Journal of Computer Science 10 (4): 640-646, 2014
ISSN: 1549-3636

doi:10.3844/jcssp.2014.640.646 Published Online 10 (4) 2014 (http://www.thescipub.com/jcs.toc)

A FRAMEWORK TO MAGNITUDE THE PERFORMANCE AND BEHAVIOR OF WEB SERVICES USING ONTOLOGY

N. Danapaquiame, E. Ilavarasan and Neeraj Kumar

Department of Computer Science and Engineering,
Pondicherry Engineering College, Pondicherry-605014, India

Received 2013-06-26; Revised 2013-07-25; Accepted 2013-12-21

ABSTRACT

A web service is a software interface that describes a collection of operations that can be accessed over the network through standardized XML messaging. Web services in different domains are diverse in implementation techniques thus requiring us to meet a wide range of test requirements. Testing services and service centric system poses new challenges to testing approaches. Several web services testing were developed to address these new challenges. Without testing, the web service is likely to suffer from various issues in terms of speed, scalability and stability. In this study, a generalized ontology model is used to measure the non-functional testable parameters and find the dependencies between them. The web services are got from the service providers and various testing is being executed over these services. Web service testing is performed to ensure that their functionality is in accordance with the Service Level Agreement (SLA). Metrics for each dependency will be formulated and invoked during every test and thus the test results which contain all the results about each service will be generated using Web Application Performance Tool (WAPT).

Keywords: SLA, WAPT and Ontology

1. INTRODUCTION

Internet is a world wide web which is the main source for the web services. A web service is a system of communication between two electronic devices over the web (internet). Millions of web services are published across the internet which can be made use of, according to the requirements of the consumers. These services might be available as WSDL files or sometimes the services might be available directly. The growing popularity of web services can be ascribed to a movement towards Service-Oriented Architecture (SOA).

In our proposed framework, the web services to be tested are handed over to the WSTM who is a third party, for testing. As per the SLA profile, the service consumers provide the test requirements to the WSTM. The WSTM has access to the service registry, WSDL Set, QOS repository, OWL Set and Audit Log to perform the functionality and non-functional testing of the web services.

Under functionality testing, input-output test and dependency tests are performed using XML DOM and ontology. Input-output testing is used to check if the output data types and output data values of the web service are in accordance with the XML and WSDL files of the web service. Dependency testing under functionality testing is performed by creating an ontology model and getting the parsed output in C#.net. Under the non-functional testing, each of the web services are monitored using performance testing tools and the results are tabulated. An ontology model is created for non-functional testing comprising of a few testable QOS parameters. The dependencies between these parameters are mapped onto the ontology and the deviations of the service performance are computed. A final test report consisting of test and analysis results will be given back to the test requestor, based on which he can decide to consume the service or not.

2. RELATED WORKS

Hong and Zhang (2012); Yusof *et al*. (2010); Kannammal *et al*. (2006); Rathore and Suman (2011); Palanikkumar and Kousalya (2012); Vasanthi and

Corresponding Author: N. Danapaquiame, Department of Computer Science and Engineering, Pondicherry Engineering College, Pondicherry-605014, India Tel: 9629124512

Wahidabanu (2012); Salva (2011) and Nabiollahi *et al*. (2011) presents a archetype execution of the framework in semantic WS and exhibits the feasibility of the framework by running examples of building a testing tool as a test service, embryonic a service for test executions of a WS and composing contemporary test services for more complicated testing chores. Experimental evaluation of the context has also demonstrated its scalability.

But our system performs all levels of testing using ontology based on the Protégé tool ontograf can be generated based on the parameters using ontograf tool.

3. PROPOSED SYSTEM

The study benevolences a prototype implementation of the framework in semantic WS and demonstrates the feasibility of the framework by running examples of building a testing tool as a test service, developing a service for test executions of a WS and composing existing test services for more complicated testing tasks. Experimental evaluation of the framework has also demonstrated its scalability.

In our framework, there exist three major roles namely service provider, service consumer and Web Service Test Manager (WSTM). The service provider gives its entire WS to the WSTM for testing. As per the SLA Profile, the service consumers provide the test requirements to the WSTM. The WS TM has access to the service registry, WSDL set, QoS repository, OWL Set and Audit Log to perform the functionality and non-functional testing of the web services.

Under functionality testing, there is an I/O test and a dependency test. For I/O testing, the input and output data types and values are retrieved from the WSDL and XML files of the respective web services and tested for their correctness.

In dependency testing, an ontology model is created for their entire system and the dependencies between them are mapped using an ontology mapping engine. The entity Level, operational level and attribute level dependencies are tested using the OwlDotNetApi.

Under the non-functional testing, an ontology model is created comprising of a few testable QoS parameters which are present in the QoS repository. The various relationships between the QoS parameters are identified. Using Tools such as WAPT, application manager, the QoS parameters are measured for each service and stored for future reference. A few metrics are defined and calculations are performed for each service which will help in identifying the deviations in the system performance under various conditions.

A final test report consisting of test and analysis results will be given back to the test requestor.

3.1. Functional Testing

3.1.1. Architecture Diagram

Web services with incorrect responses can lead to problems. Web service functional testing ensures that the web service is functionally correct.

Automated web service functional testing involves carrying set of tasks automatically and comparing the result of same with the expected output and ability to repeat same set of tasks multiple times with different data input and same level of accuracy. Implementing functional test for web service early in the software development cycle speeds up development improves quality and reduces risks towards the end of the cycle. We propose a framework to test the functionality of the web service. Under functional testing, there are two phases namely: Input-Output testing and dependency testing. The architecture diagram for the testing of web services is shown in the **Fig. 1**.

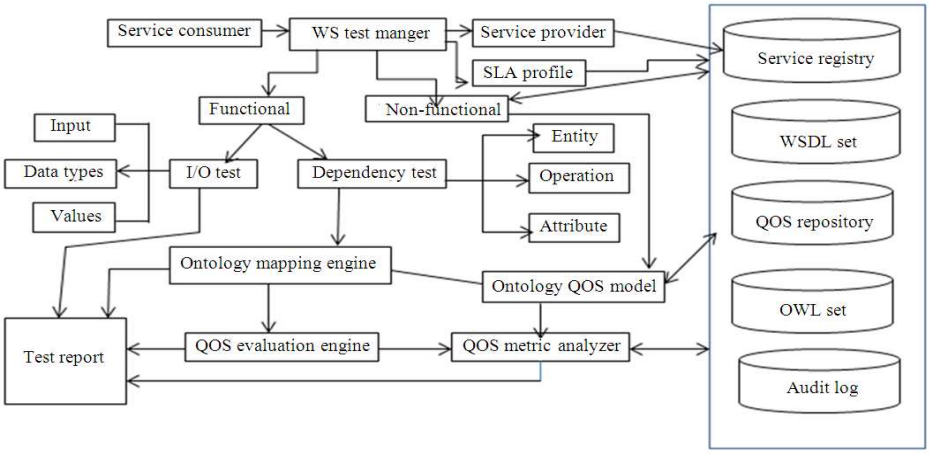

Fig 1. Architecture diagram

The services are given by the service consumer and it is send to WS test Manager. WS Test Manager is divided into functional and non-functional testing. Further functional testing is divided into I/O test and Dependency Test. I/O test performs the input, data type and values from the services. Test report is generated with the help of I/O test and ontology mapping, QOS evaluation and QOS metric analyzer. WS test manager communicates with the service provider and SLA profile. Service provider will interact with the service registry, WSDL set, OWL set and Audit Log. Dependency testing performs entity, operation and attribute.

4. INPUT OUTPUT TESTING

Figure 2 shows the input output testing of the web services is done to check whether the web service is working properly according to its functionality. First the WSDL file of each of the web services is taken. Then the XML files of the corresponding web services can be taken by running the web service and those files can be stored in a separate XML repository. Now to perform the input output testing the service name and the service location is retrieved from the WSDL files of each of services. Since the WSDL files are in tree format the contents can't be retrieved in an ordinary manner. They can retrieve using XML DOM and the retrieved contents are now stored in the database. Then using the service name the XML files of the corresponding service is opened from the XML repository. Form this file the data type of the output and the output value is been retrieved using XML DOM and stored in the database. Now we can check whether the service is working properly by checking whether the output data type is correct and whether the output value is the exact expected output value. We can also check whether the value lies between the threshold of the service or not. This can be shown in the **Fig. 2**.

4.1. Dependency Testing

For checking the dependencies, we create respective ontology for our domain using Protégé 4.1. Using OwlDotNetApi, we retrieve the sub-classes, relationships, entities, instances, disjoint classes and sibling classes present in the particular owl file. Parsing of the OWL file is performed to identify the various dependencies between the entities, operations and the attributes. The main idea behind finding the dependencies of the web service is to explore the dependencies between each of the services and giving a clear idea about the usage of those services in the domain. For example in the airline domain if a person books a ticket and if he wants to book a hotel then he can book the hotel only if he has a valid ticket number. Now while booking the hotel the ticket number of this service is dependent with the ticket number of the airline service is shown in the **Fig. 4**. So such kinds of dependencies can be explored using the ontology model. It creates an OWL file and for retrieving contents form the OWL file the OWL parser is used.

4.2. Ontology Creation Using Protégé

Protégé is software used to create ontologies for various domains. The main class is thing for the entire domain. Under the main class the sub classes can be included. Two ontologies are created one for the main domain to perform dependency testing and the other is for the non functional testing which contains the testable parameters and also their dependencies are shown. Once the ontology has been created an ontograf is generated which is the pictorial representation of the corresponding ontologies.

4.3. Non Functional Testing

For non-functional testing, generalized ontology model for all the testable quality of service parameters and measure parameters such as average response time, bandwidth, processing time, throughput, successful hits and failed hits has been used. So far there exist models which only measure each parameter individually. Here the parameters are measured individually and also find the inter-dependencies between them. Various tools like WAPT, Application Manager and Soap Sonar has been used to measure the parameters at several instances using which the deviations in the performance of the system can be determined.

More web services have been created accordingly to our domain and functionality testing (input and output testing) has been performed accordingly for each of the web services created. An ontology model has been created for the whole system specifying the various relationships between the entities, attributes and operations using Protégé. The dependencies between the various entities, attributes and operations have been identified and have been mapped onto the ontology model. For Example, ticket booking operation under airline entity and hotel booking operation under hotel entity have a ticket number as a joint attribute. Using .net string builder class, the dependency relationships has been extracted present in the ontology model as specified above and displayed them, which constitutes the dependency testing through **Fig. 5**.

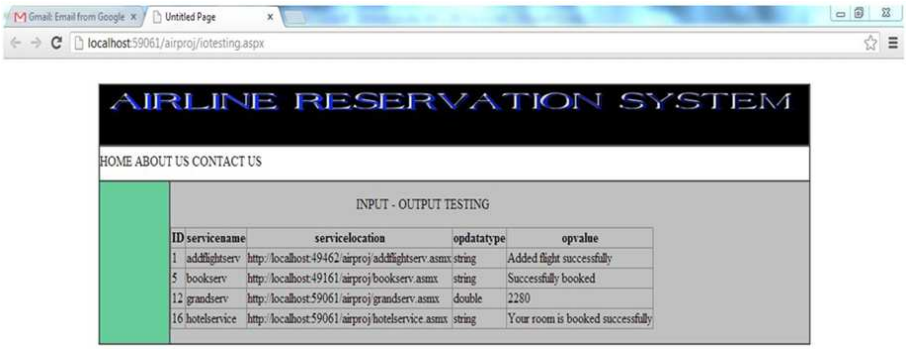

Fig. 2. Input output testing

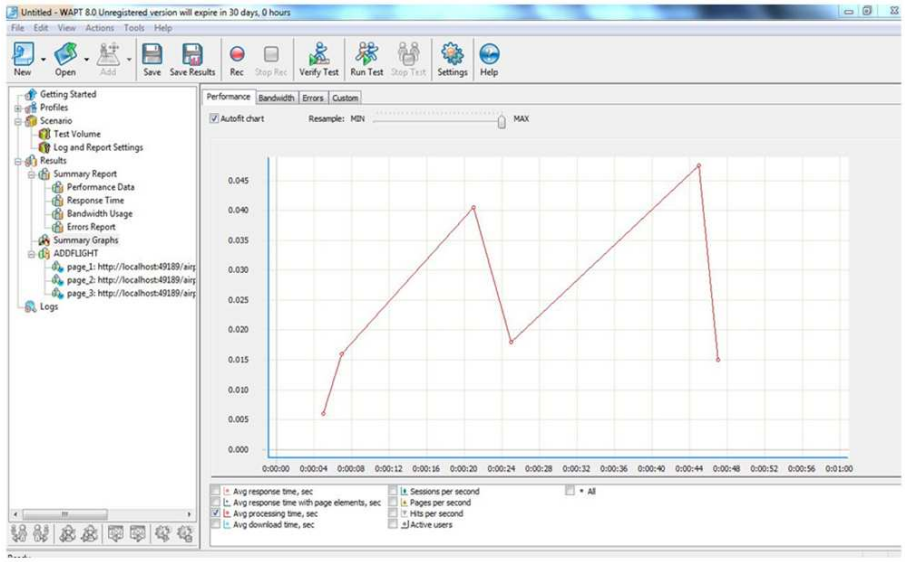

Fig. 3. Graph for response time

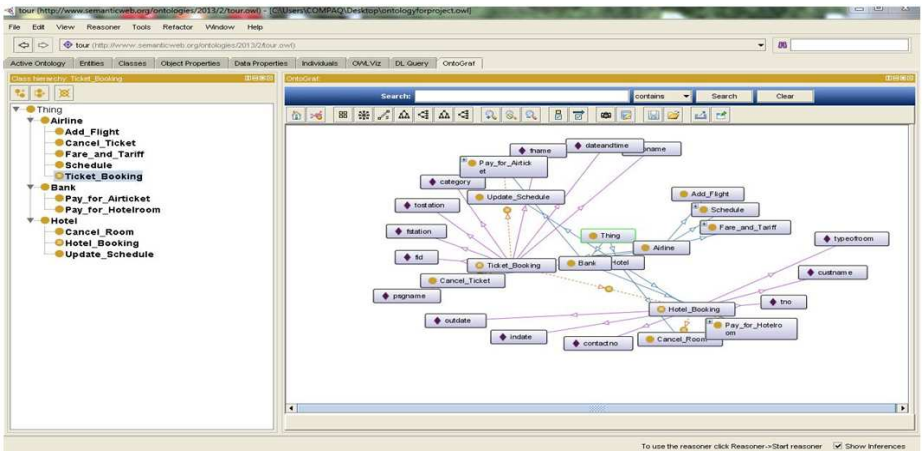

Fig. 4. Ontograf for dependency testing

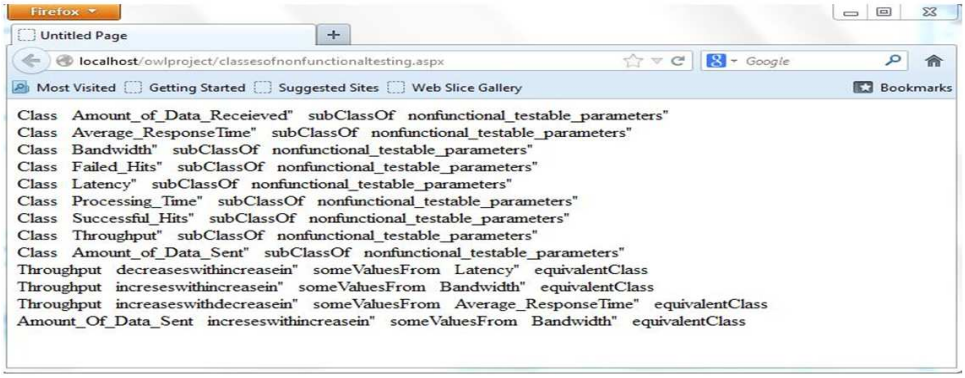

Fig. 5. Parsed output of the dependency testing of non-functional parameters

Under non-functional testing has been performed for all the web services created using WAPT testing tool. Each of the web services has been tested under various instances by varying the monitoring period and several instance values of the tests performed are stored in a database and the test results are displayed using a grid view. Based on the test results obtained the relationships between the non functional parameters specified under the ontology have been identified. These relationships have been mapped into the ontology model created for non-functional testing. The Response time of the web services is shown if **Fig. 3** is specified under SLA agreement is taken into consideration and is compared with response times of several instances for each web service. The Deviation in the response time for each instance of the web services is calculated and projected to the service consumer based on which he can decide whether to purchase the corresponding web service or not. The Latency, Bandwidth and Throughput of the web services are also calculated in the same manner and all these results are projected to the service consumers for reference.

5. EXPERIMENTAL RESULTS FOR CASE STUDY

The work is to provide the results of functional and non-functional testing of the requested web services to the client. This can be brought about by creating a website. We make use of Visual Studio 2008 for this purpose. This study need to have a database comprising of the values of tested functional and non functional parameters such as service name service location, output data type and output value, response time, processing time, amount of data sent, total bytes received, successful hits and failed hits for several instances of different web services invoked by different users. For this purpose we again make use of visual studio for web service creation and also XML DOM for parsing WSDL and XML files. For monitoring these web services, WAPT and Application Manager Tools have been used.

5.1. Web Service Creation

The first module of our framework is to create web services and test their functionality. The various web services were created using visual studio. Some of the web services created is:

- User Registration service, Adding Flight service, Ticket Booking service, Fare Calculation service
- Hotel Booking service, Bank service, Cancel Ticket service, Update Flight Schedule service

5.2. Input-Output Testing

5.2.1. Response Time Graph Using WAPT

5.2.1.1. Ontograf for Dependency Testing

The above **Fig. 5** projects the parsed output comprising of the various dependencies existing between the various non-functional parameters considered for our domain.

The above figure shows the final output which shows the overall performance of the web service which is projected to the client:

Latency	=	Response Time-Processing Time
LT	=	RT-PT
Bandwidth	=	Amount of data sent/Number of users
BW	=	ADT/Ui
Throughput	=	Number of users/(Average Response time+Processing Time)
Tpt	=	No.Ui / (AvRT+PT)
Robustness	=	Number of error messages/Total number request messages
RB	=	No.ErrMsgs/Tot. ReqMsg

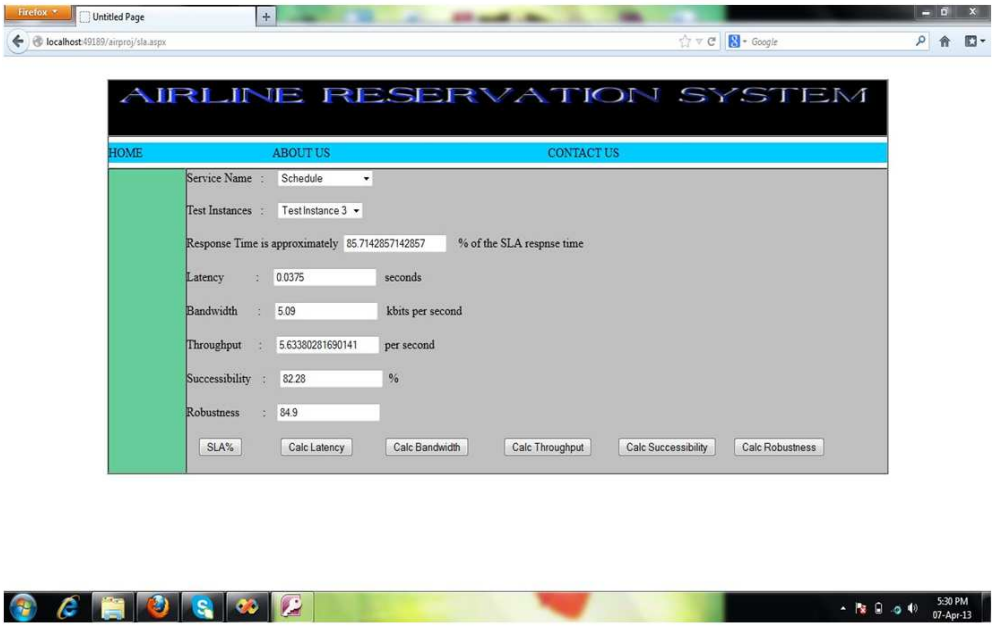

Fig. 6. Performance measure of the requested web service

Successibility = Number of response messages/number of request messages
Su = No.Resmsg/no.reqmsg
Availability = 1-(Downtime/Measurement Time)
Av = 1-(Dt-Mt)

Figure 6 shows the calculation of the Latency, Bandwidth, Throughout, Robustness, Successibility and Availability

6. CONCLUSION

The internet is a vast area where millions of web services are available according to the consumers' requirements. The shift towards Service-Oriented Architecture (SOA) can be held responsible for the ever-increasing popularity of these web services. There are a lot of challenges while testing web services especially while integrating web services which are owned by different vendors. Several strategies have proposed to test the web services. But each of them has drawbacks of their own.

Thus we have proposed a testing framework which acts as an intermediary or a third party tester between the service provider and the service consumer. Functional testing comprising of input output testing and dependency testing using ontology has been performed which shows the behavior of the web services. Under non-functional testing, the values of QOS parameters are measured for each web service and the deviations of the system's performance under various conditions have been found. An Ontology model is also created for non-functional testing which helps us to identify the dependencies between the evaluated parameters. Finally the test and analysis results are given to the consumer, based on which they can decide to purchase that web service or try out any other web service.

Our work makes use of ontology based testing framework for functional and non-functional testing of web services. We have taken into consideration only the testable QoS parameters such as response time, error rate, amount of data sent per second, amount of data received per second, average processing time, latency bandwidth, successful hits and the like. Few other testing types have not been explored much. As a future enhancement to our work testing types such as regression testing, stress testing and recovery testing might be performed over the web services which could produce a better analysis of their performance thus making it preponderant for the consumer's perusal.

7. REFERENCES

Hong, Z. and Y. Zhang, 2012. Collaborative testing of web services. IEEE Tran. Services Comput., 5: 116-130. DOI: 10.1109/TSC.2010.54

Kannammal, A., V. Ramachandran and N.Ch.S.N. Iyengar, 2006. An enhanced secure and scalable model for enterprise applications using automated monitoring. J. Comput. Sci., 2: 589-594. DOI: 10.3844/jcssp.2006.589.594

Nabiollahi, A., R.A. Alias and S. Sahibuddin, 2011. Involvement of service knowledge management system in integration of ITIL v3 and enterprise architecture. Am. J. Econ. Bus. Admin., 3: 165-170. DOI: 10.3844/ajebasp.2011.165.170

Palanikkumar, D. and G. Kousalya, 2012. An evolutionary algorithmic approach based optimal web service selection for composition with quality of service. J. Comput. Sci., 8: 573-578. DOI: 10.3844/jcssp.2012.573.578

Rathore, M. and U. Suman, 2011. A quality of service broker based process model for dynamic web service composition. J. Comput. Sci., 7: 1267-1274. DOI: 10.3844/jcssp.2011.1267.1274

Salva, S., 2011. An approach for testing web service compositions when internal messages are unobservable. Int. J. Elect. Bus. Manage., 9: 334-345.

Vasanthi, R. and R.S.D. Wahidabanu, 2012. Adaptable service component interface framework in pervasive computing. J. Comput. Sci., 8: 859-863. DOI: 10.3844/jcssp.2012.859.863

Yusof, M.M., M. Omar and Z. Shukur, 2010. A disruption-tolerant model for building a mobile application using web service. J. Comput. Sci., 6: 1430-1437. DOI: 10.3844/jcssp.2010.1430.1437