A Tool for Assessing Fault Tolerance Mechanisms applied to Web Service applications

Khaled Farj, Neil A. Speirs

A Tool for Assessing Fault Tolerance Mechanisms applied to Web Service applications

Khaled Farj and Neil A. Speirs
School of Computing Science, University of Newcastle upon Tyne
Newcastle upon Tyne, NE1 7RU, UK.
{k.a.s.farj, neil.speirs}@ncl.ac.uk

Abstract

Testing Fault Tolerance Mechanisms (FTM's) is crucial for the development of today's Web Service applications. In this work, we propose a methodology for assessing the efficacy of FTMs applied to Web services applications distributed over the Internet.

We present a tool that uses application level fault injection techniques to inject communication faults by using a network Emulator Service. The emulator also generates additional workload on the tested system in order to produce more realistic results. As well as allowing the user to generate a fault model script, the tool provides, (by analyzing WSDL documents), the ability to inject selected faults into the exchanged SOAP messages.

The tool can be used to test either a single Web Service or to test composed services without any modification to the system under test.

Keywords*:* Web Services, Fault Injection, SOAP, SWIFI, Network Emulator.

1. Introduction and Related Work

Web services are becoming progressively more important in forming the backbone of the modern Internet. Therefore testing the reliability and fault tolerance of web services has become an active research area. Fault injection is a well-proven method of assessing the reliability of a system [1]. Although much work has been done in testing single web services, there appears still more research to be carried out on composed services.

There are many Software Fault Injection tools testing software systems and their reliabilities. In [2] a tool is developed for generating and validating test cases. Tools start from the WSDL schema types and introduce some operator to generate a request with random data and a test script that manipulates the request parameters. In [3] a technique for testing Web Services using mutation analysis is proposed. A mutant WSDL document is generated by applying mutant operators to the original WSDL document. A test tool called WSDLTest [4] tool generates Web service requests from the WSDL schemas and tunes them in accordance with the pre-conditions written by the tester and verifies the response against the post-conditions offline. In [5] a testing tool is proposed based on some rules defined in XML schema or DTD. The tool modifies the value of the parameters in requests by using boundary value testing, and on interaction perturbation, using mutation analysis. Another tool in [6] introduces a framework intercepting and perturbing SOAP messages by injecting faults by corrupting the encoding schema address, dropping messages, and inserting random text in the SOAP Body. The work described in [7] helps service requesters create test cases so as to select suitable and correct Web services from public registries. It proposes a method where faults are injected into SOAP messages to test boundaries of the parameters, as specified in the WSDL document. WS-FIT tools [8] inject faults by modifying SOAP messages using scripts. The function parameters are modified by using the value boundaries specified by the tester.

A common characteristic of previous work is that their focus is only on testing single services in isolation; furthermore, most of their focus is on only injecting faults by modifying the SOAP message, since they do not emulate additional workload in the tested system which could give rise to different results.

This paper sets out an approach method and fault model for testing the fault tolerance of either a single Web Service or composed service, without any modification to the system being tested and generating background workload for the system.

2. Tool Architecture and Roles

Our tool consists of three main components as follows:

– *The Fault Injection Interceptor (FII)* is a web service application which has the capability to be a proxy Web Service to one or more Web services of the system under test. The FII role depends upon where it is deployed:

At the client side, its role is generating a proxy WSDL from the actual Web Service WSDL needed to be called by client. As a result all client requests are processed by the *FII*. Thereafter, the *FII* sends the request to its internal subcomponent, the *Fault Injection Controller (FIC)* to inject faults. Then the request is sent to another FII that is deployed on the site where the actual Web Service is running. When the client side *FII* receives a response from the web service it forwards it to the client.

At the actual web service side, the *FII* Web Service role is a little different. Request messages received from the *FII*, deployed at client side, are forwarded to the actual web service by the *FII*. When the response is received, it is redirected to the internal *FIC* for fault injection, and then the response, if any, is sent back to the *FII* deployed at the client site.

In the case of composed services, where the service has to act as both a service and client in the same system, a single *FII* can

perform both of the roles explained above. Using this way of intercepting messages, no modification is made to the system under test.

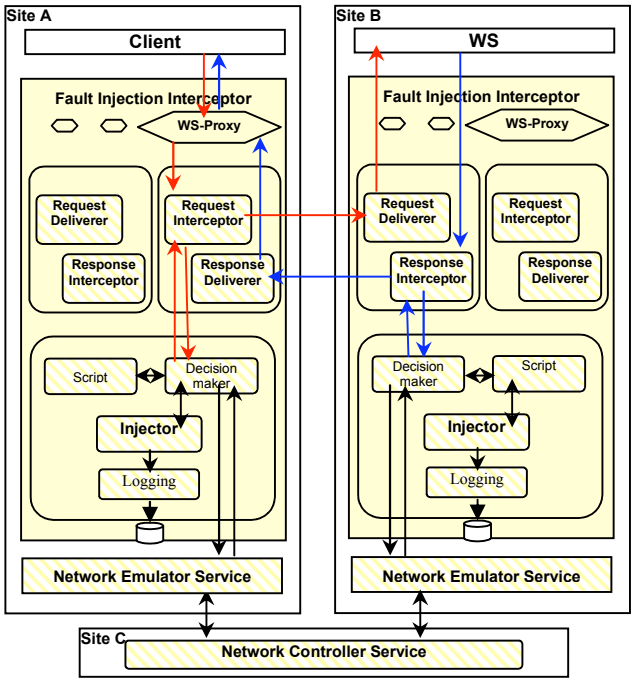

Figure1. Tool Architecture

– *Fault Injection Controller (FIC):* this is a java application inside the *FII* which is regarded as the main component of the tool. It is responsible for controlling the tool and injecting the proper faults into the messages. Faults are injected into the SOAP message based upon decisions coming from two other components of the tool – the *Network Emulation Service (NES)* and the *Script Fault Model (SFM)*. These two components can either be turned on or off at the choice of the user. The SFM is a java script program written by the user. The function parameters may be modified by using the value boundaries specified by the tester. When both *SFM* and *NES* are active, the *SFM* decision can only be applied if the decision from *NES* is not to drop or corrupt the message. The *FIC* gives network faults higher priority. The *FIC* also logs SOAP messages to be analyzed offline. The message, if it has not been dropped, is sent back to the interceptor to complete its journey to the corresponding *FII*.

– *Network Emulator Service (NES)* is an extended version of A Wide Area Network Emulator for CORBA Applications [9], which gives the applications the sense of running over a local or wide area network. It gives the applications the sense that there are other -synthetic- applications running at the same time and sharing the networking resources. In addition, it provides the ability to inject network faults (loss, delay, corruption, reordering, etc.). This work has been modified so that only Web Service technology is used. All the generated workload traffic and the faults injected use SOAP messages. The system is deployed and exposed as composed web services. The NES consists of one centralised *Network Controller Service* (*NCS*), controlling the emulated network and a set of *NES's* deployed at each node in the system which emulate the nodes of the targeted network. The *NCS* and every NES communicate with each other by exchanging SOAP messages and also communicate with the *FIC* using SOAP messages as required.

3. Conclusion and Future Work

We have built a tool that can inject faults into any Web Service application without touching the code of the application. There is great flexibility in the number and type of fault that can be injected and, furthermore, we can control the network that is used and can add background traffic. The proposed system will be used to detect problems in existing systems and also to produce metrics, based on a logging mechanism, for measuring the efficacy of Fault Tolerant Mechanisms in Web Services systems.

We are currently testing the tool with some simple TMR based Web applications to check that the enhancement in reliability of the systems makes its deployment worthwhile. A prototype version of the tool may become available for academic use in the near future.

4. References

[1] M.-C. Hsueh, T. K. Tsai, and R. K. Iyer, Fault Injection Techniques and Tools. IEEE Computer, vol. 30, pp 75-82, 1997.

[2] de Almeida, L.F. and S.R. Vergilio. *Exploring Perturbation Based Testing for Web Services*. in *Web Services, 2006. ICWS '06. International Conference on*. 2006.

[3] Siblini, R. and N. Mansour. *Testing Web services*. in *Computer Systems and Applications, 2005. The 3rd ACS/IEEE International Conference on*. 2005.

[4] Sneed, H.M. and H. Shihong. *WSDLTest - A Tool for Testing Web Services*. in *Web Site Evolution, 2006. WSE '06. Eighth IEEE International Symposium on*. 2006.

[5] Jeff, O. and X. Wuzhi, *Generating test cases for web services using data perturbation.* SIGSOFT Softw. Eng. Notes, 2004. 29(5): p. 1-10.

[6] Looker, N. and X. Jie. *Assessing the Dependability of SOAP RPC-Based Web Services by Fault Injection*. in *Object-Oriented Real-Time Dependable Systems, 2003. WORDS 2003 Fall. The Ninth IEEE International Workshop on*. 2003.

[7] Zhang, J. and R.G. Qiu. *Fault injection-based test case generation for SOA-oriented software*. in *2006 IEEE International Conference on Service Operations and Logistics, and Informatics, SOLI 2006*. 2006.

[8] Looker, N., M. Munro, and J. Xu. *WS-FIT: A tool for dependability analysis of web services*. in *Proceedings - International Computer Software and Applications Conference*. 2004. Hong Kong, China.

[9] Mohammad Alsaeed , Neil A. Speirs, "A Wide Area Network Emulator for CORBA Applications", Proceedings of the 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing, p.359-364, May 07-09, 2007.