA, aka *Service-Oriented Architecture,* is the evolution of distributed computing based on the request / reply paradigm. It is an architectural style for building software applications that use services available in a network such as the web. An application's business logic, individual functions, and/or repeatable common functionalities are modularized and presented as a service to the consumer through the network. The key to these services is their loosely coupled and interoperable nature; i.e., the service interface is independent of the implementation. Application developers and system integrators can build applications by composing one or more services without knowing the services' underlying implementations. W3C consortium 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" (W3C: WSA).

IBM defines Web services in a similar manner as "self-describing, self-contained, modular applications that can be mixed and matched with other Web services to create innovative products, processes, and value chains. Web services are Internet applications that fulfill a specific task or a set of tasks that work with many other web services in a manner to carry out their part of a complex workflow or a business transaction". According to Deploying and Managing Web services 541

to Microsoft, “A Web Service is a unit of application logic providing data and services to other applications. Applications access Web Services via ubiquitous Web protocols and data formats, such as HTTP, XML, and SOAP, with no need to worry about how each Web Service is implemented”. HP defines Web services as “modular and reusable software components that are created by wrapping a business application inside a Web service interface. Web services communicate directly with other web services via standards-based technologies”. SUN perceives a Web service as an “application functionality made available on the World Wide Web. A Web service consists of a network-accessible service, plus a formal description of how to connect to and use the service”. (Qi Yu, P540-541).

Web services are the implementation and realization of SOA. They are interoperable, heterogeneous, software components exposed as a service and made available on the Internet for the clients to consume. A group of web services interacting together in this manner defines a particular web service application in a SOA. Web services are always designed to be accessed and consumed by other applications. They vary in complexity from simple operations like conversion of temperatures, to complex processes like running CRM and ERP systems.

1.2.1 Benefits of Web Service:

Interoperability: Being most the important benefit, web services typically work outside of private networks, offering developers a non-proprietary route to their solutions. Services developed are likely, therefore, to have a longer life span, offering better return on investment of the developed service. Web Services also let developers use their preferred programming languages. In addition, thanks to

the use of standards-based communications methods, Web Services are virtually platform-independent. “Web services are designed to bring together applications from geographically distributed and heterogeneous environments and provide interoperability among them” (Sirin et al.)

Ease of Communication: Web services use SOAP over HTTP protocol. So the existing Internet network shall be used to implement web service. This is cheaper than proprietary solutions like EDI/B2B.

Code Re-use: Web services are self-contained, self-describing, discrete application components. The reusable nature of web service acts as a building block for developing a robust, enterprise application. Web services also help connect existing legacy applications to new applications.

Cost Savings: Ease of communication, code re-use, data accuracy, faster data availability, and platform, technology, and network independency adds up to significant cost savings.

1.3 Web Service Technologies

Web services architecture involves a family of layered and related technologies and protocols to describe, deliver, and interact with services. This family can further be subdivided into two groups.

a. The first group handles the issues of messaging, interface description, addressing and delivery. *Simple Object Access Protocol* (aka SOAP) is a simple

XML-based messaging protocol that allows applications to communicate information over HTTP, IIOP, SMTP, or others, without the depending on the OS platform. This protocol encodes messages so they can be delivered over the network. SOAP uses HTTP and XML as the mechanisms for information exchange. “SOAP is a WS messaging standard that enables communication among Web services in a Web Service Management System. It provides a lightweight-messaging framework for exchanging XML-based messages. SOAP is independent of languages and platforms. It supports two types of communications: messaging and RPCs” (W3C: SOAP).

Web Services Description Language (aka WSDL) is a set of XML commands that comprises of definition for the interfaces for each web service. This XML based protocol is used to send and receive information through distributed environment. “WSDL specifies the mechanisms to access a Web service in addition to the service name and signatures. Second, WSDL defines the location to invoke a Web service, whereby the service requestor can locate the service and interact with it using SOAP” (W3C: WSDL).

b. The second group of protocols and specifications define how services advertise themselves and can find each other on the network. For services to locate each other, the *Universal Description, Discovery and Integration* (UDDI) protocol defines a registry and associated protocols for locating and accessing services. “UDDI defines a standard to publish and query Web services in a WSMS. It integrates Web service description and discovery to help service requestors

locate their desirable services. The core component of UDDI is a service registry. The registry stores the general information of Web services" (W3C: UDDI).

1.3.1 Style of web services

Web services are a set of tools and technologies that can be used in the following ways:

Remote Procedure Calls (RPC): In this style of web services, a SOAP message represents a method invocation on a remote object. The arguments list the method to be invoked are serialized and moved to the remote environment in which the method is invoked.

Service Oriented Architecture (SOA): In this style, a SOAP message represents the information needed to invoke another service. This is often referred as *Message-Oriented* service.

*RE*presentational *S*tate *T*ransfer (REST): This style of web service follows Client and Server architecture. Clients initiate a request, send it to server, and the server processes the request; returning the appropriate response. This is often referred as *Operation-Oriented* service. REST web services are based on GET, POST, PUT, and DELETE. "REST WS are promising because of their lightweight and simplicity compared to SOAP-based WS" (Khaldoon).

1.4 Web Service Design

When incorporating the use of Web services, the fundamental idea behind Web services needs to be continually considered. Web services are used when communication and interoperability is necessary via heterogeneous platforms (Java, 1.1). Developing a Web service which works on only a single platform essentially defeats the purpose of the application. With this idea in mind, a generic approach to Web service models and architecture will suffice for the goals of this project.

Generic web service applications, in parallel with their original goal, often follow the aforementioned service-oriented architecture with three distinguishable steps. Some refer to this particular model framework as "Publish-Discover-Bind" (Java, 1.1). First, the web service interface itself is *published* to some sort of registry. Clients can then *discover* (Web service discover is more thoroughly discussed in the next section) the particular Web service that suits their needs, and then *binds* to it, allowing for its use. This is a general encapsulation of Web service use, but the actual implementation varies via numerous combinations of processing models and communication types. Some styles, other than service-oriented structure, have become relatively popular; including remote procedure call (RPC) and representational state transfer. Recent growing popularity in the use of XML messages has resulted in an increasing migration from RPC to REST-compliant Web services.

There are differing approaches to Web service processing models. Models may be object-centric, as well as document-centric. Object-centric approaches are controlled by method (owning to its alternate reference as the "method-centric"

approach) calls, which use the business logics of the service and apply them to objects containing the required information to complete the client's request. The incorporation, and use, of business logic is what separates object-centric models from those with a document-centric structure. Document-centric Web services separate the inherent business logic from the document's content. "The service receives an XML document, and that document contains only data and no explicit binding to the business logic that is to be applied" (Java, 1.1). In essence, request processing depends on the content details, rather than invoking specific methods, as in the object-centric model. These two ends of the model spectrum lead to unique, but often blended, types of interaction architectures.

Web service architecture can be broken into two variances: synchronous and asynchronous Web services. Their handling of requests and the resulting responses separates the two communication features. Document-centric models are often composed of asynchronous communication, in which clients request a service and then continue processing without waiting for a response from the service. Once a response is receives, the Web service resumes whatever process initiated the request. Conversely, object-centric models usually consist of synchronous messaging, in which the client waits for a response to the request before it continues on within the original process (Java, 1.2.1). For obvious reasons, processing time plays a major role in which architecture is most suitable for a given Web service. A process involving significant computation, and therefore time-consumption, would be hindered by a synchronous structure. This characteristic might explain why synchronous communication, such as RPC-oriented applications, has limited applicability, and is often overshadowed by asynchronous design.

Once a Web service has been developed, it needs to be exposed for description and discovery. Depending on its proposed use, a Web service can be published wherever the business feels it to be necessary, such as a corporate or public registry. Within the registry needs to be a description of the Web service's Web Services Description Language (the standard language for describing a service) as well as the XML references made. At this point, the Web service can function under its intended purpose. As mentioned in section 1.2, these purposes can span from simple calculations (for example: Celsius-to-Fahrenheit conversions) to fully integrated enterprise applications. Since Web services are intended to be cross-platform, their uses are practically endless.

1.5 Web Services Discovery

Web services discovery, as its title suggests, is the process behind matching a given task with a suitable Web service. As the popularity and variety of Web services continues to grow, the need for a simple search and discover method comes to light. There exist numerous individual registries for which Web services can be published, but discovery still fails to have a single solution. "Although these registries provide search facilities, they are still rather difficult to use…" (Crasso et al, 144). There are many customized versions of discovery registries that have been built by different professionals. Most, however, follow similar fundamental characteristics.

Similar to marketing your product with commercials or ads in the paper, businesses are concerned with getting their Web services in front of the correct users. Simply stated, "effective mechanisms for web service discovery and ranking

are critical for organizations to take advantage of the tremendous opportunities offered by web services, to engage in business collaborations, …to identify potential service partners…and increase the competitive edge of their service offerings" (Hao et al, 1053). Many web service discovery methods follow a similar design to the Universal Description, Discovery and Integration (UDDI) registry. UDDI enables businesses to publish their Web services, define their individual protocols (done so by providing WSDL documents), as well as explore other web applications. UDDI is, of course, independent of the operating system language and conducts itself based on XML practices. While UDDI, as an example, provides a large number of listings, that quality in of itself raises a problem.

Depending on the search query, registries like UDDL can provide too many results capable of performing the desired task. As a result, many Web service discovery methods now look to include relevance as part of the search process. Many registries now have various algorithms that can return the most fitting service, given a description of the desired task, rather than just list all of the applications that could satisfy the request. To combat the problems associated with Web services being spread out over various registries, "multicast" mechanisms such as WS-Discovery can reduce the need for centralized registries by searching entire networks. These particular applications, however, are limited to small, local networks.

1.6 Web Service Semantics

As organizations look to develop uniformly applicable Web services, it has become increasingly necessary to develop a sophisticated way in which data can be

combined from different sources and industries without losing its intended meaning. Correct data transformation, formation, and interpretation are extremely important aspects of a fluid Web service. "Semantics can improve software reuse and discovery, significantly facilitate composition of Web services and enable integration of legacy services that are described using Web Service Description Language" (Akkiraju et al.). As the quote indicates, semantics has direct implications on the previously discussed Web services discovery techniques.

The overall goal of Web service semantics is to organize descriptions in a formal, concise, and unambiguous manner, so as to allow machine interpretation of the service to be both automatic and precise (Charif et al, 33). In other words, speaking the same language, and using the same words, will result in the best communication, no matter who's speaking. "Semantic markup of Web services provides a declarative, computer-interpretable API for executing services (McIlraith et al, 47). In a semantic environment, the markup tells the agent what input is needed, what will be returned for a given request, and how to interact with the service automatically. Having predefined interpretations of a service's elements greatly decreases misunderstandings, leading to a more robust, complete interface.

Web service semantics not only aim to impose explicit understanding, but they also can make up for divergent syntax within two services. For example, suppose you wanted to get the price for a particular item. There are two separate functions available to do this: *getItemPrice()* and *costOfItem()*. They differ in syntax, such as one referring to the result as "price" and the other referring to the result as "cost." The interfaces which parameters are established could also be different.

However, if the two functions follow the same *semantics*, they will perform the same, and provide the same result.

As with Web services discovery, semantic markup has been approached in various ways through separate initiatives, such as WSDL, and OWL (Akkiraju et al.). Because of this, multiple definitions for the same service have been created. So unless everyone uses the same approach (OWL, for example), uniformity is almost impossible. A push for an industry-wide semantic markup has lead to substantial research into method standardization and composition. While some initiatives have lead to partially accepted techniques, limitations of their functionality still exist. Hopefully, business practices will eventually lead to systems based on clear, concise protocols, leading to seamless automation.

1.7 Web Services Security

Web services, which are an integral part of the Web today, have gained industry-wide acceptance because of its ability to solve IT problems using standards and standards-based technologies. They deliver a promising solution that allows IT services to be interoperable and to integrate them using XML-based messages and standard protocols. Web services security has become the most important focus, because it is necessary to ensure that exposed Web services-based business transactions are secure, reliable, and available to service consumers when they need it. When securing web services, we must take a three phase approach; first: identify the threats that affect Web service, second: define the security requirements, and third: implement a security solution.

1.7.1 Security Issues

There are many known threats, vulnerabilities, and risk factors (Shamual) for implementing Web services. Some of the known threats are:

- "Man in the middle" attack - when messages between two communication parties are intercepted and the content modified, without the knowledge either party.
- Spoofing - when an attacker uses the identity of a trusted service requester and sabotages the security of the service provider using forged service requests with malicious information.
- Replay attacks - the attacker fraudulently duplicates a previously sent request and repeatedly sends it for the purpose of causing the web services to generate faults that can cause failure.
- XML Schema tampering - the attacker modifies the externally referenced XML schemas with erroneous and inconsistent information, which affects web services failure related to message validation and verification.
- WSDL and UDDI attacks - there is a threat that the attacker will make use of publicly accessible UDDI or WSDL information to identify the service provider location and then perform a number of operations, with arbitrary input and output parameters, based on bad data. It is very important to understand and address these security issues before adopting web services in business organizations.

1.7.2 Security Requirements

Web Services also have similar security needs to any other Web based applications. Security requirements for Web services can be divided into those that are defined in terms of the communications protocols. These requirements must also ensure that a secure environment will be available to the Web services for conducting business transactions, processes and collaboration. The following security requirements must be addressed in order to be able to deliver end-to-end security architectures for web services solutions (Robert, 351):

- Confidentiality - Data privacy and confidentiality assure that the actual transmitted data is protected from unintended users. Ensuring confidentiality guarantees that data transmitted is not able to be viewed by anyone who may intercept the information during transmission between Web services client and provider.
- Integrity - Data integrity plays a vital role in ensuring that information exchanged between the requester and provider are accurate, complete, and not modified/altered during transit and storage. Ensuring data integrity protects Web services communication from intruders that may damage data.
- Non-repudiation – It must be ensured that the communicating parties accept a committed transaction. This will also assure that the sender cannot legitimately claim that they did not send the message.
- Authentication - enforces the verification of the identities and credentials exchanged between the Web service provider and the consumer. This will assure that the sender and receiver are who they claim to be. Using

authentication, the risks associated with many known threats can be mitigated.

- Authorization - defines the rules and policies associated with the required access control to the resources. Upon successful authentication, a service requester requiring access to a business service should be provided with specific access rights to the resources. It assures that an authentication entity can access only those information resources they must have access to request or provide a service.
- Availability – This ensures that the web services infrastructure is capable of sustaining operations after a failure. It provides that service interruptions will be properly addressed and avoided as best as possible.

1.7.3 Security Solution

To implement a security solution we must understand how the Web services consumer interacts with providers. For requesting a service, a Web Service client (Application Level) builds a SOAP stack (Message Level) and delivers it to a HTTP Client (Transport level). The HTTP client sends the packets to the HTTP Listener (Transport Level) on the Web Service-provided machine, the HTTP listener hands over the SOAP stack (Message Level) to the Web Service provider (Application Level) for processing (Florije). The procedure is reversed when the response, processed request is sent back to the client by the provider. The Security solutions for web services must be applied to Application, Message and transport level to ensure that all the security requirements are met for a secured Web service implementation. These three levels are detailed below.

- Application Level Security - A Web service application, which is at the highest level in the OSI model, handles its own security scheme. Traditional firewalls and other controls do not provide complete protection. Web services must use their own security domain and access control checks; all of the data fields must be encrypted using XML digital signatures.
- Message Level Security - applies to XML documents that the Web services sent as SOAP messages. The header in the SOAP message contains information relevant to the protocols and procedures for processing the specified message. With message level security, token flows are identified in the SOAP header, which allows the identify to flow through the entire length of the request - meaning the token sent by the consumer is the same token the provider receives. Encryption mechanisms like Security Assertion Markup Language (SAML) authentication, Single Sign-on (SSO), XML encryption, and XML digital signature (Takayuki, 222) must be used to bind the security token to the message contents.
- Transport Level Security – This leverages security configurations within the operating system and/or the application server. With transport level security, token flows are identified in the HTTP header, outside of the SOAP envelope. It is point-to-point, meaning the identity token sent with a web service request, through an intermediary, will not be able to flow to the service provider. The HTTP over SSL protocol must be used to protect the messages during transport between consumers and provider. To support non-repudiation, Web services must use authentication with client certificates.

A combination of all three levels, application, Message and Transport level, security mechanisms must be used for a successful Web services implementation.

1.8 Web Services Reliability

Web services have become the most popular means used by organizations to offer services to their customers and to interoperate with business partners. These services are accessed through messages. Since messaging is crucial to the business, in terms of the services and transactions that are exchanged between providers and subscribers, it has become essential to ensure reliable messaging. Reliability defines the ability of Web services to guarantee message delivery between provider and subscriber with an acceptable level of quality of service. Reliable service is an important measure of the quality of Web services. It ensures message persistence, acknowledgement, duplication elimination, orderly delivery, and provides appropriate status of delivery to both sender and receiver. Message-oriented middleware (asynchronous) technology, discussed in section 1.4, must be used for routing web services messages; this will guarantee that the messages sent will reach its destination even if the consumer or provider is not available.

1.9 Publicly Available Web Services

Web services are useful for exposing organizations internal systems for use by external entities. Large organizations like Google, Amazon, and UPS are publishing their Web Services for use by outside developers. It is a three step process for a developer to consume these public services: searching and finding the service using search tools, understanding the functionality of the services, and integrating the service into the internal application (Wang et al, 141). Some of the most popular public Web services are: (1) Google Inc., which published their search engine API for developers to consume it within their application for searching information and publishing the result inside the internal application, (2) UPS, USPS, and FedEx provide the API for tracking status of packages shipped, (3) Amazon provides an API to get prices of products available on their web site, and (4) PayPal provides an API for online stores to accept Payments.

Concluding Synthesis

In an age where information is a key to the success of businesses, both big and small, harnessing Internet communication in an efficient manner demands attention. Web services not only function on this principle, but they expand the reaches of internet-based business offerings. From initial conception of a Web service and its ideology, through the design, and on to security and implementation, it is clear that Web services are based on an intricate technological cohesion. For this reason, gaining a thorough understanding of Web services requires knowledge of a host of messaging protocols and communication semantics. This technology is a great asset when developed and secured properly.

In this examination of Web services as a whole, it appears there are two areas necessitating future attention. First, a clear, strict set of semantic protocols needs to be uniformly accepted and followed. This will aid in constructing a ubiquitous standard for Web service design. Second, security must be a pivotal concern for organization looking to make use of Web service technology. Just like any Web-based communication, data is highly susceptible to interception and/or manipulation, so security is crucial. Overall, Web services have almost unlimited applications, which many companies have already picked up on. Their popularity will likely persist into the foreseeable future.

Works Cited

Advantages & Disadvantages of Webservices. http://social.msdn.microsoft.com/forums/enUS/asmxandxml/thread/435f43a9-ee17-4700-8c9d-d9c3ba57b5ef/ Retrieved July 16, 2011.

Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M., Sheth, A., Verma, K. W3C Member Submission. (7 November 2005). *Web Service Semantics – WSDL-S*. Retrieved July 15, 2011, from http://www.w3.org/Submission/WSDL-S/

Bonella, Robert J. (2004), *Web Services and Web Services Security*, Communications of the Association for Information Systems (Volume 14). Retrieved July 17, 2011.

Charif, Yasmine, & Sabouret, Nicolas. (2006). *An Overview of Semantic Web Services Composition Approaches.* Electronic Notes in Theoretical Computer Science. Retrieved July 15, 2011, from www.sciencedirect.com

Crasso, Marco, & Zunino, Alejandro, & Campo, Marcelo. (2008). *Easy web service discovery: A query-by-example approach*. Buenos Aires, Argentina: ISISTAN Research Institute, UNICEN University, Campus Universitario, Tandil. Retrieved July 15, 2011.

Hao, Y., Zhang, Y., Cao, J. (2009). *Web services discovery and rank: An information retrieval approach*. Future Generation Computer Systems. Retrieved July 15, 2011, from www.sciencedirect.com

http://www.w3.org/TR/ws-arch/#technology. Retrieved July 16, 2011.

Ismaili, F., Sisediev, B. *Web Services Research Challenges, Limitations and Opportunities*. Retrieved from ACM Digital library on July 15, 2011.

Java. (2002). *Using Web Services Effectively.* Retrieved July 15, 2011, from http://java.sun.com/blueprints/webservices/using/webservbp.html

Khaldoon Al-Zoubi, Gabriel Wainer (2009). "Using REST Web-Services Architecture for Distributed Simulation". June 2009 PADS '09: Proceedings of the 2009 ACM/IEEE/SCS 23rd Workshop on Principles of Advanced and Distributed Simulation

Lior Fink, Seev Neumann (July 2009). "Taking the high road to web services implementation: an exploratory investigation of the organizational impacts" SIGMIS Database, Volume 40 Issue 3 Publisher: ACM

Maeda, T., Nomura, Y., Hara, H. (2003). *Security and Reliability for Web Services.* Retrieved from http://www.fujitsu.com/downloads/MAG/vol39-2/paper10.pdf on July 14, 2011.

McIlraith, Sheila A., Cao Son, Tran., Zeng, Honglei. (2001). *Semantic Web Services.* Standford University. Retrieved July 15, 2011

New to SOA and web services
http://www.ibm.com/developerworks/webservices/newto/service.html#HOWZC657. Retrieved July 16, 2011.

Qi Yu · Xumin Liu · Athman Bouguettaya ·Brahim Medjahed (2006). "Deploying and managing Web services: issues, solutions, and directions". Received: 12 August 2005 / Accepted: 19 April 2006 / Published online: 4 November 2006 © Springer-Verlag 2006

Rahaman, S., Uddin, A. *A Short Survey on Security Issues In Web Services*. Retrieved from http://web2.uwindsor.ca/courses/cs/aggarwal/cs60564/surveys/AbuShamual.doc on July 17, 2011.

Ratnasingam, P. (2004). "The Impact of Collaborative Commerce and Trust in Web Services". The Journal of Enterprise Information Management, Vol.17, No.5, pp. 382

Service-Oriented Architecture (SOA) and Web Services: The Road to Enterprise Application Integration (EAI). http://www.oracle.com/technetwork/articles/javase/soa-142870.html. Retrieved July 16, 2011.

Sirin, E., Hendler, J., Parsia, B.: Semi-automatic composition of web services using semantic descriptions. In: Web Services: Modeling, Architecture and Infrastructure Workshop in Conjunction with ICEIS2003 (2003)

Valacich, J. S., & Schneider, C. (2010). Information Systems Today: Managing in the Digital World (4th ed.). New Jersey: Prentice Hall Retrieved July 16, 2011.

W3C: SOAP. *Simple Object Access Protocol (SOAP)*. http://www.w3.org/TR/SOAP/ (2003)

W3C: UDDI. *Universal Description, Discovery, and Integration (UDDI)*. http://www.uddi.org (2003)

W3C: WSDL. *Web Services Description Language (WSDL)*. http://www.w3.org/TR/wsdl (2003)

W3C: WSA. *Web Services Architecture*. http://www.w3.org/TR/wsarch/ (2003)

Wang, L., Liu, F., Zhang, L., Li†, G., Xie, B. (2010). *Enriching Descriptions for Public Web Services using Information Captured from Related Web Pages on the Internet*. Service Oriented System Engineering. Retrieved July 17, 2011.

WS-Security Policy 1.3. Retrieved from http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/os/ws-securitypolicy-1.3-spec-os.pdf on July 17, 2011.