Volume 4, Issue 12, December 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

An Improved Methodology for Deploying Robust Web Services

Manish Tejvir Singh Siddhu
Computer Department & University,
Pune, India

Prof. Shivani Sthapak
Information Technology Department & University
Pune, India

Abstract—It is very difficult to develop robust web services to provide flawless services to the world. There are large number of web services which faces robustness problem like unexpected behavioural outputs to an unexpected inputs given to web services. There are already proposed robustness identification techniques by other authors but there is no practical approach to fix the system. The work proposed by N. Laranjeiro et.al [1] to detect robustness issues and solve them using well defined parameter domain. This will automatically improve three different implementation of TPC-App services and services available publicly over the internet.

Keywords—Web Service Integration, Reliability, Interoperability, Code Tuning, Debugging, Testing

I. INTRODUCTION

Many businesses use Service Oriented Architecture like, banking, e-commerce online shopping, transportation, etc. which serves clients and market suppliers. Framework of web services are divided into three parts which are service discovery, communication protocol and service descriptions. There is an established language XML of each specification area like WSDL, SOAP which are XML-Based which enables I20N (internationalization), data encoding and independence of platform. It is the web services which provides simple interface in between both provider and client. Here provider offers sets of distinct services to customers which are used by customers latter. Major faults in softwares causes system failures. Web services are dynamic, use an immense diversity of software systems. There is immense need of robust web services on online client applications even when the input is wrong. Robustness is the efficient and effective technique which characterizes the behaviour of the system. BlackBox approach of this type of interface is used to give enormous inputs to system to determine the systems' robustness testing.

II. PROBLEM DEFINITION

I. Lee et.al [7] and M. Kalyanakrishnam et.al [4] described being faults in softwares with program bugs or defects in program are major cause of computer system failures. In service oriented environments particularly faults in interface are problems in the interaction among software components/modules. At present, web services use immense diversity of system software, web services are dynamic, evolving and are frequently deployed over public unreliable network. Besides, web services developers usually have to face time to market pressure, which leads web services developers to focus on functionality development and neglect important testing activities, which may result in software with left behind bugs designated by N. Laranjeiro et.al [1]. Web services must be able to provide with robust client applications even in the presence of invalid inputs to the systems that might occurred because of bugs in the client applications, data exploitations can happen due to network failures or even security attacks. When businesses run through web application and in any case it is not acceptable that a server application is not capable to handle invalid parameters approaching from faulty or malicious client, this may impact the service provider's business finances, customer loyalty, reputation or collaboration with other organizations.

III. PERSPECTIVE SOLUTION

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 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 step. After fixing the robustness of web service should verify the behaviour of service.

IV. RELATED WORK & LITERATURE SURVEY

There are web services providers which have infrastructure which typically has application server, OS and set of external systems like payment gateway, databases, etc. There are frequent faults found in service oriented environments dynamic reuse and collaboration between services is frequent. While the system or software developer should create and test its services other than regular client requests, the system should handle unexpected security attacks, handle exceptional cases due its bugs. P. Koopman et.al [5] and M. Rodrı´guez et.al [8] described web services robust testing is the adaption of previous traditional robustness testing it would be typically for OS and micro kernel. Altered SOAP messages was the first approach to assess the behaviour of web services. SOAP has a set of robustness tests based on invalid call parameters. Services are classified by P. Koopman et.al [5] for OS is the adaption of CRASH robustness scale. Client side inputs are not always directly related to execution flow that handles a particular request. For example when client is using some web service and making an online payment which invokes payment gateway service, the responses from payment gateway service like transaction complete or insufficient balance are important for web services which client is using. Which makes sense and it is very important to test the web service robustness at initial request point so that client will not be able to produce an input that can reach a particular error handling code point on its own. This problem will increase high significance while in use with other external services which are controlled by third parties where their implementation of web service quality is proper and uncontrollable and when time passes web service may have more bugs and input errors. N. Laranjeiro et.al [1] describes the web services description file (defined using WSDL) is parsed initially and mutation operators are applied to it resulting in several mutated documents that will be used to test the service. In spite of the effort set on this approach, these parameter mutation operators are very limited and consist basically in switching, adding, deleting elements, or setting complex types to null. Generation of representative workloads, capable of exercising the web service code in a complete way, especially in the case of random workload generation approaches are most important key aspects for robustness testing of web services.

V. PROPOSED 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 servers. Get all the information about the web services thoroughly by getting all the list of parameters, operations, data types and also parameter domain. This will create workload and execute the workload by choosing one or more generating strategies of workload. Also create a service workload. After executing the created workload it will be verified by the service answers. After measuring the workload coverage and if it is found unacceptable percentage is achieved, can generate service workload again and again or reconfigure or change the workload generation strategy. Get if any information of existing external service calls of domain exists. This will identify the issues of web services by running a set of robustness tests.
If needed can go to previous steps for example when there is any need to redefine parameter's domain. After fixing the robustness of web service will verify the behaviour of service to identify potential deviation.

VI. PROPOSED ARCHITECTURE

A set of tests is to be conducted to identify the robustness problem on web services inputs, bearing in mind both external responses to web service and call parameters. Fault Adder tool apply combinations of exceptional and acceptable inputs values for robustness testing. In Fig.1 a design of fault addition procedure includes relation between a simple web service, a client, and the fault adder tool.

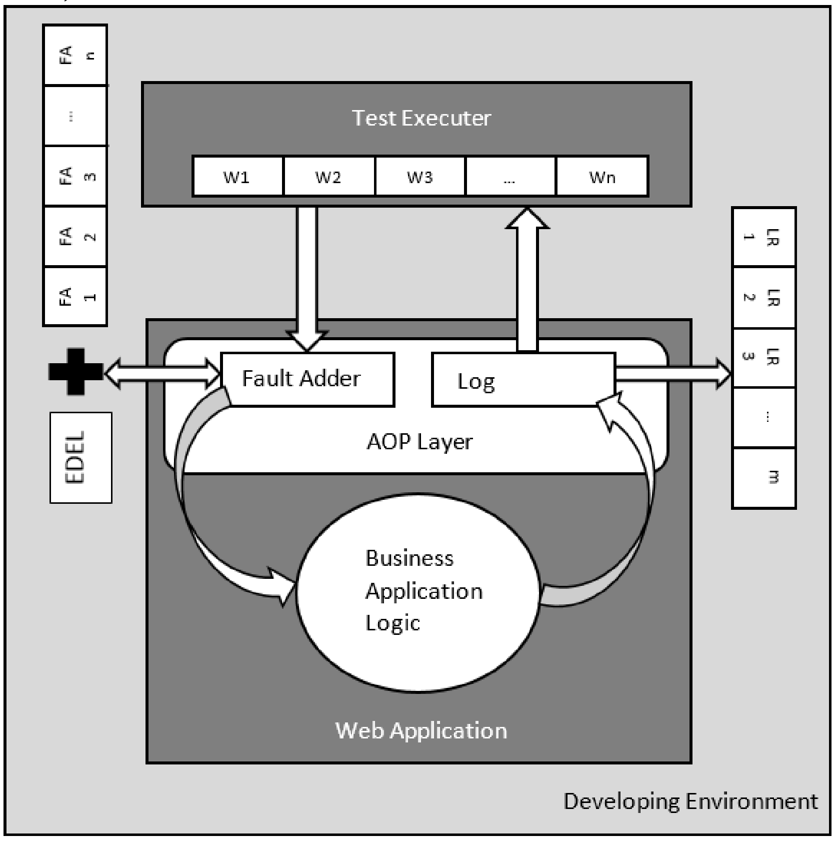

Fig. 1 Fault Adding Procedure

VII. SCOPE OF WORK

It's an approach to improve robustness of web service applications. Currently, there are many web service applications which provide online services for example, be an Internet service provider, email provider, news provider (press), entertainment provider (music, movies), search, e-shopping site (online stores), e-finance or e-banking site, e-health site, e-government site. These services are extremely huge and there is an need to validate, test application thoroughly, handle exceptional inputs by coming from malicious client or servers, validate data generated, provide upmost good quality of service to these web service providers' clients with minimal or no bugs or errors in the web applications.

VIII. CONCLUSION

The work proposed by N. Laranjeiro et.al [1] the improvement in robustness testing 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 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.

The work proposed by N. Laranjeiro et.al [1] an approach to create a robustness web services and online web applications. 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 servers.

REFERENCES

[1] Nuno Laranjeiro, Marco Vieira, Henrique Madeira, "A Technique for Deploying Robust Web Services," *IEEE Trans. On Service Computing,* January-March 2014.

[2] A. Avizienis, "The Methodology of N-Version Programming," Software Fault Tolerance, *University of California, Los Angeles and Vytautas Magnus University, Kaunas, Lithuania,* pp. 23-46, 1995.

[3] M.D. Barros, J. Shiau, C. Shang, K. Gidewall, H. Shi, and J. Forsmann, "Web Services Wind Tunnel: On Performance Testing Large-Scale Stateful Web Services," Proc. IEEE/IFIP Int'l Conf. Dependable Systems and Networks (DSN '08), pp. 612-617, June 2008.

[4] M. Kalyanakrishnam, Z. Kalbarczyk, and R. Iyer, "Failure Data Analysis of a LAN of Windows NT Based Computers," *Proc. Symp. Reliable Distributed Database Systems, pp. 178187, 1999.*

[5] P. Koopman and J. DeVale, "Comparing the Robustness of POSIX Operating Systems," *Proc. 29th Ann. Int'l Symp. Fault-Tolerant Computing, 1999.*

[6] M. Vieira, N. Laranjeiro, and H. Madeira, "Assessing Robustness of Web-Services Infrastructures," *Proc. IEEE/IFIP 37th Ann. Int'l Conf. Dependable Systems and Networks, pp. 131-136, 2007*

[7] I. Lee and R.K. Iyer, "Software Dependability in the Tandem GUARDIAN System," *IEEE Trans. Software Eng.,* vol. 21, no. 5, pp. 455-467, May 1995.

[8] M. Rodrı´guez, 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.