Designing Web Services using Web Service Specifications

G. M. Tere[1], R. R. Mudholkar[2], B. T. Jadhav[3]
[1] Department of Computer Science,
Shivaji University, Kolhapur – 416 004, India
girish.tere@gmail.com
[2] Department of Electronics,
Shivaji University, Kolhapur – 416 004, India
rrm_eln@unishivaji.ac.in
[3] Department of Electronics and Computer Science,
Y.C. Institute of Science, Satara – 415 001, India
btj21875@gmail.com

ABSTRACT
A Web service is a programmatic interface for application-to-application communication that is invoked by sending and receiving XML. Web services are often developed using Web Service Description Language (WSDL) interface. WSDL is contract between service provider and service clients. WSDL files are basically XML files and because of it more time is required to process WSDL files in Java environment. We need to perform serialization and deserialization in efficient way. We propose alternative to WSDL, called as Web Service Specification (WSS) using which a performance of web services can be improved. Our experiments show that we have reduced the execution time of web services between 70% to 80% of execution time required for web services with WSDL.

KEYWORDS
SOAP, Web service, WSDL, WSS, XML

1. INTRODUCTION

With Web Services, information sources become components that you can use, re-use, mix, and match to enhance Internet and intranet applications ranging from a simple currency converter, stock quotes, or dictionary to an integrated, portal based travel planner, procurement workflow system, or consolidated purchase processes across multiple sites. Each is built upon an architecture that is presented in this paper as an illustrated stack of layers, or a narrative format. Each vendor, standards organization, or marketing research firm defines Web Services in a slightly different way. Gartner [7], for instance, defines Web Services as "loosely coupled software components that interact with one another dynamically via standard Internet technologies." Forrester Research [17] takes a more open approach to Web Services as "automated connections between people, systems and applications that expose elements of business functionality as a software service and create new business value." Although we have a variety of Web Services architectures, Web Services, at a basic level, can be considered a universal client/server architecture that allows disparate systems to communicate with each other without using proprietary client libraries, according to the WebMethods whitepaper, Implementing Enterprise WebServices with the WebMethods Integration Platform [17]. The whitepaper points out that "this [architecture] simplifies the development process typically associated with client/server applications by effectively eliminating code dependencies between client and server" and "the server interface information is disclosed to the client via a configuration file encoded in a standard format (e.g. WSDL)." Doing so allows the server to publish a single file for all target client platforms. 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. WSDL [2] is an interface definition language, written in XML, used for describing Web services. The term "WSDL-centric" means creating a Web service by building its WSDL and using that WSDL document with references to the Java elements that implement it.

JWS has weaknesses, particularly when it comes to the development approach known as Start from WSDL and Java [3]. The JWS standards present a Java-centric approach to Web Services. This approach can be troublesome when we need to work with established SOA standards and map Java application to existing XML Schema documents and WSDLs. In such situations, it's helpful to be able to take a WSDL-centric approach to Web Services development. In this area, JWS is less strong [20].

2. USE OF REFLECTION

Reflection [8] is commonly used by programs which require the ability to examine or modify the runtime behavior of applications running in the Java virtual machines. Reflection is a powerful technique and that can enable applications to perform applications which would otherwise be impossible [10]. Reflection is the process by which a computer program can observe (do type introspection) and modify its own structure and behavior at runtime [10]. In many computer

architectures, program instructions are stored as data—hence the distinction between instruction and data is merely a matter of how the information is treated by the computer and programming language. Normally, instructions are executed and data is processed; however, in some languages, programs can also treat instructions as data and therefore make reflective modifications. Reflection can be used for observing and/or modifying program execution at runtime. A reflection-oriented program component can monitor the execution of an enclosure of code and can modify itself according to a desired goal related to that enclosure. This is typically accomplished by dynamically assigning program code at runtime.

In object oriented programming languages such as Java, reflection allows inspection of classes, interfaces, fields and methods at runtime without knowing the names of the interfaces, fields, methods at compile time. It also allows instantiation of new objects and invocation of methods. Write the body of the paper here.

3. DESIGN OF FRAMEWORK

Web Service Framework

We designed Web Services framework [4,5] which is based on open industry standards, defining four requirements: description, discovery, request/response, and transport. Fig. 1 shows how various parts of the Web Services framework are related to one another. Starting in the upper-left, Consumer Applications send XML Service Requests to the Web Services Client Library, using SOAP and ICE. The Client Library provides Java and PL/SQL interfaces to the Web Services Broker. Interacting with the Broker for Web Services and Database Services is accomplished though SOAP, Java reflection, or JDBC via software components called adapters. When the Broker returns results to the Services, it dispatches them to the Consumer applications for display to end-users. Software components called transformers allow the framework to support several output formats, including HTML pages, and pages formatted for wireless and mobile devices.

The Web Services framework insulates developers from the complexity of interacting with multiple information sources, protocols, and delivery channels.

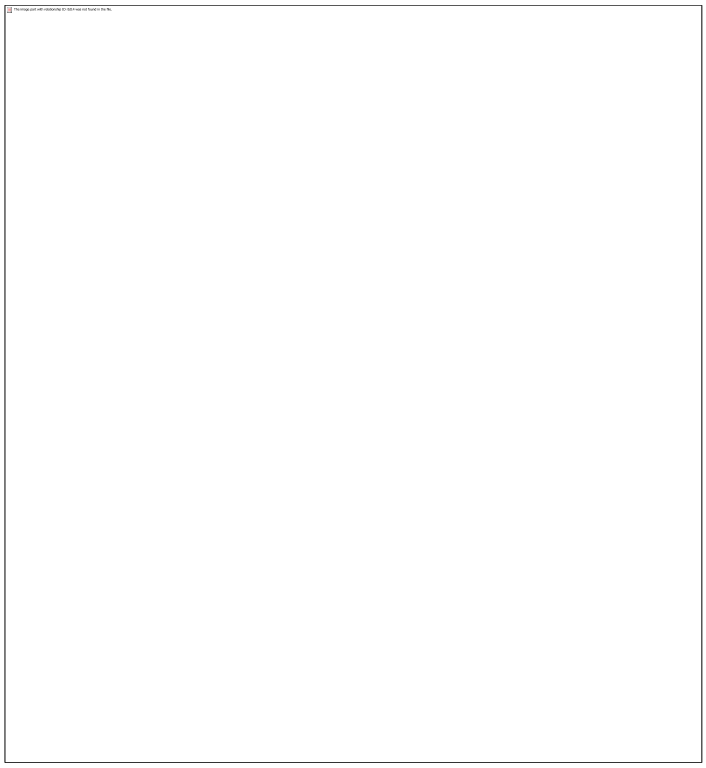

Figure 1. Web Services Framework

It is component-based to maximize re-use. It includes tools for creating, managing, and monitoring services. Fig. 2 gives a developer's view of how various parts of the framework interact.

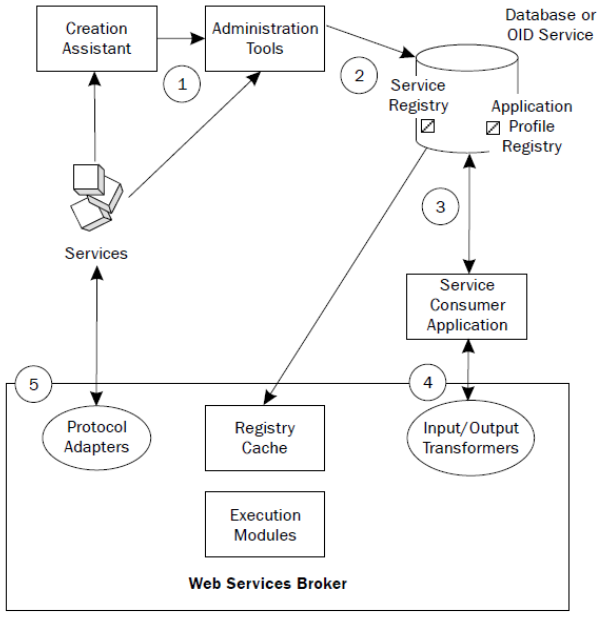

Figure 2. Interactions within Web Services Framework

Working clockwise from the left, the steps are:

1) A Service Provider can start by reusing an existing Web or database application - ideally, one that returns results in XML. If not, Web Services includes utilities that map HTML and other data sources to XML. Web Services also includes a Creation Assistant that generates a simple service from a Web page.
2) Next, a Service Administrator uses a tool of choice to register the service, making it available to consumers.
3) Service Consumer Applications query the Service Registry to get the data required to find and invoke a service. Data about Service Consumer Applications, including access privileges, is stored and maintained in the Application Profile Registry.
4) Then, the Service Consumer Application interacts with the Web Services Broker, which uses an input transformer, if needed, to convert the Consumer's request to a format it can use internally. When the service returns a result, the Web Services Broker applies an output transformer, if needed, and dispatches the data to the Service Consumer application. The Service Consumer Application can display the data to end-users, or use it in the flow of some business logic.
5) The Web Services Broker invokes the service via an adapter appropriate for the service's protocol (HTTP, SMTP, etc.).

4. WEB SERVICE SPECIFICATIONS (WSS)

We need WSDL to connect to a Web Service. This language is an XML format for describing network services. With it, service requesters can search for and find the information on services via UDDI, which, in turn, returns the WSDL reference that can be used to bind to the service. WSDL is a specification [20] defining how to describe web services in a common XML grammar. WSDL describes four critical pieces of data:

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

Using WSDL, a client can locate a web service and invoke any of its publicly available functions. With WSDL-aware tools, you 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. Fig. 3 shows a concise representation of the WSDL specification.

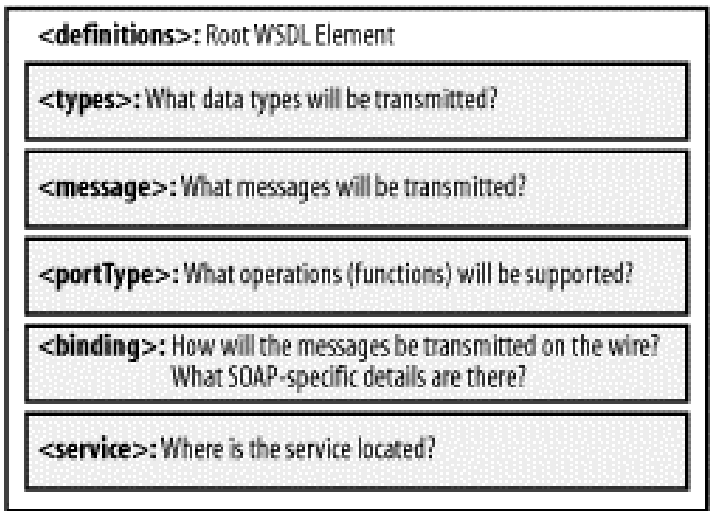

Figure 3. The WSDL specifications [20]

Web Service Conversational Language (WSCL) helps developers use the XML Schema to better describe the structure of data in a common format (say, with new data types) the customers, Web browsers, or indeed any XML enabled software programs can recognize. This protocol can be used to specify a Web Service interface and to describe service interactions. Given the WSDL file one can manually create a SOAP client to invoke the service. We can also automatically invoke the service via a WSDL invocation tool as shown in Fig. 4.

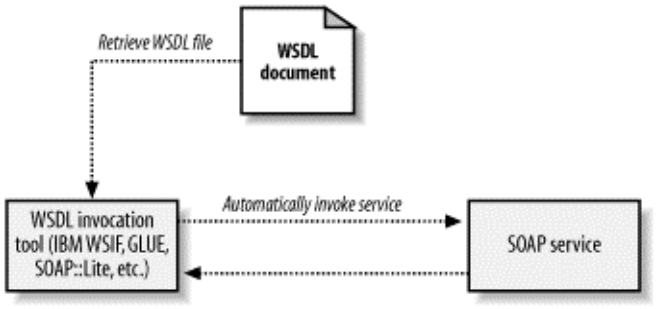

Figure 4. WSDL Invocation Tool

WSDL documents are essentially XML documents and are very lengthy. Therefore more computer resources are used to parse and then process the WSDL documents. To improve performance we need some alternative to WSDL. In this paper, we propose Web Service Specifications (WSS) as alternative to WSDL. All necessary information to invoke the web service will be available in WSS files. We have developed application using Eclipse and J2EE. Screen shot of the application is shown in Fig. 5.

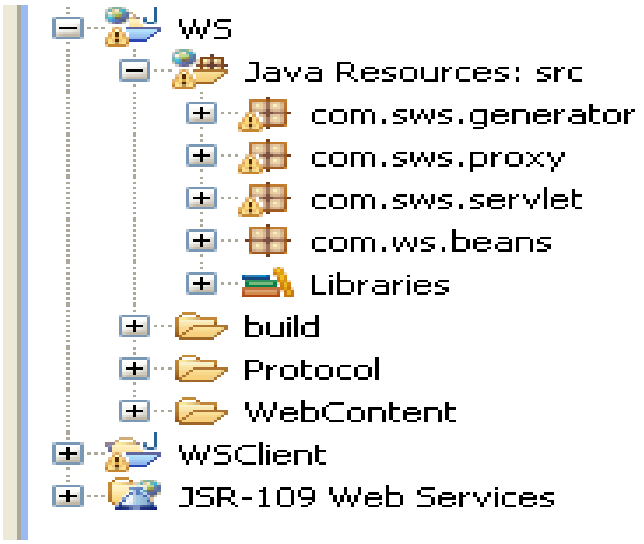

Figure 5. Application to develop Web Services

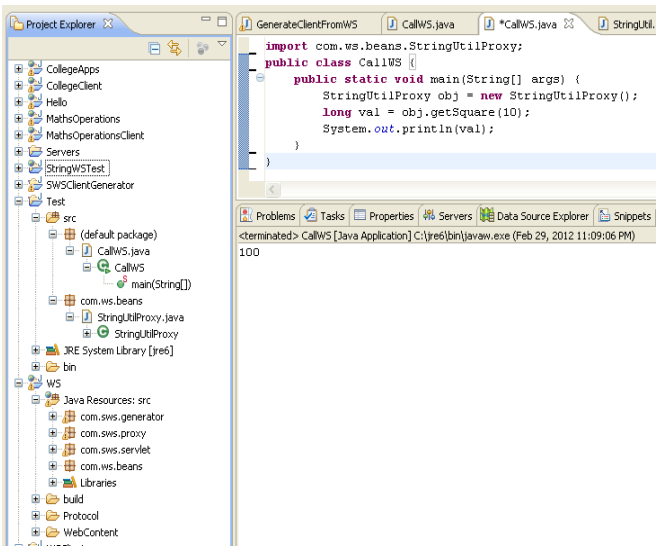

Figure 6. Client for calling Web Service

WSS file is similar to WSDL of WS, it has all the information required for creating web service. From the specification files our framework can generate :

- Specification documentation.
- The specification documentation is a set of static HTML pages containing the description of specification. It lists all APIs, functions, types and error codes. The pages also provide an easy way to call your function using the web application.
- The web application - The web application is packaged as a WAR file that you can deploy in any servlet container. Once the WAR file is deployed, you can access your API through HTTP.
- The client-side API - The client API is a JAR file that can be used to invoke remotely the API functions from Java programs.
- It features various advanced features, such as load-balancing, fail-over, extensive logging, etc...
- The Javadoc.
- The Javadoc contains the definition of the API for the Java classes, including the generated classes. The
- javadoc can be generated either for the server side classes or the client side classes.

5. DESIGNING WEB SERVICES USING WSS

In Fig. 7 we give the sample HelloWS.wss file. This wss file contain the information to call HelloWS. The corresponding HelloWS.wsdl file will be very large. It is not shown here, but is available on internet.

```
#Name
WSNAME %% com.ws.beans.HelloWS
#URL
WSURL %% http://localhost:8080/WS/HelloWS.sws
#Unique ID of WS
WSID %% 27763265-177f-4770-bb22-16fcd6e9d105
RFN %% sayHello
RTYPE %% String
PCNT %% 1
PARAM  %% String
```

Figure 7. Sample WSS file, HelloWS.wss

A client application can be a servlet. As shown in Fig. 8, we wrote a servlet, which calls a Web service, which returns a string in capital. Corresponding .wss, StringUtil.wss, file generated is shown in Fig. 9. Output of Web service is shown in Fig. 10.

```
protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
  StringUtilProxy proxy = new  StringUtilProxy();
  String result = proxy.toCapital("web services are very
       useful.");
   response.getWriter().println(result);
}
```

Figure 8. Servlet calling to a Web Service

```
#Name
WSNAME %% com.ws.beans.StringUtil
#URL
WSURL %% http://localhost:8080/WS/StringUtil.sws
```

```
#Unique ID of WS
WSID %% 622f024c-12af-4c44-9fb2-d5ea67df36ce
RFN %% toCapital
RTYPE %% String
PCNT %% 1
PARAM  %% String
RFN %% getSquare
RTYPE %% long
PCNT %% 1
PARAM  %% int
```

Figure 9. StringUtil.wss

Figure 10. Output of Web Service

6. RESULTS OBTAINED

We run the web services on DELL INSPIRON CORE 2 Duo CPU @ 2 GHz with 4 GB RAM. We measured Round Trip Time (RTT) in msec of different web services using following Java code fragment:

```
long start = System.currentTimeMillis();
<...call to web server ...>
System.out.println("Time taken: "+
        (System.currentTimeMillis() - start)) + "ms");
```

We run the HelloWorld and String Utilities web services five times using WSDL and WSS approach and calculated the average time needed to complete round trip. These values are shown in Table 1 and graphically plotted in Fig. 11.

Table 1. Comparison of different approaches

	Round Trip Time of different web services in msec (Average)	
Web Service	**Using WSDL**	**Using WSS**
Hello World	45	35
String Utilities	128	96

7. CONCLUSION

In this paper we have proposed alternative to WSDL. WSDL are basically large XML files. Therefore to process WSDL files significant time is needed. We developed our own Web Service Specification, WSS, approach developed using Java reflection. Using our approach we can develop and call web services fast as compared to the traditional approach. We have reduced the time to three fourth of the traditional approach.

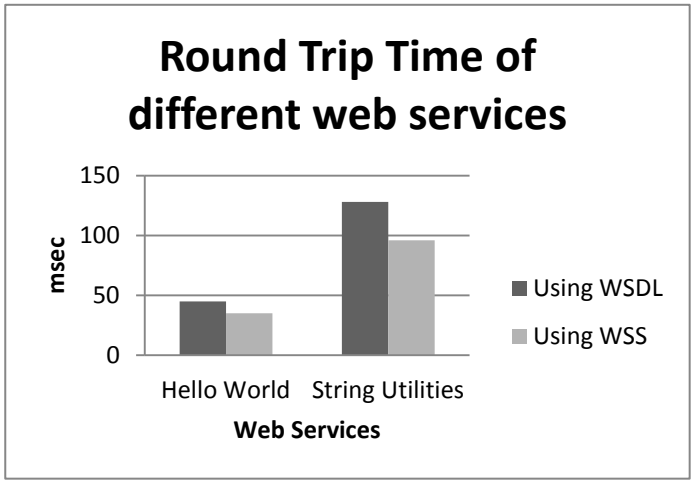

Fig. 11 RTT for different frameworks

FUTURE SCOPE

We need to examine our approach with web services which return records from a database like MS Access, Oracle or SQL Server.

REFERENCES

[1] Apache CXF, http://cxf.apache.org/ , Accessed on 10th Feb 2012

[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] Bobby Bissett, Building JAX-WS 2.0 Services with NetBeans 5.0 and GlassFish, http://jax-ws.java.net/articles/jaxws-netbeans/glassfish.html, Accessed on 10th Feb 2012

[4] Cao Hong-Hua; Ying Shi; Cui Hua; Xiao Yang; , "Towards a Framework for Designing, Deploying and Executing Semantic Web Service-Based Process," Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on , vol., no., pp.1-4, 12-14 Oct. 2008

[5] Chaoying Ma; Bacon, L.; Petridis, M.; Windall, G.; , "Towards the Design of a Portal Framework for Web Services Integration," Telecommunications, 2006. AICT-ICIW '06. International Conference on Internet and Web Applications and Services/Advanced International Conference on , vol., no., pp. 163, 19-25 Feb. 2006

[6] 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.

[7] Gartner, http://www.gartner.com/technology/core/products/research/topics/webServices.jsp, Accessed on 12th Feb 2012

[8] Gordon S. Blair, Geoff Coulson, Lynne Blair, Reflection, Self-Awareness and Self-Healing in OpenORB, WOSS '02, Nov 18-19, 2002, Charleston, SC, USA. Copyright 2002 ACM

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

[10] Ira R. Forman and Nate Forman, Java Reflection in Action (2005), ISBN 1-932394-18-4

[11] 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

[12] Kumar, A.; , "Distributed system development using Web service and Enterprise Java Beans," Services Computing, 2005 IEEE International Conference on , vol.2, no., pp. xiii vol.2, 11-15 July 2005

[13] Li Zhang; , "Requirement engineering for Web applications," Web Site Evolution, 2008. WSE 2008. 10th International Symposium on , vol., no., pp.1, 3-4 Oct. 2008

[14] Preeda Rajasekaran, John Miller, Kunal Verma, and Amit Sheth, Enhancing Web Services Description and Discovery to Facilitate Composition, SWSWPC 2004, LNCS 3387, pp. 55 – 68, 2005. © Springer-Verlag Berlin Heidelberg 2005

[15] Rama Pulavarthi, Monitoring SOAP Messages Made Easy With JAX-WS RI 2.0.1, http://weblogs.java.net/blog/ramapulavarthi/archive/2006/08/monitoring_soap.html, Accessed on 10th Feb 2012

[16] Rama Pulavarthi, Useful Goodies for Web Service Developers in JAX-WS 2.1 RI, http://weblogs.java.net/blog/ramapulavarthi/archive/2007/02/useful_goodies.html, Accessed on 10th Feb 2012

[17] Ted Schadler, Web Services: The Next Technology Thunderstorm, Forrester Research, Mar 2002

[18] 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

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

[20] Web Services Description Language (WSDL) Version 2.0 Part 1 and Part 2: Core Language. W3C Working Draft, August 3, 2005. www.w3.org/TR/wsdl20/