Designing Application Framework using WSDL

G M Tere
Thakur College of Science and Commerce,
Mumbai, Maharashtra - 400101, India
Mobile: 091-9920319945

B T Jadhav
Y.C. Institute of Science,
Satara, Maharashtra - 415001, India
Mobile: 091-9421215973

ABSTRACT
Because of loose coupling, interoperability and reusability in a service-oriented architecture (SOA) many business applications are developed using SOA. Main components os SOA based applications are service provider, service consumer and service repository. Communication between these components is achieeved by exchanging SOAP messages, which are XML documents. XML to Java mapping is difficult process. This process is time consuming. WSDL2WS is an application framework for developing Java web services. Using XML and WSDL we can develop a Java web service. Our application framework attemps to remove some drawbacks in developing web services using Java EE approaches. Web services interfaces, in the form of WSDL files, can be published to a business registry; these interfaces can then be dynamically looked up by clients. We compared performance of our framework with Apache Axis 1.0 and 1.2. for simple Web services.

General Terms
Performance, Design

Keywords
Application framework, Web services, WSDL, XML, WSDL2WS

1. INTRODUCTION
Application frameworks are set of guidelines and specifications that provide platforms, tools, and programming environments for addressing the design, integration, performance, security, and reliability of distributed and multi-tiered applications. An application framework includes the presentation services, server-side processing, session management, business logic framework, application data caching, application logic, persistence, transactions, security, and logging services for applications.

Application development tools and application servers are built on top of application frameworks. The aim of the application framework is to provide a single and unified software infrastructure that reduces the number of enterprise software products to support, maintain, and integrate. WSDL2WS is built on top of Jax-WS. Jax-WS is a toolkit, and WSDL2WS is an application framework on top of the toolkit, much like servlets is a toolkit, and Struts is an application framework that runs on top of the servlets. Consider example of a purchase order object. Using Jax-WS mappings to that, it will create a purchase order XML document, but it may not be the same structure as the purchase order document that business partners are using [4]. So more efforts are needed for mapping out one to the other. Using tools provided by WSDL2WS Java purchase order looks like the standard purchase order.

1.1 Introduction to SOA
Because robust Web Services technology is the foundation for implementing SOA[10], Java now provides the tools modern enterprises require to integrate their Java applications into SOA infrastructures. SOA is a distributed software model. The key components of SOA include services, dynamic discovery, and messages. Basic SOA architecture is shown in Figure 1.

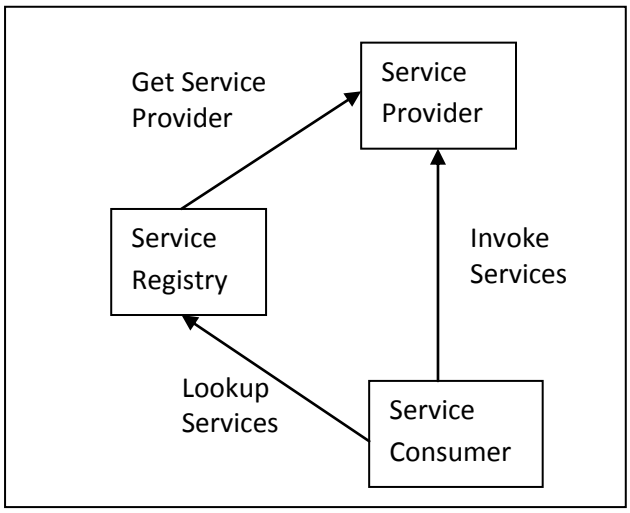

Fig 1: Basic SOA Architecture

1.2 Web services for SOA
Web services constitute a distributed computer architecture made up of many different computers trying to communicate over the network to form one system. They consist of a set of standards that allow developers to implement distributed applications - using radically different tools provided by many different vendors - to create applications that use a combination of software modules called from systems in disparate departments or from other companies. Web services are built on top of open standards and platform-independent protocols. A web service uses SOAP over HTTP for communication between service providers and consumers [2]. Services are exposed as interfaces defined by WSDL (Web Service Definition Language), whose semantics are defined in XML. UDDI, a language-independent protocol, is used for interacting with registries and looking for services. All of these features make web services an excellent choice for developing SOA applications [6][7][10]. Web services can be developed from WSDL file. This is called as Contract first approach.

1.3 WSDL
WSDL is an XML-based language for describing Web services and how to access them. It specifies the location of the service and the operations the service exposes. WSDL is a specification defining how to describe web services in a

common XML grammar. WSDL describes four critical pieces of data [13]:

1. Interface information describing all publicly available functions
2. Data type information for all message requests and message responses
3. Binding information about the transport protocol to be used
4. Address information for locating the specified service

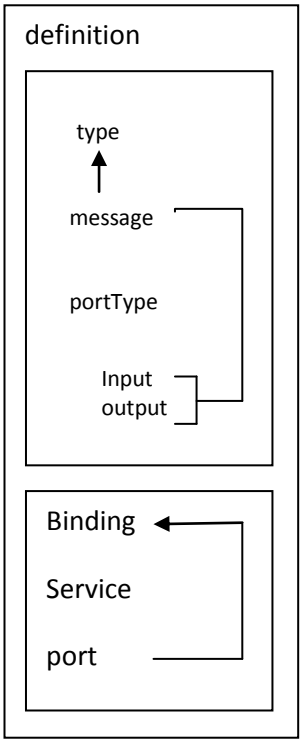

Fig 2: WSDL Document

Figure 2 show structure of typical WSDL document. WSDL describes a the Web service interface. It consists of messages that are exchanged between the client and server. The messages are described abstractly and then bound to a concrete network protocol and message format. Web service definitions can be mapped to any implementation language, platform, object model, or messaging system. WSDL represents a contract between the service requestor and the service provider, in much the same way that a Java interface represents a contract between client code and the actual Java object. The crucial difference is that WSDL is platform- and language-independent and is used primarily to describe SOAP services [9]. Using WSDL, a client can locate a web service and invoke any of its publicly available functions. With WSDL-aware tools, one can also automate this process, enabling applications to easily integrate new services with little or no manual code. WSDL therefore represents a cornerstone of the web service architecture, because it provides a common language for describing services and a platform for automatically integrating those services [16].

2. WSDL2WS ARCHITECTURE

This paper introduces the application framework, WSDL2WS, for SOA. This framework is WSDL-centric. Using WSDL2WS we can develop web services which can be used as components within SOA. WSDL2WS can be used to map existing Java applications into an SOA framework. Using Java Web Services (JWS) standards we designed a framework that facilitates WSDL-centric construction of Web Services. Using WSDL2WS, a Web service can be created by building its WSDL and annotating that WSDL document with references to the Java elements that implement it. WSDL-centric approach is perfect for situations where thesre is need to create Web services that integrate into a standard corporate or eBusiness framework [15][16]. We designed this application framework to support service orientation for Java applications.

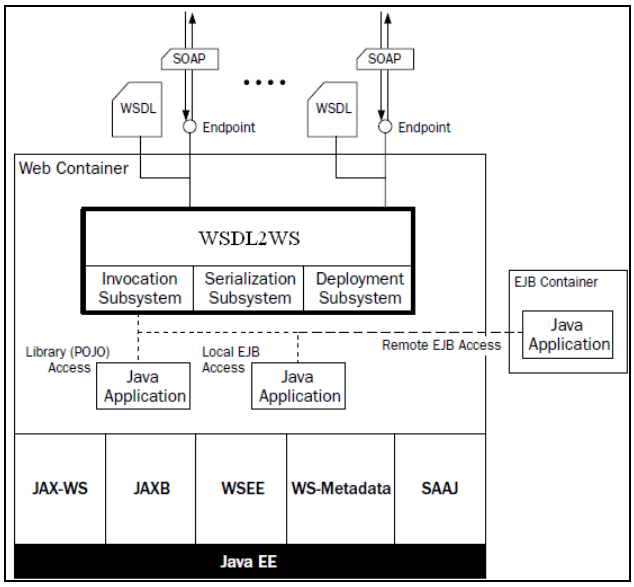

Fig 3: Server side architecture of WSDL2WS

WSDL2WS is designed to run inside a Java EE container and to leverage the Web Services infrastructure provided by JAX-WS, JAXB, WS-Metadata, and WSEE. WSEE defines the programming model and run-time behavior of Web Services in the Java EE container.Unlike Apache Axis [1] and other standalone Web Services server solutions, WSDL2WS does not replace the internal Web Services capabilities of the Java application server, but enhances them. Figure 3 explain basic WSDL2WS architecture. WSDL2WS is deployed as a Web module (WAR). WSDL2WS is packaged as a port component implementing the JAX-WS Provider interface [4] and therefore can be deployed as a servlet endpoint. In this manner WSDL2WS is built on top of the JAX-WS implementation provided by the Java EE container [6]. WSDL2WS also takes advantage of WSEE and WS-Metadata for packaging and deployment, and JAXB for serialization. Serialization is the process of transforming an instance of a Java class into an XML element. The inverse process, transforming an XML element into an instance of a Java class, is called deserialization. Because it is built on these portable technologies, WSDL2WS can be deployed and run on any Java application server that supports Java EE .

As shown in Figure 3, WSDL2WS binds Web services to endpoints. Each endpoint published by WSDL2WS has an associated URL where SOAP requests can be posted. A

WSDL document is also associated with an endpoint to describe the Web services available at that URL. Endpoints are dynamically configured and can be created and removed at runtime. There is no need to undeploy the WSDL2WS module, or restart the Java EE server, in order to create, change, or delete an endpoint. WSDL2WS can receive SOAP requests at any of the endpoints that have been configured by the user [9]. When WSDL2WS receives a SOAP request at a configured endpoint, it invokes all the components of the Web Services Platform Architecture to process that request: the deployment unit, the invocation unit, and the serialization unit. The deployment unit determines which Java class and method to invoke—based on the structure of the SOAP request and how the endpoint has been configured.

The invocation unit invokes the proper class and method, using the serialization unit to translate between XML and Java objects. Let us understand how WSDL2WS is configured to publish its endpoints. Figure 4 shows the configuration process.

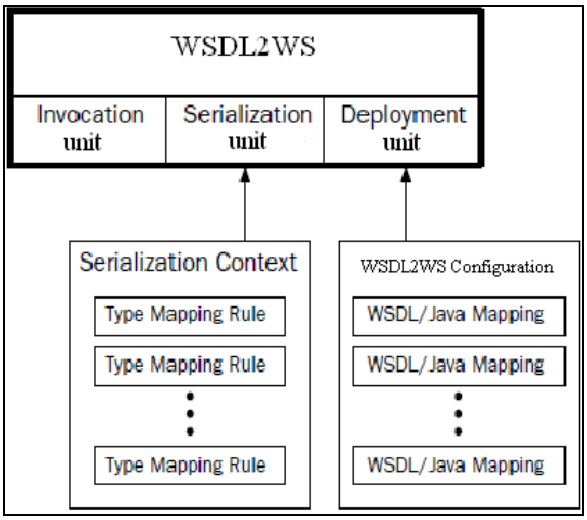

Fig 4: WSDL2WS configuration

An endpoint is configured using a WSDL/Java mapping. The set of WSDL/Java mappings that is used to configure WSDL2WS is called the WSDL2WS Configuration. Each WSDL/Java mapping defines pairings of Java class/methods with wsdl:operation instances from a particular wsdl:port defined in a WSDL document. The Java class/method associated with a wsdl:operation gets invoked when a SOAP message is received that references that wsdl:operation. Configuration of WSDL2WS also requires a Serialization Context. The Serialization Context is a set of type mapping rules. Each type mapping rule tells WSDL2WS how to either (i) serialize instances of a particular Java class to instances of a particular XML type; or (ii) deserialize instances of a particular XML type into instances of a particular Java class. The serialization unit uses this Serialization Context to deserialize SOAP parameters into Java parameters, and to serialize the Java return type instance into a SOAP parameter [9]. The type mapping rules are written in a declarative XML language defined by the Adaptive Serialization Framework. The WSDL2WS invocation unit manages the sequence of events from receiving the SOAP request, to dispatching it to the correct Java class/method, to invoking the serialization unit. Next sections describe in detail how WSDL2WS is implemented.

3. WSDL-CENTRIC DEVELOPMENT

As discussed in Section 2, WSDL2WS is configured using an WSDL2WS Configuration file. We construct a Web service by creating its WSDL via the WSDL2WS Configuration. The structure of the WSDL2WS Configuration file is same as the WSDL structure.

The core components of an WSDL2WS Configuration file are the W2WOperation elements. Such operation elements define a mapping from a Java class/method to a wsdl:port and wsdl:operation. The wsdl:operation is specified by the operationName attribute. The wsdl:port is defined by the surrounding W2WPort element [11][14]. The W2WOperation elements are grouped under W2WPort elements to define a wsdl:port. Hence, WSDL2WS can create a wsdl:port using a variety of Java classes and methods. This is more flexible than the JWS model, where all the operations on a single wsdl:port must be implemented by methods from the same Java class.

4. Invocation Unit

The invocation unit is responsible for receiving SOAP messages. As implemented in WSDL2WS, the invocation unit is responsible for:

1. Receiving a SOAP message

2. Determining the message's target service—i.e., which WSDL operation is the message intended to invoke Given the target WSDL operation, dispatching the message to the correct Java class/method to invoke

4. Handing off the SOAP message to the Serialization unit to deserialize it into Java objects that can be passed to the Java target as parameters

5. Invoking the Java target using the parameters generated by the Serialization unit and getting the Java object returned by the target method

6. Handing off the returned object to the Serialization unit to serialize it into an XML element conformant with the return message specified by the target WSDL operation

7. Wrapping the returned XML element as a SOAP message response conforming to the target WSDL operation and, if an exception has occurred, mapping it to a SOAP Fault that will be the response

8. Sending the SOAP response

Figure 5 provides a static class UML diagram showing the highlevel implementation of the WSDL2WS invocation unit. SOAP requests are first received by the W2WDispatcher. The W2WDispatcher is an HttpServlet that simply provides a base context root for all WSDL2WS endpoints managed within a single WSDL2WS configuration. For each SOAP request received, the W2WProvider instantiates a RequestController—the class that handles WSDL2WS request processing. The RequestController contains an InputDataProcessor instance for processing the SOAP request and creating an instance of WSRequest—the WSDL2WS internal representation of a SOAP request message and its context. In addition, the RequestController contains a FlowInterpreter instance, which is responsible for instantiating the appropriate WSDL2WS operation (i.e., dispatching), invoking it, and creating a WSResponse instance to contain the results.

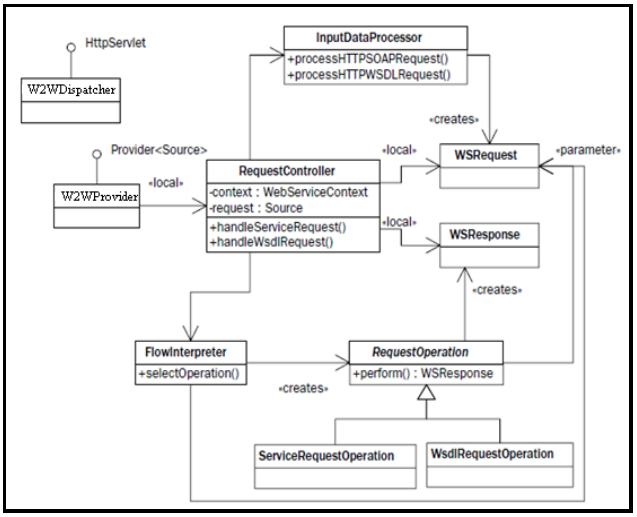

Fig 5: Invocation process used in WSDL2WS

WSResponse is the WSDL2WS internal representation of a SOAP response message. As shown in Figure 5, the RequestOperation (an abstract class) has two subclasses: ServiceRequest-Operation and WsdlRequestOperation. That is because this invocation unit handles WSDL requests and SOAP requests. WSDL2WS follows the convention that HTTP GET requests posted to the endpoint suffixed with "?wsdl" result in HTTP responses containing the target WSDL. Figure 6 shows the high-level structure of the WSRequest and WSResponseclasses. Focusing on WSRequest first, notice that it has two subclasses:

WsdlRequest for encapsulating a request for the WSDL document, and ServiceRequest for encapsulating a SOAP request posted to a Web service endpoint. In addition, the WSRequest contains a reference to the W2WConfiguration. The ServiceRequest contains the XML parameters—the children of the wrapper element in the SOAP request body. These will be deserialized to become the Java parameters used to invoke the method that implements the target service. In addition, the ServiceRequest class contains the operationName (the wrapper element's name) and the endpoint (from the HTTP request headers) in order to dispatch the request.

Similarly, the WSResponse class has two subclasses: WsdlResponse and ServiceResponse. In this case, the ServiceResponse has, at most, one XmlParameter—the serialized instance of the return type produced by the target Java class/method. The WsdlResponse contains the WSDL2WS-generated WSDL.

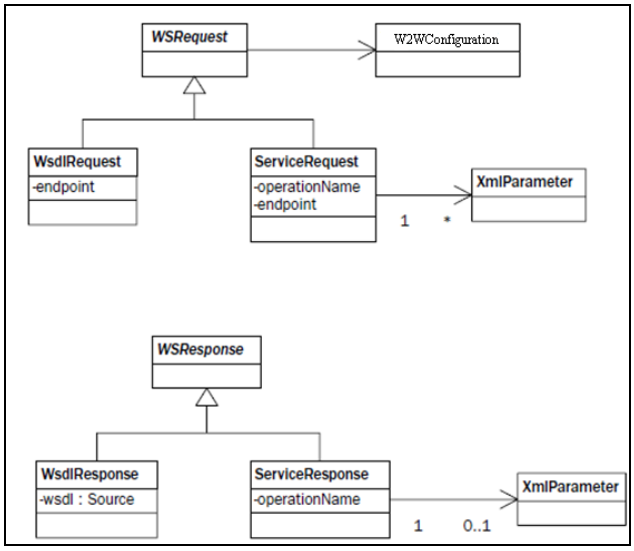

Fig. 6: The WSRequest/WSResponse structures

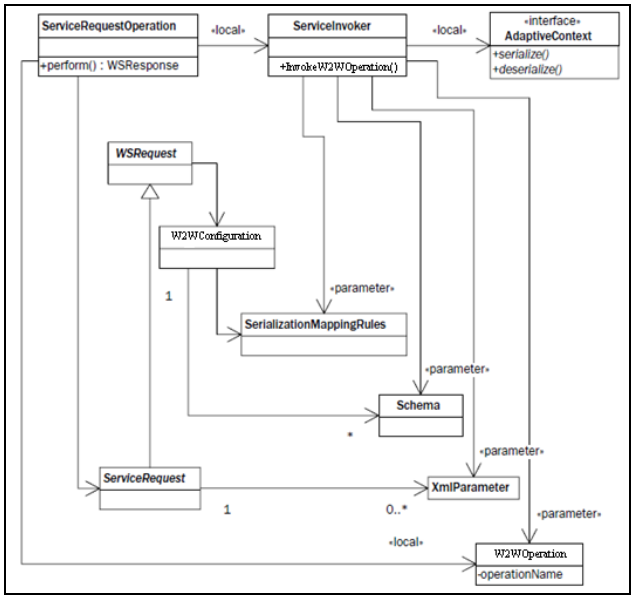

Fig 7: ServiceRequestOperation creates a ServiceInvoker that invokes the W2WOperation.

Figure 7 shows the class structure, related to ServiceRequestOperation, that handles the processing of the ServiceRequest (WSRequest) to invoke a Web service. As shown in Figure 7, the perform() method implements invocation using locally created instances of W2WOperation and ServiceInvoker. W2WOperation is created from the WSDL2WS configuration file information accessed through the ServiceRequest. It is an encapsulation of the Java class/method to be invoked.

5. SERIALIZATION UNIT

It is very difficult to deploy existing Java classes against an existing schema. With JAXB 2.0 and JAX-WS 2.0 [4], We basically get WSDL based on the schema JAXB 2.0 generates from classes and annotations. Annotations are not a good way to implement such type mappings because of their following limitations:

1. To change a type mapping, one need to edit the source, recompile, and redeploy.

2. Annotations are a very unintuitive way to create type mappings. We need to keep regenerating the resulting schema (via JAXB), and try to see whether we can "come close" to the target schema. This approach reduces performance.

3. Many type mappings are impossible to generate with JAXB annotations.

4. Good design practice show that a mapping layer should insulate the XML (WSDL) representation of Web services from the Java implementation. JAXB annotations, although convenient, destroy that mapping layer inside the Java source code. This makes it difficult to understand and maintain.

6. COMPARISON OF WSDL2WS WITH OTHER FRAMEWORKS

We have tested the performance of WSDL2WS application framework with other such as Apache Axis 1.1 and Apache Axis 2.0 and Eclipse IDE. Our experiment show that using WSDL2WS application framework which is based on Contract first give better performance than the mentioned framworks.

Table 1. Comparison of different Application Frameworks

	Time required to get response from Web service in msec		
Web Service	**Apache Axis 1 with Eclipse**	**Apache Axis 2 with Eclipse**	**Using WSDL2WS Application Framework**
Hello World	250	346	156
Adder	367	487	202

Experiment were performed using Dell Inspiron Laptop with Intel Core 2 Duo CPU, Physical Memory 4 GB, Processor Speed 2 GHz with Windows XP OS. We used JWS, J2EE, Eclipse, Apache Axis and Tomcat 6 server and GlassFish. We developed Hello World and Adder Web service which add two integers. Results are shown in Table 1 and graphically in Figure. 8.

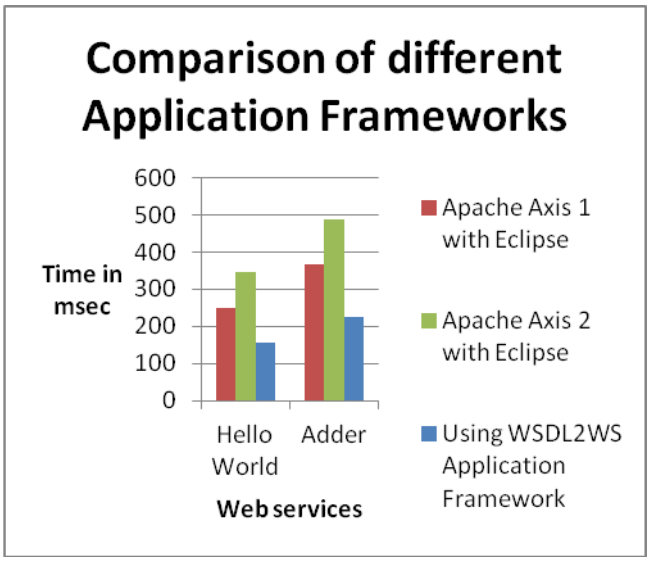

Fig 8: Comparison of different Application frameworks

7. CONCLUSIONS

We developed the application framework WSDL2WS which is used for developing Web services. Using Java Web Services technology the application framework WSDL2WS was developed. Web services are used for developing SOA based application. Different contemporary SOA characteristics are observed in the process of developing Web services. Using WSDL2WS framework, leveraging the power of JWS, we can efficiently develope Java applications in a contemporary SOA. We have tested the performance of WSDL2WS application framework with other such as Apache Axis 1.1 and Apache Axis 2.0. Our experiment show that using WSDL2WS application framework which is based on Contract first give better performance than the mentioned framworks.

8. FUTURE WORK

Currently, WSDL2WS handles only SOAP over HTTP. We need to extend it to handle other transport protocols, such as JMS. The framework need to be tested for bigger Web services.

9. ACKNOWLEDGMENTS

We wish to thanks to teachers of Department of Computer Science, Shivaji University, Kolhapur and Principal of Thakur College of Science and Commerce, Mumbai for motivating us for this research work.

10. REFERENCES

[1] Apache Axis 2.x., http://ws.apache.org/axis2

[2] Ben Shil, A.; Ben Ahmed, M.; Additional Functionalities to SOAP, WSDL and UDDI for a Better Web Services' Administration, 2nd International Conference on Information and Communication Technologies, 2006. ICTTA '06. Volume : 1, pp: 572 - 577

[3] Crockford D. "The application/json Media Type for JavaScript Object Notation (JSON)." The Internet Engineering Task Force (Network Working Group) RFC-4627, July 2006, http://tools.ietf.org/html/rfc4627

[4] Eckstein, and Robert Rajiv Mordani. "Introducing JAX-WS 2.0 with the Java SE 6 Platform, Part 2," November 2006.http://java.sun.com/developer/technicalArticles/J2SE/jax_ws_2_pt2/Monson-Haefel, Richard. J2EE Web Services. Addison-Wesley Professional, ISBN 0130655678, October 2003.

[5] Haidar, A. N.; Abdallah, A. E.; Abstractions of Web Services, 14th IEEE International Conference on Engineering of Complex Computer Systems, 2009, pp: 182 - 191

[6] Jen-Yao Chung; An industry view on service-oriented architecture and Web services, IEEE International Workshop on Service-Oriented System Engineering, 2005. SOSE 2005. pp:59

[7] Sandy Carter, "The New Language of Business: SOA & Web 2.0", IBM Press, 2007

[8] Siblini, R.; Mansour, N.; Testing Web services, The 3rd ACS/IEEE International Conference on Computer Systems and Applications, 2005. pp: 135

[9] SOAP Version 1.2 Part 0: Primer. W3C Recommendation, June 24 2003, www.w3.org/TR/soap12-part0

[10] Thomas Erl, "Service-Oriented Architecture: Concepts, Technology, and Design", Pearson Education, Inc., 2007.

[11] Tsai, W.T.; Paul, R.; Yamin Wang; Chun Fan; Dong Wang; Extending WSDL to facilitate Web services testing, Proceedings of 7th IEEE International Symposium on High Assurance Systems Engineering, 2002, pp: 171 - 172

[12] Walmsley, Priscilla. Definitive XML Schema. Prentice-Hall PTR, ISBN 0321146182, December 2001.

[13] Web Services Description Language (WSDL) 1.1. W3C Note, March 15, 2001, www.w3.org/TR/wsdl

[14] Web Services Addressing 1.0—Core. W3C Recommendation, May 9, 2006, www.w3.org/TR/ws-addr-core/

[15] Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. W3C Working Draft, August 3, 2005. www.w3.org/TR/wsdl20/

[16] Web Services Description Language (WSDL) Version 2.0 Part 2: Adjuncts. W3C Working Draft, August 3, 2005. www.w3.org/TR/wsdl20-adjuncts