Proceedings of the 12th INDIACom; INDIACom-2018; IEEE Conference ID: 42835
2018 5th International Conference on "Computing for Sustainable Global Development", 14th - 16th March, 2018
Bharati Vidyapeeth's Institute of Computer Applications and Management (BVICAM), New Delhi (INDIA)

Study of Automated Testing of Web Services

Anwar Bari
Department of Computer Application, Integral University, Lucknow, India.
dranwarbari@gmail.com

Mohd. Faizan Farooqui
Department of Computer Application, Integral University, Lucknow, India.
faizan_farooqui2000@yahoo.com

A. A. Zilli
Department of Comp. Sc. & Engg, Lucknow University, Lucknow, India.
aazilli@gmail.com

Abstract – **Web services are popularly supported by top industrial players. To guarantee the perfection in working of web service testing is done. As human testing is tiresome, efficient techniques are required which automatically generate test cases, execute and analyze the result for testing web services. Web services are easily accessible on the internet but the original implementation information is hidden. Having understanding about WSDL, automation tools generates the code to run as a client. The automation tools generate unit test cases and run these unit tests, which calls the service for testing. This paper offers a study on automated testing of web service. Although a lot of research is being done for testing web service. Web services progress rapidly to cope up with technological and organizational policies modifications. Testing is therefore necessary for ensuring that these updated or new services are working properly. Such kind of testing is done in audit testing which is a kind of regression testing. This paper is focused on proposing a framework which can perform both web services testing and auditing.**

Keywords – Audit Testing; Automated Testing; Web Services.

I. INTRODUCTION

Web service is an independent software module having definite boundary containing a collection of methods which is available on the Internet [1]. Every service is developed so as to execute a definite utility for any kind of operation. Service providers do not share internal source code or company policies to any third party for testing their web service. As web services are available without awareness of its internal coding details black-box testing is the only approach to test web services [2].

In this paper, a study has been carried out on automated testing of web services. The coding of web services is independent of programming language and platform, only requirement is usage of Web Services Description Language (WSDL) as standardized XML interface description, Simple Object Access Protocol (SOAP) as standardized messaging protocol and Universal Description, Discovery, and Integration (UDDI) [3].

During automation of web services the tester is having information related to service interface in WSDL, code is generated for service invocation to the service provider. Test classes are also generated that relate a single method to all the service operations. Generated classes are given to a test generation tool for object-oriented programs, which generates tests. Execution of the unit tests routinely sends requests to the web service.

This is quite clear that knowing only a web service WSDL, test cases considered precisely for validation of web services can certainly be automatically generated [4].

II. THEORETICAL BACKGROUND

Having knowledge about interface description from WSDL of a service, start with generating source code in Java to support the test cases generation process and also their execution.

Wrapper classes are also generated which relate each individual method to all operation offered by the service respectively. Then test suite is generated by giving these wrapper classes to test generation tool.

This generated test suite is used with the generated source code [1]. Executing this test suite will automatically call the web service by sending request messages [1]. The outcomes returned from the web service are then collected. Then problems can be detected by analyzing these accumulated outcomes.

In particular for automatic testing of web services the following four steps are generally followed [1]:

A. *Code Generation*

The code generation module produces necessary stubs and skeleton in Java using the WSDL [1]. WSDL is written in XML and describes the protocol bindings, message formats, and supported operations [2] needed for communicating with the web services enumerated in the UDDI [2]. The Axis utility, WSDL2Java, analyzes WSDL to generate the source code from a service provider's WSDL [1]. Axis [5] provides necessary Java code implementation of the SOAP protocol. The code generation component uses Axis to produce four classes or interfaces from the WSDL file. First classes in Java are generated for all methods considering the input and output parameters contained within service. Second a Java interface defining the connection point is generated for each port type. For all bindings, a stub class describing the message format and protocol is generated. The binding element has attribute name describing name of the service and a type attribute describing the port of the binding. Third service interface and their codes are generated. The service input output parameters defines the visible web services. These input output parameters assists in calling web service for testing purpose [1]. Fourth wrapper classes are generated to allow calling of available service. These wrapper classes have the web service invocation code for all methods held in the web service.

B. *Test Generation*

The test generation module is provided with these generated classes for generating unit test cases in Java for creation of a validation suite that drills those services pointed by the WSDL[1]. Required test cases are acquired by analyzing and converting test suites which are created for the system using the service's features [12]. It is required that any expression part of an assertion needs to be calculated and decoded into a literal. This methodology can function independent of the tool used for test generation [5] (e.g. JCrasher [8], Agitar Agitator [9], and Parasoft Jtest [2]).

More specifically, test generator examines the kind of information available in Java classes and creates a code portion which builds different types of calls to the web service for testing the performance of all procedures contained in the web service. The code portions along with the unit tests generated are called for providing inputs to all supported operations in a web service under test [2]. The supporting operation to be tested is within the wrapper class. For each service in WSDL a method is declared and for each input parameter in that service a method argument is designated [1]. Test generator produces unit tests which pass the required input parameters in order to call the web service.

C. *Test Execution*

The test execution module mainly executes the generated validation suite on the wrapper class in the stubs [1]. Each test case generated invokes the supporting operation within the wrapper class. These supporting operations are developed so as to pull the generated stubs which intern invokes the remote skeleton to be tested [1].

JUnit [2] invokes the unit tests corresponding to the generated wrapper class. JUnit is used to run a unit-test from the validation suite as regression testing framework against the wrapper class for testing [9]. If any failure occurs this execution of validation class can throw an exception [12].

D. *Response Analysis*

Whenever the service is invoked the consumer diverts the request to Response Analysis module, where the request is noted and redirected to the provider. This module will also note the responses or error condition as returned by the provider and redirects the same to the consumer. The response analysis component works as a man-in-the-middle in order to collect web service responses, between the consumer and the provider[1].

The collected responses are analyzed to indicate problems and may be presented for manual inspection.

Upon occurrence of an exception, manual assessment might determine if it might be triggered by an error in the code of web service or the inputs given did not follow the service provider's assumptions [1]. Such execution of the web service must return useful error message.

The response analysis module shortlists those tests from the responses which might show errors and sends the particular tests for manual assessment.

III. PROBLEMS AND NEED OF AUDIT TESTING

A critical analysis of issues occurring in Audit Testing of Web Services was performed in our previous work [12]. When services are updated, the user software or websites which incorporates these new services are required to be retested, so as to be sure that the already working functionalities are not affected because of updates. Audit testing of web services has restriction on time and cost.

We identified these problems which need to be well-thought-out while performing research on audit testing of services.

A. *Problem 1: Online Testing*

The test of services by working online should be compulsory done by audit testing for checking that newly incorporated services are well integrated and the intended application is working as required [4]. Portion of testing time should certainly be given for online testing, or testing by invoking services in real. Although such service invocation could upset the external applications in an undesired manner permanently. Currently there is no standard or infrastructure for supporting online testing of services [12].

B. *Problem 2: Change Detection*

When services are updated, it is not be detected by clients using these services, this is because of lack in standard for notifying of updates to clients using the service. Applications therefore depend on monitoring to judge each time something uncertain happens during invocation [12]. Therefore updates in service should first be detected so that audit testing of updated service can be done [4]. As updates are not suitably notified, changes can be detected by taking into account traces of executions of the service and analyzing the performance automatically.

C. Problem 3: Test Case Selection, Minimization and Prioritization

Having so less knowledge about services accessible for audit testing of service, we consider that the current scenarios for automated testing in use cannot be accepted for audit testing of services [12], for the reasons discussed below:

1. Test Case Selection:

As new compositions of service are being done from time to time, test case selection is used to identify those test cases which can target only the updated part of web service composition [3].

The portion of web service which can be validated by any test case is mysterious, so selecting the test cases for the code portion where changes have been made in the service composition is challenging for audit testing of services, due to non-observance of the service execution [12]. Therefore test case selection remains dependent on self-determining test cases which might call the updated service. Thus conditions for selection are not so strong. Further dominant procedures for selection are required.

2. Test Case Minimization:

The main objective of test case minimization is majorly in attaining the smallest section from the selected test cases which fulfills some acceptability norm [3].

Due to less observability of services test case minimization depending on code portion tested by test case (e.g., searching the least number of test suite which reserves certain level of coverage) is difficult to attain during automated testing [12]. It is unclear if test suite minimization can attend a defined code portion for audit testing of service which goes through updating. We reason that innovative minimization strategies are required for minimization [4].

3. Test Case Prioritization:

The main objective of test case prioritization is to put the minimized test cases in a manner such that the utmost significant test cases are called first (e.g., possible finding error) [3].

Although test case prioritization depending on percentage of service composition tested by applying test case is a useable choice, but undoubtedly not the efficient choice [12]

Using this problems centric discussion it is rather pointed out that extra research is needed, as neither common nor effective testing techniques are present for discussed problems and which can efficiently perform audit testing of web services [12].

IV. METHODOLOGY FOR AUDIT TESTING OF WEB SERVICES

Audit testing is useful for being sure that if any new service is introduced or old service is updated, then that service is correctly incorporated in the current system or application. In this present study, a framework has been proposed for testing and auditing of web service as shown in fig. 1, its efficiency in comparison to available techniques needs to be checked.

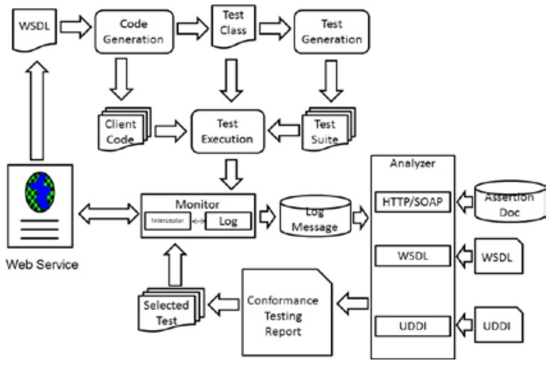

Fig. 1. Framework for Audit Testing of Web Services

As shown in fig. 1 the testing procedure is divided into steps. In step 1 taking into account the input and output parameters and other information available from WSDL of a service, we will first automatically generate code for creating stub for working as a client (service consumer) and send service requests to the skeleton (service provider). We will generate a wrapper class which covers all methods to a respective existing service operation. This wrapper class contains the service calls. This wrapper class will be passed on to any one of the test generation tool, which generates unit test.

In step 2 run this generated validation suite with the stub code, which will send SOAP requests to the skeleton. These will inevitably effect in sending web service requests to the service provider.

In step 3 Monitor intercepts the messages between service requester and web service in recessive method, transform the intercepted message into standard format, collect the responses returned and then outputs it to message logs.

In step 4 Analyzer judges whether the components of web service meets the requirement of the standards using a testing suite. After analyzing the large number of request-response pairs, we generate conformance report.

Hence, recognize the need for audit testing of a web service. Next we will implement measures for selection, minimization and prioritization of audit test cases. Test the technique. Analyze and evaluate the results obtained.

V. CONCLUSIONS AND FUTURE WORK

We have studied about the automatic testing of web services which included the automatic generation of code for implementing a client along with its supporting wrapper class and invocation of web services from the given service provider's WSDL[1]. Then with the use of automated unit test generation tools generated unit tests for the wrapper class and finally executed the generated unit test cases that invoked the service for testing. It shows that many web service tests can be successfully accomplished for any service provider. The generated requests are ready to expose bugs in the service provider's code.

While researches are being done on testing of web services. Web services are progress at a fast pace to fulfill the technological and organization policies update. Because of this, clients often get confused in deciding whether to be use up-to-date by incorporating the latest updates in the service composition, or keep using the current version in spite of knowing that this current version may be having issues, risks and limited support[12]. Most frequently the clients accept the updated version. As the current version may be discarded, as moving on to the latest version becomes necessary. Audit testing is done to cope up with these problems i.e. for being convinced that updated services are correctly incorporated in the current system or application.

In this regard a framework has been proposed for both testing and auditing of web services. In future work, the implementation of this approach may be planned to improvise and validate the effectiveness of this approach.

REFERENCES

[1] Evan Martin Suranjana Basu Tao Xie, "Automated Testing and Response Analysis of Web Services", IEEE International Conference on Web Services (ICWS 2007), 07/2007.

[2] E. Martin, S. Basu, and T. Xie, "Automated robustness testing of web services", In Proceedings of the 4th International Workshop on SOA And Web Services Best Practices (SOAWS *2006)*, October 2006.

[3] Masood, Tehreem, Aamer Nadeem, and Shaukat Ali. "An automated approach to regression testing of web services based on WSDL operation changes", 2013 IEEE 9th International Conference on Emerging Technologies (ICET), 2013.

[4] http://www.s-cube-network.eu/results/deliverables/wp-jra-1.3/PO-JRA-1.3.1-Survey-of-qualityrelated-aspects-relevant-for-SBAs.pdf

[5] Apache. Axis. http://ws.apache.org/axis/

[6] M. Di Penta, M. Bruno, G. Esposito, V. Mazza, and G. Canfora, "Web services regression testing," in Test and Analysis of Web Services, L. Baresi and E. D. Nitto, Eds. Springer, 2007, pp. 205–234.

[7] M. D. Penta, M. Bruno, G. Esposito, V. Mazza, and G. Canfora, "Test and Analysis of Web Services - Web Services Regression Testing". Springer Berlin Heidelberg, 2007.

[8] C. Csallner and Y. Smaragdakis. "JCrasher: an automatic robustness tester for Java. Software: Practice and Experience", 34:1025–1050, 2004.

[9] Agitar. Agitar Agitatior 2.0, Novermber 2004. http://www.agitar.com/

[10] E. Gamma and K. Beck. JUnit, 2003. http://www.junit.org

[11] K. Ashok, G. Kumar, A. Dhawan, and A. Dharani, "Automated regression suite for testing web services," in Int. Conf. Software Maintenance (ICSM), November 2009, pp. 590–592.

[12] Tao Xie. "WebSob: A Tool for Robustness Testing of Web Services", 29th International Conference on Software Engineering (ICSE 07 Companion), 05/2007.

[13] Anwar Bari, A. A. Zilli and S. Qamar Abbas. "Critical Analysis of Issues in Audit Testing of Web Services" International Journal for Scientific Research and Development 4.8 (2016): 645-650.

[14] C. D. Nguyen, Alessandro Marchetto, Paolo Tonella, "Challenges in Audit Testing of Web Services" 2011 Fourth International Conference on Software Testing, Verification and Validation Workshops

[15] Cu D. Nguyen, Alessandro Marchetto and Paolo Tonella, "Test Case Prioritization for Audit Testing of Evolving Web Services using Information Retrieval Techniques," In.proc.of the IEEE International Conference on Web Services (ICWS), Washington, DC, pp.636 - 643, 2011.