The Design and Implementation of Unified Invoking Component Based on Web Services Framework

Wenpeng Su[a], Zhonghua Yan[a,b*], Chenghui Liang[a]
[a] School of Mechanical, Electrical and Information Engineering, Shandong University, Weihai 264209, China
Email: conanswp@163.com, liangchenghui@sdu.edu.cn
[b] Integrated Electronic Systems Lab Co. Ltd., Jinan 250100, China
Email: yanzhonghua@ieslab.cn

***Abstract*— Web Services is a platform which enables the applications interoperate on the Internet. It is widely used in designing and building systems in open and dynamic distributed environments such as EAI (Enterprise Application Integration) and B2B (Business to Business). As the development of framework technology, it is convenient and standardized to use framework to develop web applications. For Web Services, the frameworks Axis, Axis2, XFire and CXF are widely used. By the performance testing of the four frameworks, this paper not only introduces the four framework but also analyzes the differences of the four frameworks, and then makes some suggestions for developers to choose the appropriate one. The framework can simplify the development process and decrease the development time. However, because of the differences of the frameworks, the interoperation between different web service's server and client may cause incompatible problems. This paper analyzes the reasons of this incompatible problems and finally presents a mechanism based on unified invoking component to solve this problems. By parsing and repacking the SOAP messages, the incompatible problems between client and server can be solved successfully.**

***Index Terms*— Web Services, WSDL, SOA, performance testing, unified invoking component, framework**

I. Introduction

As the information technology develops constantly and the capability of the Internet improves gradually, more and more application systems are established on Internet. Web Services is a software component independent from platform and realization, which enables the applications interoperate on the Internet and publish, discover and invoke services through Web. Web Services is an implementation of SOA (Service Oriented Architecture) and can compose different units of application programs through neutral interfaces in order to realize loose coupling between different applications [1]. By using the standard protocols such as XML, SOAP, UDDI, and WSDL, Web Services has good encapsulation and strong integration capabilities and is widely used in the area of EAI (Enterprise Application Integration) and B2B (Business to Business).

In order to simplify and standardize the development process, frameworks are widely used in all field of software. For example, it's very convenient to use Hibernate and Spring for web application developers. In the field of Web Services, there are four famous frameworks called Axis, Axis2, XFire and CXF. All of the frameworks encapsulate the low-level information and provide powerful development APIs for developers, which can really reduce the development difficulty and save development time.

The QoS (Quality of Service) refers to resource reservation control mechanisms, and it is an ability to provide different priority to different applications, or to guarantee a certain level of performance to a data flow [2]. QoS can be objective (encompassing reliability, availability, and request-to-response time) or subjective (focusing on user experience) [3]. For web service, it's necessary to evaluate the performance of the service [4]. Performance testing is a generic term that can refer to many different types of performance-related testing such as performance test, load test, stress test and capacity test. All of them are executed to determine how a system performs in terms of responsiveness and stability under a particular workload [5] [6]. All the four Web Services frameworks provide high quality of service and reliable message transmission. Besides, Axis2 and CXF support standards such as WS-Policy, WS-Security and WS-Reliable Messaging. This paper will analyze the performance of service based on the four frameworks in order to test the performance of the framework. Finally, the result of the performance testing will be given and suggestions will be provided for developers in order to help them to choose the appropriate framework.

However, the interoperation between different service's server and client may cause incompatible problems. For example, the client developed by Axis2 can not always invoke the web service based on CXF directly. This incompatible problems decrease the generality of the web service. WSDL is used to describe web service [7] [8]. By some targeted test, this paper analyzes the WSDL files produced by different frameworks on the same service, and finally gets the conclusions that the reasons of the incompatible problems are the differences of WSDL file. For the same service, different frameworks generate different WSDL files. The differences will be introduced in this paper. Like in [9], by analyzing the WSDL file's differences and repackaging the SOAP messages, this

*Corresponding author, Email: yanzhonghua@ieslab.cn.

doi:10.4304/jsw.9.8.2073-2079

paper presents a mechanism based on unified invoking component. By deploying a unified invoking component between the client and server, the incompatible problems can be solved successfully.

The rest of this paper is organized as follows. Section II provides an overview of Web Services technology and summarizes the main technology used in Web Services. In section III, the architecture and use method of the four frameworks will be introduced. Then, the performance testing of the four frameworks will be introduced in section IV and the WSDL differences of different frameworks will be presented in section V. In section VI, this paper will introduce the design and implementation of the unified invoking component. Finally, conclusions are made in section VII.

II. Web Services Technology

Software delivery models are changing constantly: from standalone applications to client-server architecture, then from browser-server architecture to service oriented architecture. SOA is a software design architecture which connects different units of application programs through good defined interfaces. Interfaces are defined in neutral ways in order to realize loose coupling between services.The SOA architecture makes sure the software can select and invoke components dynamically. The architecture of SOA and the interoperation of each components are shown in Fig. 1.

Web Services is an implementation of SOA and it aims to simplify the interoperation among different systems by defining a standardized mechanism to describe, locate, and communicate with applications on the web. In [10], the authors divided the Web Services architecture into three areas - communication protocols, service descriptions, and service discovery.

A. Communication protocols

In order to communicate with applications in different platform, the communication mechanism must be platform-independent, secure and as lightweight as possible. SOAP (Simple Object Access Protocol) is a protocol specification for exchanging structured information in the implementation of Web Services. It relies on XML for its message format, HTTP or SMTP for its Application Layer protocols and RPC for its call method [11]. SOAP is independent from hardware, platform and programming language. SOAP messages are composed by envelope and attachments. Envelope is made up by header, body and fault. The body contains the request and response messages.

B. Service descriptions

WSDL (Web Services Description Language) is a XML-based interface description language which is used for describing the functionality ordered by Web Services. WSDL defines a service's abstract description in terms of messages exchanged in a service interaction. It explains

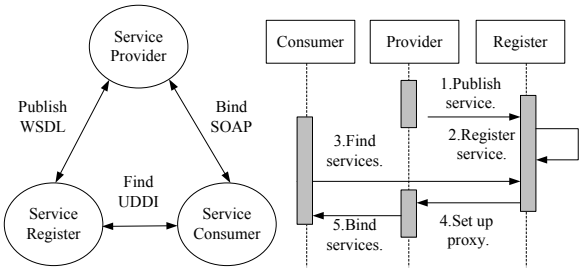

Figure 1. SOA architecture and component interoperation.

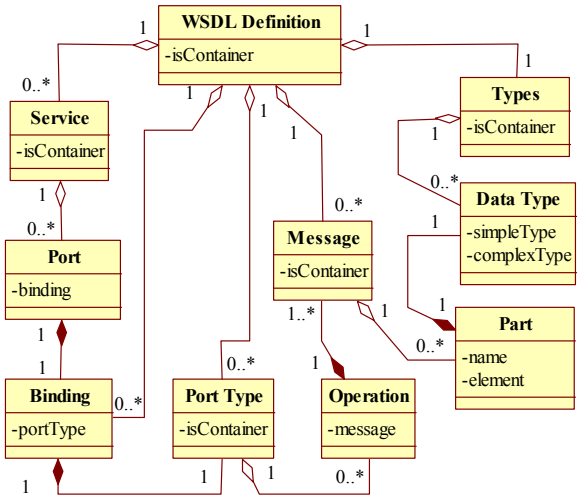

Figure 2. The Structure of WSDL document.

what Web Services do, where it is and how to access it [12]. A WSDL document is defined by a service provider and used by service consumers. The structure of WSDL document is shown in Fig. 2. It is divided by two parts: the interface document and the implementation document. The interface document is a container which is used to describe the abstract service. It is independent of implementation. The implementation document describes the details of the concrete service and deployment information such as part, operation and binding.

C. Service discovery

UDDI (Universal Descriptions, Discovery, and Integration) is a platform-independent, XML-based registry. It provides a mechanism to register and find web service applications [13]. By UDDI, the services can be resisted and the consumer can find and invoke services [14]. The transmission process of the UDDI message is shown in Fig. 3. By this process, the client can obtain the service methods and invoke the service.

III. Web Services Framework

Web Services frameworks such as Axis, Axis2, XFire and CXF are widely used in the development of web service. All of them provide well-defined framework and easy-use methods in order to make the development of web service easy and normalized.

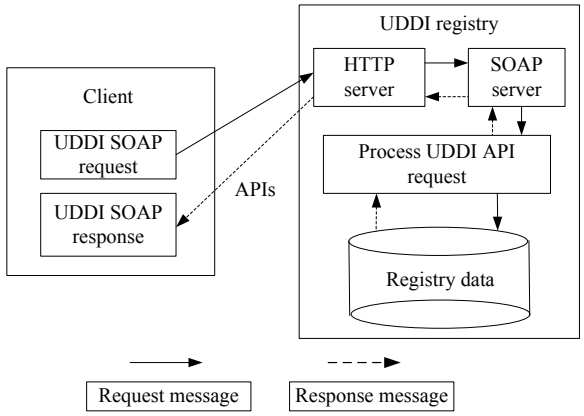

Figure 3. Transmission process of UDDI message.

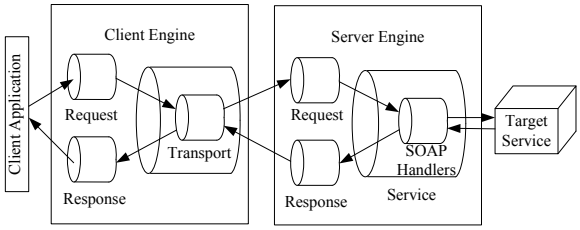

Figure 4. Architecture of Web Services frameworks.

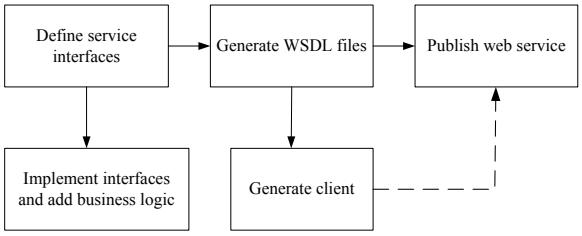

Figure 5. Development procedure of Web Services frameworks.

A. Axis

Axis is the abbreviation of Apache Extensible Interaction System. It is essentially a SOAP engine framework for constructing SOAP processors [15]. Axis framework achieves the interoperability of SOAP messages between server and client. Axis supports standard SOAP protocol and offers tools for monitoring TCP/IP packages and convert tool between WSDL and Java code, such as WSDL2Java and Java2WSDL. It provides two approaches to publish web service: Instant Deployment and Custom Deployment.

B. Axis2

Axis2 is an engine of Web Services, SOAP and WSDL. It is not developed on the basis of Axis but is redesigned with new architecture. This new architecture is more flexible, efficient, and configurable in comparison to Axis's architecture. What's more, Axis2 is built on Apache AXIOM, a new high performed, pull-based XML object model [16]. It uses its own object mode and StAX (Streaming API for XML) parsing to achieve significantly greater speed. Compared with Axis, Axis2 is more efficient, more modular and more XML-oriented. Moreover, there are two implementations of the Apache Axis2 Web services engine - Apache Axis2 for Java and Apache Axis2 for C++.

C. XFire

Codehaus XFire is the next-generation Java SOAP framework. The core of XFire is light message processing mode based on StAX which is used to exchange messages with SOAP message. XFire supports different types of band mechanism, container and transport protocol. It also supports SOAP, WSDL, WS-I Basic Profile, WSAddressing, WS-Security, supports Spring framework and the code generation of server and client [17].

D. CXF

CXF is the continuation of XFire project and is regarded as XFire 2.0. CXF helps us build and develop services using frontend programming APIs, like JAX-WS and JAX-RS. CXF supports both contract first development with WSDL and code first development starting from Java [18]. Besides, CXF and Spring have been integrated well, which making the developments of Web Services more convenient.

Although the implementation and the performance of the four frameworks are different, the architecture and the development procedure of the four frameworks are mainly the same. The architecture is shown in Fig. 4. The whole framework is made by two parts. One is the client engine, the other is server engine. The client engine is used by client application to send requests and receive responses. The function of the server engine is to parse and pack the SOAP messages, then invoke the target service and return response messages [19]. The two parts are connected by transport which can be seen as a message dispatcher. The development procedure of the four frameworks is shown in Fig. 5. Firstly, we define service interfaces, including the service name and parameters and so on. Then, the WSDL files can be generated by the tool which is provided by the frameworks. And we should implement the interfaces and add business logic to complete the service on the server. As to the client, it can be auto-generated by the framework or written by developers. Finally, the client can call the service on the server through Internet.

The framework is powerful and can really simplify the development procedure. But it is a troublesome thing to choose which frameworks to use. The performance testing of the four frameworks may give some suggestions.

IV. The Performance Testing of the Four Frameworks

According to the development procedure mentioned above, we develop four web services of the same function based on the four frameworks. The service's class diagram is shown in Fig. 6. It contains two methods: testPrimitive

and testReference. The testPrimitive method is used to test the primitive types such as byte, double and so on. The testReference method is used to test the reference types such as class, interface and arrays. Next, we publish the service as web service by Axis, Axis2, XFire and CXF, and generate clients separately. We use the client to call the service and test the performance of the four frameworks. The Apache JMeter is open source software which is designed to load test functional behavior and measure performance. It can be used to simulate a heavy load on a server, network or object to test its strength or to analyze overall performance under different load types. We can use it to make a graphical analysis of the performance of our web services under heavy concurrent load. There are some important indexes than can be used to evaluate the performance testing.

1) Average time is the average response time of each request. It is an index which can be used to evaluate the response speed.
2) 90% Line means that 90% of the samples take no more than this time. The remaining samples take at least as long as the value. This is a standard statistical measure.
3) Throughput means the quantity of the successful request per second. It is supposed to represent the load on the server.
4) Error% is equal the value which is calculated by the number of the error requests divided by the total requests. It can be used to measure the stability of the service.

The version of the framework is shown in Table I. In JMeter, thread is used to simulate the users who try to assess the server and invoke the service. We test 10 times for each frameworks, with the threads grows from 1000 to 5500 in order to simulate the growth of the workload. The result of the performance testing is shown in Fig. 7. The horizontal axis represents the threads and the vertical axis represents average time, 90% line, throughput and error%. The average time shows that the Axis2 and CXF do better than Axis and XFire, which means that they have a quickly response time. 90% line shows that Axis2 and CXF have a high stability. Compare to Axis and XFire, Axis2 and CXF are more powerful in throughput. Finally, the four frameworks do well in dealing with errors in low thread. The performance testing of the four framework shows that Axis2 and CXF are better than Axis and CXF. This is no difficult to understand. Because of Axis2 using the Stax to parse XML, Axis2 is about 2 times faster than Axis. CXF is designed based on XFire and adopt advanced technology. We can also see that CXF performs as well as Axis2. It's really difficult to make a decision from the performance testing result. But in other aspect, Axis2 provide many visual tools and the service is deployed in servlet containers. So, it is easy and convenient to manage and configure the services. Besides, Axis support C++ and Java. CXF supports Spring, developing web service by configure XML, this makes it easy to develop web service and can integrate with other framework. This may give some suggestions for our developers when choosing web service framework.

TABLE I.
TEST ENVIRONMENT

Components	JDK	JRE	Axis	Axis2	XFire	CXF
Version	1.6	6	1.4	1.5.4	1.2.6	2.5.1

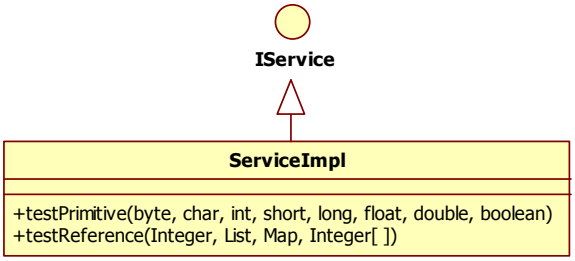

Figure 6. Class diagram of the service.

V. INCOMPATIBLE PROBLEMS BETWEEN DIFFERENT FRAMEWORK'S SERVER AND CLIENT

There are two ways to generate web service client. First, by using the tools provided by the frameworks to parsing the WSDL files, the client can be auto-generated. It is a fast way and there is no problem when we use tools to parse WSDL generated by the same framework. For example, we publish web service by Axis2 and generate WSDL files. Then we use Axis2's tools to parse the WSDL files in order to generate client. There is no doubt that the client can call the web service successfully. But, what will happen if we use CXF's client tools to parse WSDL files which is generated by Axis? This may cause problems. Because of the differences of the four frameworks, the WSDL files of the same service are not the same. We compare the WSDL files generated by the four frameworks of the same service and get the differences in Table II. The differences include three parts.

1) Target namespace is the concept in XML schema. In Axis's WSDL files, target namespace follows the format of "http://HostName:PortName/services+ServicesName". HostName, PortName and ServicesName are produced according to the server's hostname, port and the name of the service. For example, in this test, the target namespace of Axis is "http://localhost:8080/services/Service". Besides, in Axis2 and XFire, the target namespace are "http://server" while in CXF the target namespace is "http://server/", with a slash added.
2) The WSDL produced by Axis does not use schema files to restrict WSDL files. Data types are defined and used directly in the WSDL files. However, Axis2, XFire and CXF all use schema files to restrict WSDL files and quote schema files by the label "wsdl:types". The definition and use of the data types is separated, which have a good flexibility.

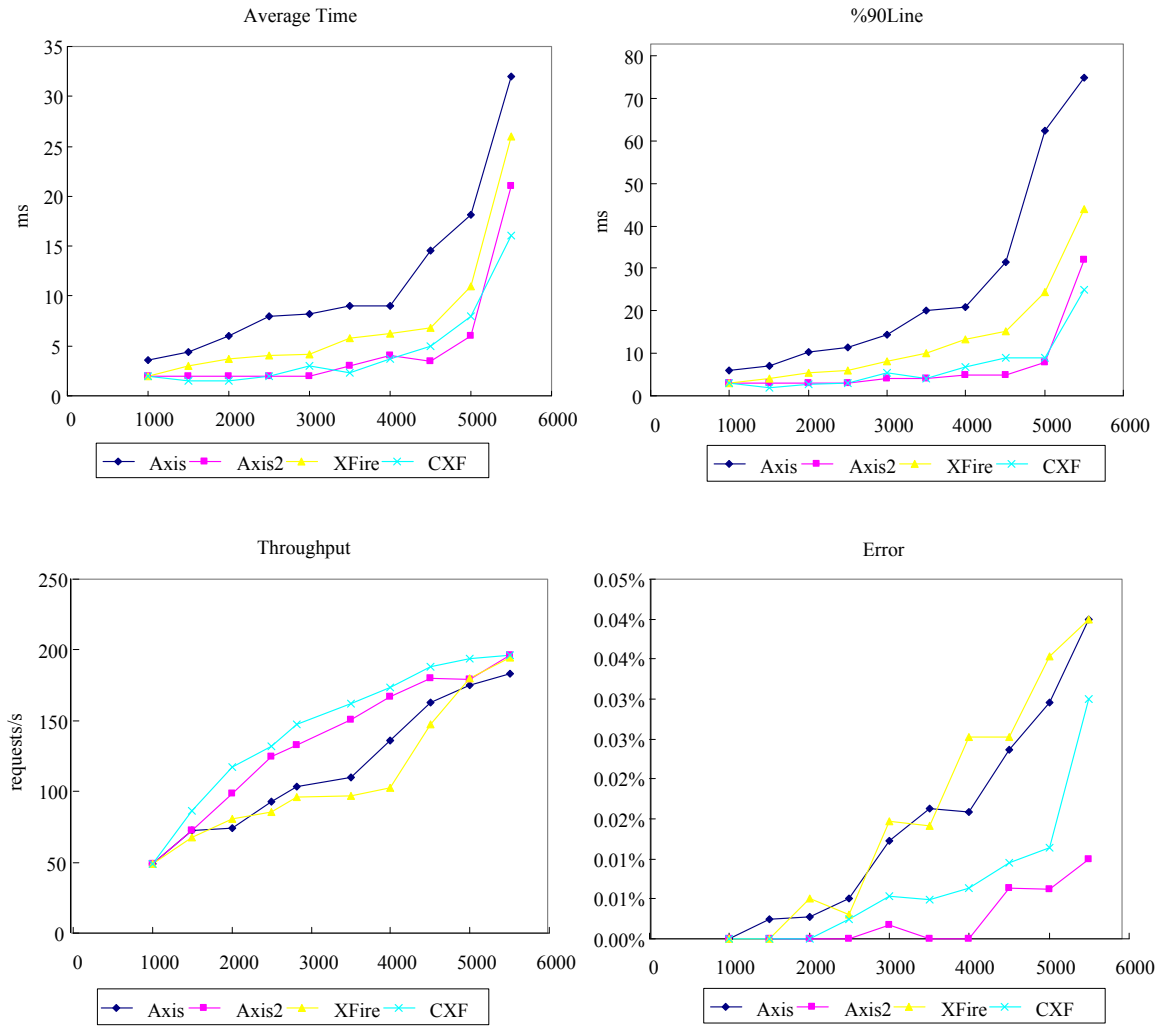

Figure 7. Results of the performance testing of different Web Services frameworks.

3) The WSDL labels, as is shown in Fig. 2, vary from framework to framework. Message label is used to define the data type. Axis uses the data types directly by the property "Name". However, Axis2, XFire and CXF all quote the data types defined in the external schema files by the label "Element". The label "portType" is made up by a group of abstract operations and relevant messages. In Axis and CXF, the content of this label is the name of the web service. But in Axis2 and XFire, the contents of this label conform to the rule of "ServiceName+PortType". The label "Service" is used to describe the interfaces of the Web Services. In Axis and CXF, the contents of this label follow the format of "ServiceName+Service". However, in Axis2 and XFire the content of this label is the name of the web service.

Because of the differences above, not all the clients generated by the four frameworks can call the service successfully. So, we can use the second way to generate web service client. We write the client manually by using the client APIs provided by the frameworks. In this way, developers should parse the WSDL files and extract parameters like target namespace, method name and so on. It is time-consuming and easy to make mistake. So, we want to deploy an unified invoking component to call the services based on different frameworks in spite of which client we use. By this component, we can use framework's tools to generate client in order to save development time, and at the same time, the incompatibility existed in the web service client can be eliminated.

VI. Design of the Unified Invoking Component

Through the analysis above, we know that the WSDL differences produced by different frameworks are the reasons of the incompatibility exist in different web service client. To solve this problem, we can deploy a unified invoking component in the client. The architecture of the component is shown in Fig. 8. It works like a proxy server. The main function of the unified invoking component is to receive SOAP request messages from different clients and then repack the SOAP messages in order to call the service on the server. After the call processes succeed, the component receives the result SOAP messages and transmits the results to client. By this mechanism, the diversity among different frameworks can be eliminated.

TABLE II.
THE WSDL DIFFERENCES ON THE SAME SERVICE.

WSDL difference	Web Services framework			
	Axis	Axis2	XFire	CXF
Target Namespace	http://HostName:PortName/+axis/services+ServicesName	http://server		http://server/
Schema	None	XML Schema		
Message	Use directly by the property Name	Use schema files by wsdl:types		
PortType	ServiceName	ServiceName+PortType		ServiceName
Service	ServiceName+Service	ServiceName		ServiceName+Service

The unified invoking component is made up by three modules: Transport, Client-to-Server and Server-to-Client. The functions of each module are as following:

1) Transport is an injection point. It used to receive request messages of different framework client and transmit the response messages to server according to the URL of the service. It works like a two-way message pipe which is used to dispatch message.
2) The module Client-to-Server is made up by three parts, SOAP parser, WSDL parser and SOAP wrapper. The SOAP parser is used to parse the request messages and extract the message such as the location of the web service, method name, and request parameters. The function of WSDL parser is to parse the WSDL files of the target web service and generate requests SOAP message that meet the requirement. Finally, the SOAP wrapper packs the SOAP messages with the parameters extracted from the SOAP parser and send them to the server.
3) The module Server-to-Client is used to parse the respond messages from the server and transmit the SOAP message to client. It is made up by SOAP parser and SOAP wrapper. Firstly, the SOAP parser receives the response message and extracts the result. Then, the SOAP wrapper packs the SOAP message with this result and sends it to the client. The procedure of the unified invoking component is shown in Fig. 9. The client calls the web service via the unified invoking component which acts like a proxy server. Then the component parses the SOAP message and extracts parameters like the location of the web service, method name and parameters and so on. The WSDL parser analyzes the WSDL and packs a correct SOAP request messages with the parameters extracted from client request. Then, the service can be invoked successfully and return a SOAP response. The component receives the response and extracts the result. Finally, the result will be repacked and transmitted to the client. By the analysis of the WSDL and repackage of SOAP messages, the requests raised by different client can satisfy the WSDL requests of different server. Therefore, client can access different server. On the base of the existing web service client, the unified invoking component solves the incompatible problems caused by different client call the service and maintains the unity of the web service.

The unified invoking component is implemented by Java language. The WSDL parser, which is the most important part of the component, is completed by wsdl4j package. It is mainly used to parse the WSDL files and extract service parameters like method name, endpoint and request parameters and so on. Request SOAP messages from the client are catched and parsed to extract the request parameters. After extracting the service parameters, the correct request SOAP messages are generated and sent to the server in order to invoke the service by RPC. Finally, the server returns the responses and the client extracts the results from the response SOAP messages.

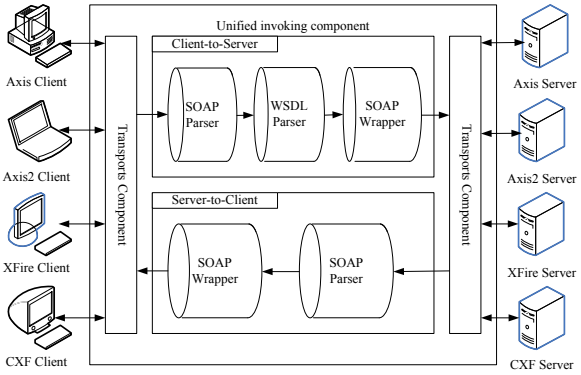

Figure 8. Architecture of the unified invoking component.

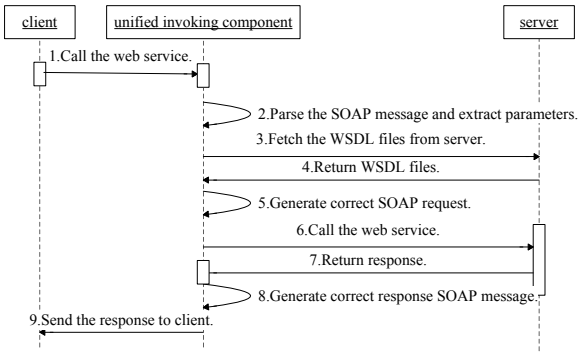

Figure 9. The corresponding procedure between server and client.

VII. CONCLUSIONS

After experiencing the stage of procedure oriented, object oriented and component oriented, service oriented is now prevalent in the development of software. And Web Services play an important role in SOA. It is platform independent and widely used in EAI and B2B. Web Services frameworks such as Axis, Axis2, XFire and CXF support well-designed architecture and easy used APIs which make the development of web service standard

and simple. However, there exist incompatible problems when different clients call the service. By a performance testing of the four frameworks, this paper analyzed the performance and gave some suggestions for developers to choose the appropriate framework. As regards to the incompatible problem, this paper recommended unified invoking component to solve this problem. The unified invoking component, which is based on the existing client, analyzed the WSDL of the service by WSDL parser and extract service parameters. Then, by the SOAP parser and SOAP wrapper, correct SOAP messages, which contain client requests, were generated and sent to the server. Finally, the results were extracted from the responses by SOAP parser. By the unified invoking component, the incompatible problems between different client and server can be solved successfully.

ACKNOWLEDGMENT

The authors would like to thank Integrated Electronic Systems Lab Co. Ltd. for providing the research environment and resources.

REFERENCES

[1] B. Benatallah, F. Casati and F. Toumani, "Web services conversation modeling: A Cornerstone for E-Business Automation," *IEEE Internet Computing*, vol. 8, no. 1, pp. 46-53, 2004.
[2] M. A. Serhani and A. Benharref, "Enforcing Quality of Services within Web Services Communities," *Journal of Software*, vol. 6, no. 4, pp. 554-563, 2011.
[3] L. Zh. Zeng, B. Benatallah, A.H.H Ngu, M. Dumas, J. Kalagnanam and H. Chang, "QoS-aware middleware for Web services composition," *IEEE Transactions on Software Engineering*, vol. 30, no. 5, pp. 311-327, 2004.
[4] X. Y. Bai, C. C. Zhao and G. L. Dai, "Research on Web Service Testing," *Computer Science*, vol. 33, no. 2, pp.252-256, 2006.
[5] C. A. Sun, G. Wang, B. H. Mu, H. Liu, Z. S. Wang, and T. Y. Chen, "Metamorphic testing for Web Services: framework and a case study," in *Proc. IEEE Int. Conf. Web Services*, 2011, pp. 283-290.
[6] Microsoft Corporation, Performance Testing Guidance for Web Applications, Microsoft Press, 2007.
[7] P. Sripairojthikoon and T. Senivongse, "Concept-based readability of web services descriptions," in *Proc. Int. Conf. Advanced Communication Technology*, 2013, pp. 853-858.
[8] Y. Jarma, K. Boloor, M.D. De Amorim, Y. Viniotis, and R.D. Callaway, "Dynamic Service Contract Enforcement in Service-Oriented Networks," *IEEE Transactions on Services Computing*, vol. 6, no. 1, pp. 130-142, 2013.
[9] S. Benbernou and M. S. Hacid, "Dynamic Web Service Calls for Data Integration," *Journal of Software*, vol. 1, no. 1, pp. 1-10, 2006.
[10] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi and S. Weerawarana, "Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI," *IEEE Internet Computing*, vol. 6, no. 2, pp. 86-93, 2002.
[11] B. Li, "Research and Application of SOA Standards in the Integration on Web Services," *Education Technology and Computer Science*, vol. 2, no. 1, pp. 492-495, 2010.
[12] S. Graham, Building Web Services with Java: Mading Sense of XML, SOAP, WSDL and UDDI, China Machine Press, Beijing, 2003.
[13] K. Sivashanmugam, K. Verma and A. Sheth, "Discovery of Web services in a federated registry environment," in *Proc. IEEE Int. Conf. Web Services*, 2004, pp. 270-278.
[14] M. Varguez-Moo, F. Moo-Mena and V. Uc-Cetina, "Use of Classification Algorithms for Semantic Web Services Discovery," *Journal of Software*, vol. 8, no. 7, pp. 1810-1814, 2013.
[15] "Apache:Axis," http://ws.apache.org/axis/.
[16] "Apache:Axis2," http://axis.apache.org/axis2/java/core/.
[17] "Codehaus:XFire," http://xfire.codehaus.org.
[18] "Apache:CXF," http://cxf.apache.org.
[19] Zh. Q. He, L. F. Wu, H. G. Lai and Zh. Hong, "Semantics-based Access Control Approach for Web Service," *Journal of Software*, vol. 6, no. 6, pp. 1152-1161, 2011.

Wenpeng Su was born in Shandong province, China, in 1988. He received the B.S. degree in electronic information science and technology from Shandong University, Weihai, China in 2011. Now he is pursuing his M.E. degree in circuits and systems in Shandong University, Weihai, China. His research fields are mainly intelligent measurement and control system.

Zhonghua Yan was born in Zhejiang province, China in 1966. He received the B.S. degree from Shandong University, China, in 1989. Now he is the professor of School of Mechanical, Electrical and Information Engineering, Shandong University, China and the vice chairman of Integrated Electronic Systems Lab Co. Ltd., Jinan, China. And he also is an expert who enjoys the State Council special subsidy. He has been engaged in research, development and engineering service work of power dispatching control center system since 1988.

Chenghui Liang was born in Shandong province, China in 1972. He received the B.S. degree in intelligent measurement and control system from Shandong University, China. Now he is a lecturer in the School of Mechanical, Electrical and Information Engineering, Shandong University, Weihai, China. His research fields are mainly object-oriented technology, distributed systems analysis and design and IEC61979/61968.