SURVEY ON WEB SERVICE TESTING METHODOLOGIES[1]

***Gaurav Jadhav, **Margret Anouncia S**

**School of Computing Science and Engineering, VIT University, Vellore, India.*

**gsj1991@gmail.com, ** smargretanouncia@vit.ac.in*

ABSTRACT

The Service Oriented Computing (SOC) architectural designs are allowing computer systems to interact with each other in new ways. SOC allows composition of distributed applications free from their platform and thus reduces the cost of such compositions and makes them easier and faster to develop. Currently web services are the most widely accepted service technology due to the level of autonomy and platform-independency they provide. However, web services also have some challenges. For example, testing web services at the client side is not as straightforward as testing traditional software due to the complex nature of web services and the absence of source code. This paper surveys the previous work undertaken on web service testing, showing the strengths and weaknesses of current web service testing strategies.

Key words : Service Oriented Computing, Web Services, Testing, SOA.

INTRODUCTION

Web services is a rapidly growing concept that drives the Service-Oriented Computing (SOC) at present. Web services present important challenges to software testers. These challenges has led to much work on techniques for testing web services. This paper explore the overall existing work.

According to Papazoglou [1], SOC is a new computing paradigm that utilizes services as the lightweight constructs to support the development of rapid, low-cost and easy composition of distributed applications. This is one of the widely used definition of SOC.

A recent survey by Canfora and Di Penta [2] summarizes the testing of web services and categorises the research undertaken, that three categories are: functional, regression, integration.

A. Service-Oriented Computing

SOC changed the traditional aspect of software application design, implementation and delivery. The main idea behind the use of SOC is that it should be able to create a more systematic and a more efficient way of building distributed applications. SOC based application are having more advantages over traditional distributed applications. SOC have two main characteristics

1. Platform independent services
2. Simple service integration

[1] How to cite the article:
Jadhav G., Margret A.S., Survey on Web Service Testing Methodologies, *International Journal of Advances in Engineering Research*, April 2013, Vol 3, Issue 4, 65-73

Recent studies underlined that many product manufacturer are moving toward web service provision.

B. Service Oriented Architecture

Service Oriented Architecture is a backbone for web services. Service Oriented Architecture is a powerful architectural design pattern which aims on loose coupling of software agents by reducing dependency among these software agents. Software Service Agents provides implementation for both service provider and service consumer/ customer /user. A work carried out by service provider for the service requestor is measured as service. Web Services are organised in such a way that service requestor can access them for their business need without knowing underlying implementation. Composite service based on requestors need can be formed by participation of more than one organisation.

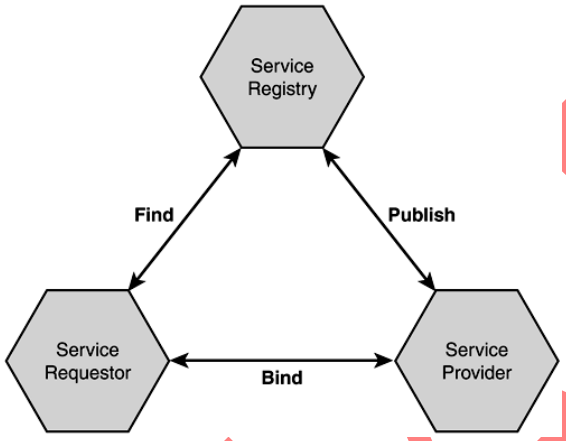

Fig 1: Service Oriented Architecture

C. Web Services:

A web service is defined as "a software system designed to support interoperable machine-to-machine interaction over a network". The main aim of the web service platform is to provide the interoperability among different applications using web standards.

There are different web service styles, they are all based on the SOA but different in the interfaces that they use. Some of them are

1. Representational State Transfer (REST)
2. Simple Object Access Protocol (SOAP)

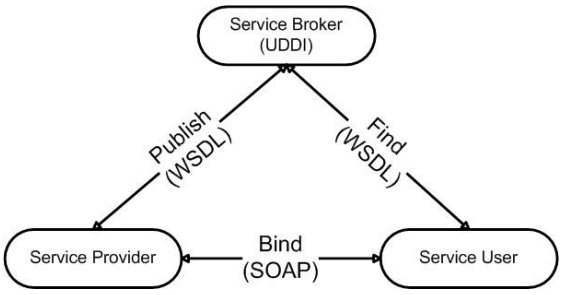

Fig 2: Web Service Architecture

1. **Simple Object Access Protocol (SOAP):**

   SOAP is an XML-based protocol that allows data exchange over the Hypertext Transfer Protocol (HTTP). The W3C definition of SOAP is "a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment" [3]. Since SOAP as a protocol combines platform independent XML and HTTP, SOAP messages can be exchanged between applications regardless of their platform or programming language.

2. **Web Service Description Language (WSDL):**

   W3C defines WSDL as "an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information" [4].

3. **Universal Description Discovery and Integration (UDDI):**

   UDDI is defined as "a set of services supporting the description and discovery of businesses, organizations, and other Web services providers, the web services they make available, and the technical interfaces which may be used to access those services" by OASIS (The Organization for the Advancement of Structured Information Standards) [5].

LITERATURE SURVEY

A. Testing Web Services

Testing web services at the client side is one of the major problems that slows down the wider use of web services. The increased use of SOC and web services required more attention on their testing. Testing web services includes testing of the basic web service functionality and web service interoperability [6].

According to Bloomberg [7], the history of web service testing is divided into three phases, based on the functionalities that are added to web service testing tools during these periods:

1. **During phase one (2002-2003)** web services were considered to be units and testing was performed in a unit testing using web service specifications.

2. **During phase two (2003-2005)** testing of SOA and its capabilities emerged. The testing in this phase included testing the publishing, finding, binding capabilities of web services, the asynchronous web service messaging capability and the SOAP capability of SOA.
3. **During phase three (2004 and beyond)** the dynamic runtime capabilities of web services were tested. Testing web service compositions and web service versioning testing emerged during this period.

Web Service testing brought more challenges to tester compared to traditional product testing. These challenges include:

1. Specification based testing using all or some of the web service specifications such as WSDL.
2. Runtime testing of web service activities (discovery, binding, publishing) due to the dynamic nature of web services.
3. Testing in collaboration due to the multiple parties involved in web service activities.

B. Test Case Generation

1. Unit Testing of Web Services

Unit testing can be considered the most basic and natural testing technique applicable to any system. In unit testing, individual units of a system that can be independently executed are consider as a units. In terms of web services, the operations provided by a service can be considered as units to be tested.

Unit testing of web services is performed by sending and receiving SOAP messages by the tester. The tester generates the SOAP messages for the operation under test using the information from the WSDL file. In this way, unit testing can verify both the correctness of the WSDL and the correct functioning of the System-Under Test (SUT).

According to Canfora and Di Penta [8] one of the main problems of functional testing of web services is its high cost. In testing, the best way of reducing the cost is automation and for unit testing there are existing tools that provide automated testing such as Parasoft SOA test [9], SOAP Sonar [10], HP service Test [6] and Oracle Application Testing Suite [16].

The need for tools that can automate unit testing was addressed by the research community. For example, Sneed and Huang [11] introduce a tool called WSDL Test for automated unit testing. WSDL Test is capable of generating random requests from WSDL schemata. WSDL Test is also capable of verifying the results of test cases.

Lenz et al. [12] propose a model-driven testing framework that can perform unit tests. In this approach, JUnit tests are generated using the requirement specifications.

An automated solution to well-known oracle problem is proposed by Tsai et al. [13]. Tsai et al. propose the adaptation of blood group testing to web services and call this technique Adaptive Service Testing and Ranking with Automated oracle generation and test case Ranking (ASTRAR) [14]. ASTRAR is similar to n-version testing where multiple web services that

have the same business logic, internal states and input data are tested together with the same test suite. Even though the main goal of group testing is to test multiple web services at one time to reduce the cost of testing and increase the efficiency, it also helps in solving the reliable test oracle problem within its testing process. ASTRAR can be applied at two different levels of testing: unit and integration testing. The ASTRAR testing process is divided into two phases:

1. A fast testing phase called the training phase.
2. A hierarchical testing phase called the volume testing phase.

Unit testing is one of the most important testing techniques that every system must undergo. The main challenge faced in unit testing of web services is the high cost of testing which can be minimized by automating the testing process. Most of the testing approaches explained in this section provide automated test data generation and test execution, though they lack automated test oracle generation. Tsai et al. [13], Chen et al. [15] and Heckel and Lochmann [16] address this problem and Tsai et al.'s approach provides fully automated test oracle generation.

2. Contract-Based Testing of Web Services

DbC [17] is a software development approach where contracts define the conditions (pre-conditions) for a component to be accessed and the conditions (post-conditions) that need to hold after the execution of methods of that component with the specified pre-conditions. Using contracts, some unexpected behaviour of the SUT can be detected and the information from contracts can also be used to enhance the testing process itself. Software testing using contracts has been applied to traditional software by many researchers [18].

Since traditional web services only provide interface information, researchers have proposed the use of contracts to be used in different aspects of SOA such as service selection, service composition and service verification. These contracts carry information on different aspects of SOA such as behaviour of services and QoS. This extra information on the behaviour of a service such as pre and post-conditions of operations increases the testability of services.

3. Fault-Based Testing of Web Services

According to Morell [9], fault-based testing aims to prove that the SUT does not contain any prescribed faults. The difference between fault-based test cases and regular test cases is that the fault-based test cases prove the nonexistence of known faults rather than trying to find faults that exist.

Fault-based testing can test web services for common faults that can exist in the SOA environment and increase the dependability of services. Hanna and Munro [5] classify test data generation for different testing techniques as follows

1. Interface propagation analysis that is performed by randomly inserting the input to a software component.
2. Boundary value based robustness testing where test data is chosen around the boundaries of the input parameter.

3. Syntax testing with invalid input where the rules of the specification of the input parameter are violated.
4. Equivalent partitioning with invalid partition class where the input space or domain is partitioned into a finite number of equivalent classes with invalid data.

4. Collaborative Testing

Collaborative software testing is the testing concept where multiple parties involved in a web service, such as developer, integrator, tester and user, all are participated in testing process. Canfora and Di Penta [8] identified some of the challenges that required collaborative solutions. These challenges that might require collaborative solutions are:

1. Users not having a realistic test set.
2. Users not having an interface to test webservice systems.
3. The need for a third-party testing and QoS verification rather than testing by each service user

Tsai. et al. [14] propose a Co-operative Validation and Verification (CV&V) model that addresses these challenges instead of the traditional Independent Validation and Verification (IV&V). In this collaborative approach all parties of a web service such as UDDI broker, providers and clients also participate in the same testing process.

Bai et al. [7] also propose a contract-based collaborative testing approach that extends Tsai et al.'s enhanced UDDI proposal. Bai et al. propose a Decentralized Collaborative Validation and Verification (DCV&V) framework with contracts. The proposed framework consists of distributed test brokers that handle a specific part of the testing process. Bai et al. suggest two types of contracts for the Decentralized Collaborative Validation and Verification (DCV&V) approach:

1. Test Collaboration Contracts (TCC) that enforce the collaboration among the test brokers.
2. Testing Service Contract (TSC) that is used for contract-based test case generation.

Zhu [19, 18] proposes another collaborative approach. In Zhu's approach service developers or third party testers provide testing services that help with testing the actual services.

5. Regression Testing of Web Services

Regression testing is the reuse of the existing test cases from the previous system tests. Regression testing is performed when additions or modifications are made to an existing system. In traditional regression testing it is assumed that the tester has access to the source code and the regression testing is done in a white-box manner. Performing white-box regression testing helps mainly with test case management.

According to Canfora and Di Penta [8], one of the main issues in Web Services Regression Testing (WSRT) at the user side is not knowing when to perform regression testing. Since the service user has no control over the evolution of the web service, the service user might not be aware of the changes to the web service. There are two possible scenarios for informing the service user about the modifications. These scenarios are based on the service provider's knowledge about the service users.

a. The first scenario arises when the SUT is registered to a UDDI broker or is not a free service.
b. The second scenario arises when the web service that requires regression testing is a public web service with no UDDI registration and the provider does not have information about its users.

Another challenge in WRST is the concurrency issues that might arise during testing due to the tester not having control over all participating web services. Ruth and Tu [17] discussed these issues and identified possible scenarios. Ruth et al. [12] also propose an automated extension to deal with the concurrency issues that might arise during WSRT.

Lin et al. [3] propose another Graph Walk Approach-GWA based regression testing approach where CFGs are created from Java Interclass Graph (JIG) [6]. A framework that performs RTS on the transformed code of a Java based web service is also proposed by Lin et al.

Mei et al. [9] propose a different black-box test case prioritization technique for testing web service compositions. In this approach test case prioritization is based on the coverage of WSDL tags in XML schemas for input and output message types.

Tarhini et al. [14] propose another model-based regression testing approach. The proposed model is capable of representing three types of modifications to the composite services:

a. Addition of a new service to the system.
b. Functional modifications to an existing service in the system.
c. Modifications to the specification of the system.

Tsai et al. [15] propose a Model-based Adaptive Test (MAT) case selection approach that can be applied to both regression testing and group testing. This approach defines a model called Coverage Relationship Model (CRM) that is used for test case ranking and selection. Using this model, test cases with similar aspects and coverage can be eliminated. Tsai et al. define multiple rules that guarantee the selection of the most robust test cases and prove that the less robust test cases never cover the more robust test cases.

Di Penta et al. [7] propose a collaborative regression testing approach that aims to test both functional and non-functional properties of web services. Di Penta et al.'s approach allows the developer to publish test cases along with services that are used in the initial testing and regression testing. The approach also reduces the cost of regression testing by monitoring service input-output.

6. Integration Testing of Web Services

It is important to perform integration testing in software engineering to make sure all the components work as a system. Since the idea behind SOA is to have multiple loosely coupled and interoperable services to form a software system, integration testing in SOA becomes a needful thing. By performing integration testing, all the elements of SOA can be tested including services, messages, interfaces, and business processes.

Bendetto [12] defined the difference between integration testing of traditional systems and service oriented systems. Canfora and Di Penta [8] point out the following challenges in integration testing of SOA:

1. Integration testing must include the testing of web services at binding phase, workflows and business process connectivity.
2. Stateless nature of SOA environment make integration testing harder.
3. Availability of services during testing might also be a problem.
4. Dynamic binding makes the testing expensive due to the number of required service calls.

CONCLUSION

SOC changed the business understanding of the whole software industry. However, the change from traditional software to services and the service usage is still not at the expected rate. One of the most important issues that prevent the wider use of web services is the issue of trust. One of the effective solutions to this trust issue is testing.

This survey has focused on the functional testing techniques and approaches that have been proposed for testing web services. Testing web services is more challenging than testing traditional software due to the complexity of web service technologies and the limitations that are caused by the SOA environment. The complexity of web services due to their standards not only affects the testing but also slows down the transition to web services.

REFERENCES

[1]M. P. Papazoglou, (2006); JDL special issue on service-oriented computing: Advanced user-centered concepts; *International Journal on Digital Libraries*, vol. 6, pp. 233–234.

[2] G. Canfora and M. Di Penta, (2008); Service-oriented architectures testing: A survey; in *Proceedings of the 31st International Spring Seminar on Electronics Technology (ISSSE2008)*, pp. 78–105, Budapest, Hungary.

[3] SOAP Version 1.2, [Online] http://www.w3.org/TR/soap12-part1/.

[4] Web Services Description Language (WSDL 1.1), [Online] http://www.w3.org/TR/wsdl.

[5] UDDI Spec Technical Committee Draft, [Online] http://www.oasis-open.org/committees/uddispec/doc/spec/v3/uddi-v3.0.2-20041019.htm.

[6] W. T. Tsai, X. Wei, Y. Chen, and R. Paul, (Oct, 2005); A robust testing framework for verifying webservices by completeness and consistency analysis; in *SOSE '05: Proceedings of the IEEE International Workshop*, pp. 151–158, Beijing, China, Oct. 2005, IEEE ComputerSociety.

[7] J. Bloomberg, (Sep, 2002); Web services testing: Beyond SOAP. [Online], http://searchsoa.techtarget.com/news/article/0,289 142,sid26 gci846941,00.html.

[8] G. Canfora and M. Di Penta, (March 2006), Testing services and service-centric systems: challenges andopportunities; *IT Professional*, vol. 8, pp. 10–17.

[9] SOA Test, [Online] http://www.parasoft.com/

[10] SOAP Sonar, [Online] http://www.crosschecknet.com/.

[11] H. M. Sneed and S. Huang, (Sep 2006); WSDL Test - a tool for testing web services; in *WSE '06: Proceedings of the Eighth IEEE International Symposium on Web Site Evolution*, pp.14–21, Philadelphia, PA, USA, IEEE Computer Society.

[12] C. Lenz, J. Chimiak-Opoka, and R. Breu, (May 2007); Model Driven Testing of SOA–based software; in *Proceedings of the Workshop on Software Engineering Methods for Service- oriented Architecture (SEMSOA 2007)*(D. Lubke, ed.), pp. 99–110, Leibniz Universit¨ at Hannover, FG Software Engineering, Hannover, Germany.

[13] W. T. Tsai, Y. Chen, D. Zhang, and H. Huang, (June 2005); Voting multi-dimensional data with deviations for web services under group testing; *in ICDCSW '05: Proceedings of the 4thInternational Workshop on Assurance in Distributed Systems and Networks (ADSN)*, pp. 65–71, Columbus, OH, USA, IEEE Computer Society.

[14] W. T. Tsai, Y. Chen, R. Paul, H. Huang, X. Zhou, and X. Wei, (July 2005); Adaptive testing, oracle generation, and test case ranking for web services; in *COMPSAC '05: Proceedingsof the 29th Annual International Computer Software and Applications Conference*, vol. 2, pp. 101–106, Edinburgh, UK, IEEE Computer Society.

[15] Z. Q. Zhou, D. H. Huang, T. H. Tse, Z. Yang, H. Huang, and T. Y. Chen, (Oct 2004); Metamorphic testing and its applications; in *Proceedings of the 8th International Symposium on Future Software Technology (ISFST 2004)*, Xian, China.

[16] R. Heckel and M. Lohmann, (March 2005); Towards contract-based testing of web services; in *Proceedings of the International Workshop on Test and Analysis of Component Based Systems (TACoS 2004)*, vol. 116, pp. 145– 156, Barcelona, Spain

[17] B. Meyer, (Oct 2992), Applying 'Design by Contract'; *Computer*, vol. 25, pp. 40–51.

[18] Y. Jiang, S.-S. Hou, J.-H. Shan, L. Zhang, and B. Xie, (Sep, 2005), Contract-based mutation fortesting components; in *ICSM: 21st IEEE International Conference on Software Maintenance (ICSM'05),* pp. 483–492, Budapest, Hungary, IEEE Computer Society.