ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 7, July 2015

An Improved Methodology for Deploying Robust Web Services using UniLoG Technology

Manish T. Siddhu

P.G. Student, Dept. of Computer Engineering, Dhole Patil College of Engineering, Wagholi, Pune, India

ABSTRACT: Nowadays, many businesses use Service Oriented Architecture to serve their customers, like online shopping, banking, reservation systems for transportation, etc. But it becomes very difficult for developers to develop such a robust services over internet to serve whole world a flawless services. There are web services in large number which faces robustness problem. These web services has to face problems like unpredictable inputs provided to web services or unpredictable behavioral outputs of web services. Previous proposed techniques to identify robustness problems had no practical approach to fix the robustness of web services system. In this paper, proposed the methodology for detecting the robustness problems it is necessary to work out a practical approach with well-defined domain parameter to automatically improve web services found publically over the internet.

KEYWORDS: Web service integration, Reliability, Interoperability, Code tuning, Debugging, Testing, Networking

I. INTRODUCTION

These days, there are many businesses like banking, reservation systems for transportation, e-commerce shopping web services are based on Service Oriented Architecture to best serve their customers online. These web services are important element of Service Oriented Architecture across the internet.

Service discovery, communication protocols and service descriptions are divided in three major areas of web services framework. These major areas are all XML based like UDDI, SOAP and WSDL. XML stands for Extensible Markup Language it is a markup language. A set of rules are defined to encode XML document in such a format that human can read and understand as well as machine can read and understand too. XML is widely accepted and well established language which allows data and information encoding, internationalization and its platform independent. UDDI (Universal description, discovery and integration) offers a combined system to search service providers via a centralized system or registry. SOAP (Simple Object Access Protocol) this protocol offers a simple messaging that can run over an existing Operating Systems like Linux and Windows using transport protocols like XMPP, HTTP and SMTP. WSDL (Web Service Description Language) which is used to define web service to exchange specific messages types for communication between collections of electronic devices end to end. [1]

Program bugs and defects are the software faults which are the major cause of system failures. In service oriented environment, problems in software components, modules with bugs and faulty interfaces are present. Normally developers have to face many aspects while developing the system and they are very much under pressure like time to market, which ultimately makes developers to be focused on development of the system and ignores the importance of testing the system which finally may result system software in leftover bugs. These days, dynamic web services are deployed which uses vast range of software systems growing and deployed usually on an unreliable internet network.

It is expected that the web service should be able to provide tough web services for clients even if invalid inputs are present which may be due to bugs in the software systems, are main cause of system failures and security attacks. It is expected that that application servers should be able to deal with invalid inputs, parameters are send through malicious clients or the faulty computer systems. Which results in financial loss to service providers, loss in reputation, clients loyalty.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 7, July 2015

II. RELATED WORK

These days the web service provider serve their customers the on demand services online. The backbone of these services are the strong dedicated infrastructure which includes databases, application servers, payment gateways and operating systems [7]. As the customers use web services these services have dynamic interfaces, organizations collaborate to achieve to work together here particularly in service oriented services arise interface faults. This faults keeps service providers on their tosses to constantly keep testing and managing their web services frequently.

A. *Testing Robustness:*

There are old robustness testing techniques available [7], [11] for micro kernel and operating systems which are used to test the robustness of the web services nowadays. In [15] they have shown a way when there is tampered SOAP message how to assess the web services behaviour thoroughly. Invalid call parameters where used to test the set of robustness of the web services. Authors [7] proposed robustness scale named CRASH for Operating Systems. Following are the work proposed by [15] has some limitations.

The work conducted by [15] has some limitations like it does not considers the external responses from the web services for robust testing. This effects the comprehensive test coverage of the web services system. While running some process the flow of the running process is not always be related directly inputs from the client side. For example, when a user make the payment to some web service then that web service invokes the third party i.e payment gateway service to complete the transaction process. Payment gateway service may respond to requests like transaction denied or transaction confirmed so this kind of behaviour all depends on payment gateway service. So it is very important to test these external services for robustness because user cannot produce such kinds of inputs and so it will be difficult to handle any kind of errors by system itself on its own. While using the web service it becomes highly relevant to gain problems when the external services are depended upon the third parties services. This will result into no control of the bugs, updates, implementations and errors in the main services of the external third party.

Further, the other problem in [15], identifying the robustness issues are not fully automated. There are tools available which collects the outputs of the robustness test sometimes classification of robustness test can be done into regular response and robustness problem but sometimes its difficult to classify the robustness test because domain are not fully defined or may not able for tools.

In [13] the authors have discussed a methodology using parameter mutation to test web services. A parsed WSDL file (i.e Web Service Description Language) is applied with mutation operators which is used to test the web service. Practically the attempts for this technique there are limited operators for parameter mutation i.e adding, switching, setting complex types to null and deleting elements. Same way in [16] identical efforts have been made. In spite the studies made in [16] are more complete, but the technique used are for eXtensible Markup Language (XML) which is tightly coupled to XML.

The main point of web services robustness testing is to generate workloads, it is very important that workload should be able to exercise complete code of web service whether it may be random generation of workload. In [12] it is proposed to use state charts to generate automated test cases. Another technique is to use Markov Chians to generate characterised workload from real load patterns. There are some problems in these approaches like we need to the detailed source code analysis and are complex technique. Hence, there are preceding papers which unveils the randomly generated values are more appropriate in many cases [9].

There are tools available known as analysers of the coverage of code like Clover [3] and Cobertura [4]. These tools can used for identifying automatic the test cases. These did not have to access the source code, these tools only needs bytecode, this helps to increase the coverage of test cases.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 7, July 2015

B. *Made Improvements in Robustness:*

There were various efforts made in past to improve the systems robustness using a technique known as wrappers. Wrapper are the objects which as at least one objects identical interface which wrapper contains. Repeated calls are made to the target object, and wrapper has its own functionality that it can add after or before the call. This helps us to gain flexibility because it is possible to change the functionality of the wrapper keeping the contained component as it is without changing it.

A toolkit proposed in [5] known as HEALERS its a toolkit which helps to enhance application security and robustness. This is used for applications build in C to protect it from errors in C library functions. How this toolkit work is it stops calls towards the C library, which are found by failures of APIs of C library, arguments which are in large numbers a reason for software failures. Additionally, use of faultinjection technique is used by HEALERS to find more security and robustness problems in third party system softwares. HEALER then generates sets of fault containment wrappers by using the information from fault injection process to try to solve the problems. In fact generation of wrappers with this technique is more flexible and it is highly configurable. Wrappers generated through this process is hold its place between its shared libraries and an application given. There are some flaws in HEALERS that were discovered later like the toolkits performance and extensibility, more work has beendone to solve these problems in [14]. In [14] contributed the research on fault injection tool, a static analysis is used for arbitrary functions to achieve fault injection and a protective hypotheses in generated with a table based technique which is observation based on behaviour to protect the wrappers.The above hypotheses is fundamentally Boolean Expressions foreseen of the function to which arguments are present.

There are also robustness problem seen in off the shelf products. So, a wrapper based approach is proposed by [10] to protect the components of the off the shelf products. The technique author has described is to develop an application level protective wrappers. The wrappers developed in this approach helps to solve frequent problems can be found in components of, off the shelf products like incorrect specifications of components, some extraordinary featured components provided in the complete system and also helps solve the problems of outputs from incorrect command produced by the components in the environment. The study done will be helpful to develop wrappers from the traced information collected, there is also techniques provided to implement in current technologies and to analyse the improvement in gain there is the development in rigorous models. In [10] authors insist to developers to construct different view for the whole system and the components of the system. These views should be developed professionally and should be integrated as protector.

An off the shelf product Windows NT a technique is discussed in [6] to develop robustness wrappers. This will help to minimize the failures of Operating System and application drastically. Hence, there in need of wrappers to and it is well known technique to improve the systems robustness problems.

III.IMPLIMENTATION DETAILS

A. *Problem Statement:*

Program bugs and defects are the software faults which are the major cause of system failures. In service oriented environment, problems in software components, modules with bugs and faulty interfaces are present. Normally developers have to face many aspects while developing the system and they are very much under pressure like time to market, which ultimately makes developers to be focused on development of the system and ignores the importance of testing the system which finally may result system software in leftover bugs. These days, dynamic web services are deployed which uses vast range of software systems growing and deployed usually on an unreliable internet network.

B. *Proposed System:*

To make the web services robust it is very important to discard systems robustness problems. Following is the proposed way that will provide web services to handle invalid parameters.

Get all the information about the web services thoroughly by getting all the list of parameters, operations and data types. Create a workload and execute the workload by choosing one or more generating strategies of workload. Create

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 7, July 2015

workload of service. Execute the created workload to verify the service answers. To test the robustness run a set of tests to identify the issues of the web services. If needed can go to previous steps. After fixing the robustness of web service should verify the behaviour of service.

Next is to generate automated workload process. Aim is to create various sets of service citations which will be capable of implementing any web services. The workload generated will best serve the testing basis for robustness, this help explore the domain of parameters in very much effective way as compared to workload generation used for random values.

An improved workload generation and fault adding methodologies in Fig. 1 can to be used to test and identify the problems in robustness of the web services implemented. This will be addition to previous testing methodologies which will include automatically detect problems. Adding fault model will extend the test studies on robustness [7], [11] previously done. Results of the test will be classified automatically.

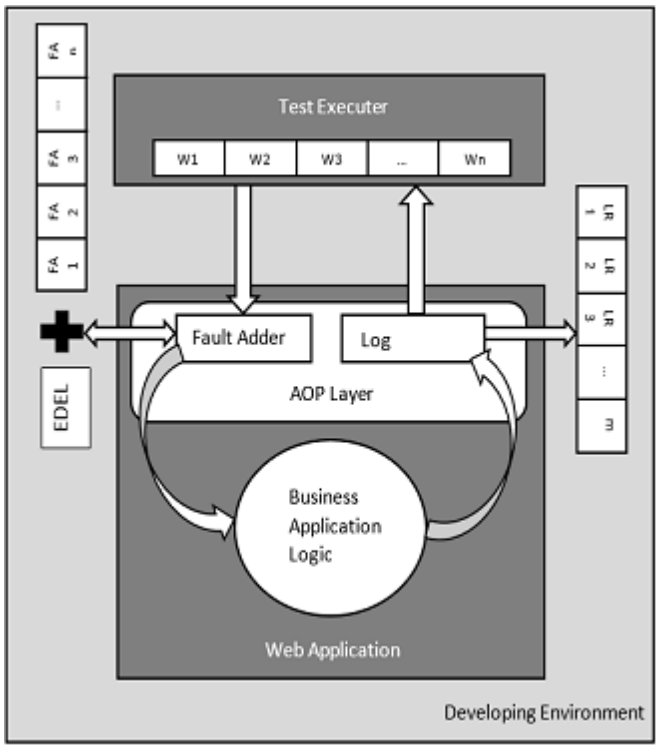

Fig. 1 Process of Fault Adder

To eliminate the robustness problems a test load technique will add the capability for performing the input verifications that will be adding mutation testing rules. This technique will be used to gain the modification to necessary targeted services.

Further an UniLoG [17] technique will be used to test the load of the web service overthe network if it's performance is good and robust to illegal activities which can harm the web services in long run. Which will affect the clients who are using the web service application.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 7, July 2015

C. *System Architecture:*

Following diagram shows system architecture.

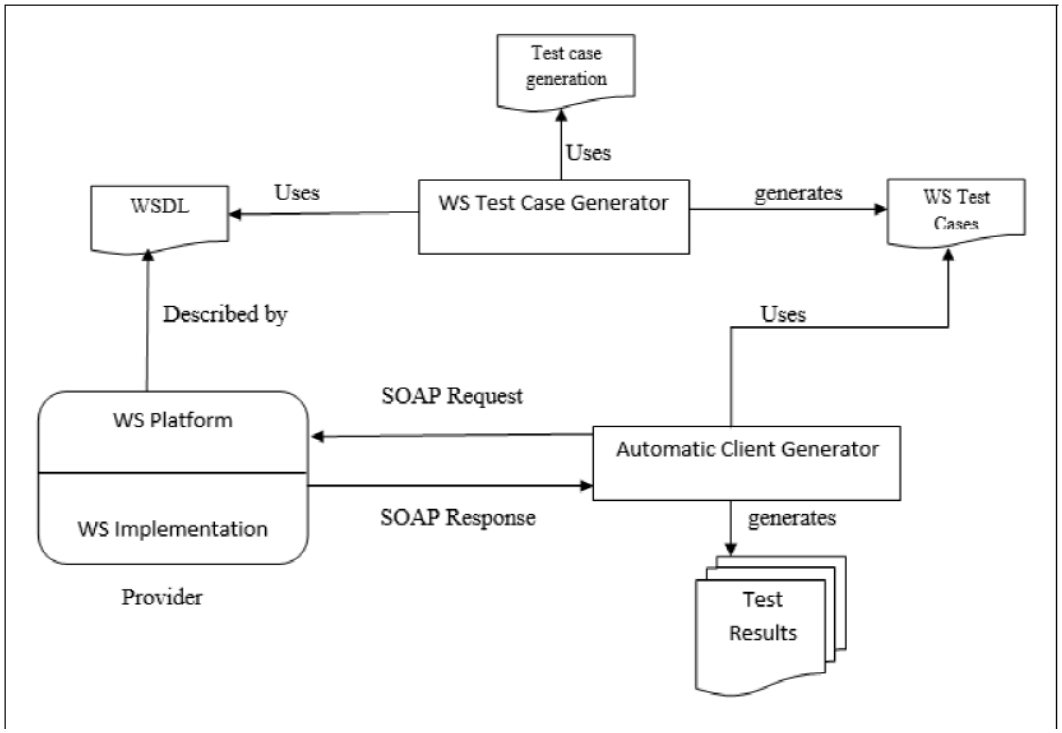

Fig.2 System Architecture

IV. EXPERIMENTAL SETUP

To test the benchmark TPC App will be used and a set of web services that are available over the internet which is open source. A widely accepted benchmark will be used for web services performance from TPC App. To test this experiment following services of TPC App will used like Product Detail, New Customer, Change Payment Method and New Product. An external web service will be called like payment gateway for New Customer and Change Method of Payment. The functions performed by Open Source web services will be adding deleting data to databases, manage information of student, mimic operations of bank and phone book address management as well as we are generating the workload on server when the web services are live, up and running. This workload will help programmers to again go through there codings to increase the server performance for Quality of Service (QoS) as well as the performance on network. This wrokload generation will be based on concept Unified Load Generator (UniLoG) [17] which is being used by many authors while generating workload for servers. Basically because of the pre-eminent importance of the WorldWide Web (WWW) a strong desire emerged to embed the capability into UniLoG of generating realistic realistic Web server loads.

V. RESULT SET

The goal of proposed model is to implement an efficient approach to web service should be able to provide tough web services for clients even if invalid inputs are present which may be due to bugs in the software systems, are main cause of system failures and security attacks. It is expected that that application servers should be able to deal with

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 7, July 2015

invalid inputs, parameters are send through malicious clients or the faulty computer systems. Which results in financial loss to service providers, loss in reputation, clients' loyalty.

The robustness testing of a system is a practical technique which characterize the overall behavior of the system software in enormous presence of erroneous inputs. These tests submits the erroneous inputs to the system to simulate the tests and trigger internal errors. It is been used to test the robustness of the operating systems [7]. Not long ago, robustness has been extended to the web services. In a scenario like reusing the component based services which makes web services reusable an interface level structure were developed separately for web services. Testing robustness will help developer to lower the efforts and make software system more robust.

Further, it is also important to test the load of the web services which are up on server online dealing with data which will help clients to serve the application with best performance and well tested robust system. Fig 3 shows the workload generation of three different services with their data being different and coding styles. The proposed system shows increased in workload coverage of system which been generated of web services which are been online up and running.

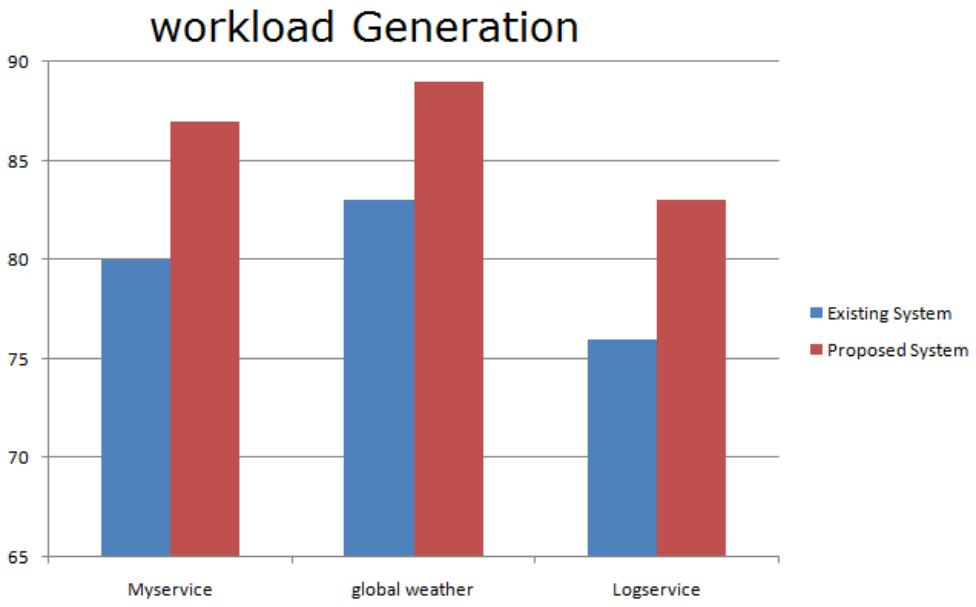

Fig 3: Workload Generation Chart

VI. CONCLUSION AND FUTURE WORK

This system will fill the gaps of previous robustness system issues which can provide validation, testing of application, validation and data generation. So an improved method for deploying web services can be achieved to mitigate the redundancy and lack of integrity problem to web applications.

In future, web services will boom more and every applications that will be on any devices from smart phones to laptops all will be web services running in device browsers. In fact whole organization will be dependent upon mobility devices. So, in future the applications build will be using browser languages but with compatible views in respective devices. This will eliminate the installation of applications, eliminate learning of languages particular to specific OS. Basically, by using browser based languages one can build built one web service application that can run on all devices. So, testing and making web services more robust especially that can be used on any devices.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 7, July 2015

REFERENCES

[1] Nuno Laranjeiro, Marco Vieira, Henrique Madeira, "A Technique for Deploying Robust Web Services". IEEE Trans. On Service Computing, January-March 2014.
[2] V. Bergmann, Databene Benerator. http://databene.org/databene-benerator, 2010.
[3] Atlassian, CloverCodeCoverage Analysis. http://www.atlassian.com/software/clover/, 2010.
[4] M. Doliner, Cobertura. http://cobertura.sourceforge.net/,2010.
[5] C. Fetzer and Z. Xiao, "HEALERS: A Toolkit for Enhancing the Robustness and Security of Existing Applications". Proc. IEEE/IFIP Intl Conf. Dependable Systems and Networks, pp. 317-322, June 2003.
[6] Ghosh, M. Schmid, and F. Hill, "Wrapping Windows NT Software for Robustness". Proc. 25th Ann. Intl Symp. Fault-Tolerant Computing, pp. 344-347, 1999.
[7] P. Koopman and J. DeVale, "Comparing the Robustness of POSIX Operating Systems". Proc. 29th Ann. Intl Symp. Fault-Tolerant Computing, 1999.
[8] N. Laranjeiro, S. Canelas, and M. Vieira, "wsrbench: An On-Line Tool for Robustness Benchmarking". Proc. IEEE Intl Conf. Services Computing, 2008.
[9] A. Mukherjee and D. Siewiorek, "Measuring Software Dependability by Robustness Benchmarking". Trans. Software Eng., vol. 23, no. 6, pp. 366-378, 1997.
[10] P. Popov, L. Strigini, S. Riddle, and A. Romanovsky, "Protective Wrapping of OTS Components", Proc. Fourth ICSE Workshop Component-Based Software Eng.: Component Certification and System Prediction, 2001.
[11] M. Rodrguez, F. Salles, J.C. Fabre, and J. Arlat, "MAFALDA: Microkernel Assessment by Fault Injection and Design Aid", Proc. Third European Dependable Computing Conf. Dependable Computing, pp. 143-160, 1999.
[12] V. Santiago, A.S.M.D. Amaral, N.L. Vijaykumar, M.D.F. Mattiello-Francisco, E. Martins, and O.C. Lopes, "A Practical Approach for Automated Test Case Generation Using Statecharts" Proc. 30thAnn. Intl Computer Software and Applications Conf., pp. 183-188, 2006.
[13] R. Siblini and N. Mansour, "Testing Web Services", Proc. ACS/IEEE Third Intl Conf. Computer Systems and Applications, p. 135, 2005.
[14] M. Susskraut and C. Fetzer, "Robustness and Security Hardening of COTS Software Libraries", Proc. IEEE/IFIP 37th Ann. Intl Conf. Dependable Systems and Networks, pp. 61-71, 2007.
[15] M. Vieira, N. Laranjeiro, and H. Madeira, "Assessing Robustness of Web-Services Infrastructures", Proc. IEEE/IFIP 37th Ann. Intl Conf. Dependable Systems and Networks, pp. 131-136, 2007.
[16] W. Xu, J. Offutt, and J. Luo, "Testing Web Services by XML Perturbation", Proc. IEEE 16th Int'l Symp. Software Reliability Eng., p. 10, 2005.
[17] Andrey W. Kolesnikov and Bernd E. Wolfinger, "Web Workload Generation According to the UniLoG Approach", 17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11)., pp. 49-60.