ISSN 2277-3061

TFCWS: Testing Framework for composite web services

Deepali Diwase, Pujashree Vidap
Department of computer engineering Pune Institute of Computer Technology, Pune, India
Email: deepali.diwase@gmail.com
Department of computer engineering Pune Institute of Computer Technology, Pune, India
Email: psvidap@pict.edu

ABSTRACT

In every business domain Web Services are more popular solutions to implement the software. Composite web service can be created by combining basic web services. Many unreliable web services are deployed on the internet. Hence, testing is required to ensure reliability. Software testers have great challenges to test web services. Source code of web services is unavailable. The Testing Framework is used to test web services without knowledge of its internal structure. In this paper, we have proposed a Testing Framework for Composite Web Services (TFCWS). It generates report which shows the total number of test cases executed for each web service with pass or fail status of each test case. It calculates the throughput of web service and response time of each test case. We have used web services response times for analysis of TFCWS, Soap UI and Storm.

Keywords

Testing and debugging, Reliability, web service.

1. INTRODUCTION

A web service is a software system identified by a URI (Uniform Resource Identifiers). It is designed to support machine to machine interaction over a Network. SOAP (Simple Object Access Protocol) is used to exchange information on web services. The XML (Extensible Markup Language) is used to transmit messages and data [11]. Functionality of web services is described by using XML based WSDL (Web Services Description Language). It gives basic information of web services that is its operations and data transmitted. WSDL is becoming popular because of SOA (Service Oriented Architecture) [7]. WS (Web Service) can be either component or composite. Component web service is a basic WS does not rely on any other WSs whereas a composite or composed WS is formed by a combination of many component WSs. For example, composite web service travel agency is a combination of the following services: travel agent, hotel, car, bike, entertainment and billing [4]. Composite web services testing is a difficult topic in web services testing. Testing is required to ensure quality of service and evaluate reliability, performance and functional correctness [3]. Unit testing, Integration testing, Regression testing are important for composite web services testing. Bugs in isolated web service are identified by Unit testing. Composite web service with its partner services can be identified by using integration testing. Regression testing is used to check whether bug fixes are successful or not. It includes external interfaces testing and testing of basic services that composite service is relied on [8].

There are two types of web service composition

1. Orchestration: It contains central architecture in which flow of partner services is controlled by the single main process.

2. Choreography: In this all partner web services involved in composition interact with each other. There is no main process.

Researcher and industries have attention towards BPEL (Business Process Execution Language) which is semi-formal flow language and used for web service orchestration. Description of complex business processes involving more web services is difficult to understand and it can be error prone hence to ensure correctness there is an increase in interest to test the flow logic of BPEL processes.

Now days we can see that web services have an important role in the construction of computer applications which are remotely used by many users. Testing techniques must be used to test these web services. TFCWS is used for functional testing of web services which are composed by orchestration. It is quality assurance process and type of black box testing. Functions are tested by providing different inputs and examining outputs.

Initially Request XML with place holders is created. User's input data is mapped with these place holders and request XML with actual data is created and sent to the server. The assertion is users expected result, stored in the framework. The response obtained from server is matched with an assertion and pass or fail results of test cases are achieved. After execution of test cases, reports are generated. These reports show the user's request and response which is obtained from the server. It calculates response times of each test case and throughput of web services. Response time obtained after execution of web services are used for analysis of TFCWS, open source software testing tools SOAP UI and Storm. SOAP UI provides features which are useful to evaluate the performance of web services. The Storm provides feature by which we can test multiple web services simultaneously.

Organization of the Paper

The rest of the paper is organized as follows: Section II presents related work, Section III describes TFCWS, Section IV results and analysis of TFCWS with other tools, Section V concludes the paper.

2. RELATED WORK

W. T. Tsai, Ray Paul have proposed testing framework Coyote which is object oriented and XML based framework used to test web services rapidly. It contains two parts 1. Test master 2. Test Engine. Test Master allows testers to specify the test scenario and converts WSDL specifications into test scenarios. Web services under test are interacted by test engine. WS have distributed nature so Coyote focuses on integration testing [1]. Hai Huang, Wei-Tek Tsai have proposed an approach for WS testing and to validate automatically generated test cases. It can be done by model checking process of OWL-S (Web Ontology Language for WS) [2]. To ensure quality of services test cases have to be generated, executed monitored and analyzed at runtime. WSDL gives basic information about WS operation and data transmitted. From this WSDL test cases are generated automatically. First WSDL is parsed and transformed into structured DOM trees. There are two perspectives to generate test cases 1. Test data generation 2. Test operation generation. According to standard XML schema syntax message data types are analyzed and Test data is generated and operation dependency analysis is used to generate operation flows [3].

S. Noikajana and T. Suwannasart have proposed test case generation method for web services testing. WS contract is described by using Service Semantics Language (WSDL-S) and the Object Constraint Language (OCL). pre- and post-conditions of Web Service operation are identified by using WSDL-S, to identify these operations OCL rules are used [6]. In WS-TAXI framework WSDL file is parsed and operations, messages, data structure such useful information is extracted. It combines WS operations with data-driven test generation. By using syntax-based testing approaches WS test cases are generated [7]. The tester can find the specification of composite web services which includes process call, input data type and output data type. Bo Yang, Ji Wu have proposed regression testing method for composite web services. This method introduces symbols to identify bugs in composite web service and Test script is used to generate test case. Test script is composed of test data and test behaviour which are independent of each other [8]. T.D. Cao, R. Castanet

and P. Felix have proposed two tools for conformance testing of web services. One tool is created for online approach and second focuses for verification of timed trace with respect to a set of constraints. Testing of WS is needed to ensure quality of service. In composite service if any component WS is of poor quality it can affect to composite service so unit testing is used. In unit testing each partner service is tested separately [9]. F. Belli, M. Linschulte have presented An Event-Based Approach for composite web services testing. Test cases are generated to test the interaction of web services within composite service. Concurrent event sequence graphs cESG are used to maintain conditions in control flow and transferred data dependency. Decision tables are used for augmentation of cESG [10]. Y.Zheng, J.Zhou, P.Krause have proposed Operational semantics and test case generation for BPEL which is based on model checking. To test whether the implementation of web services conforms to the BPEL behaviour and WSDL interface models two levels of test cases are generated [12]. B.Stepien, L.Peyton , P.Xiong have proposed web application testing framework by using TTCN-3.It can define test cases at different levels of abstraction. Features of TTCN-3 include a powerful matching mechanism that allows a separation between behaviour and the conditions governing behaviour. Web applications are used to manage the information which is verified by using TTCN-3's data types and set-based operations are used [13]. H. Zhu has proposed SOA for WS testing. Test task is created by the collaboration of test services. Software Testing Ontology for WS (STOWS) is used to describe capability and tasks of test services [14]. M. Yalla & R. Shanbhag have proposed an automation framework around open source technologies. It is

used to organize test design, to generate test data and test reports. In html format results are generated which contains a test summary, detail reports and screen shots [15].

3. PROPOSED SYSTEM TFCWS

By using set theory the TFCWS is defined as follows.

S= {I, O, S, F}

Where

Input:

I = {W, D}

W is set of composite Web Services,

W = { w1, w2 , w3,…… ,wn }

D is input data to test web services functions.

Output:

O is Testing Framework with the report O ={R, T, Rp}

Where R is set of Response times,

R = { r1, r2, r3, … rn }

It measures delay in milliseconds between the moment when a request is sent and responses are received.

For operation op and WS s response time can be calculated as q (s, op).

q (s,op) = Processing Time + Transmission Time

WS provide methods to acquire processing time.

The transmission time is calculated by using Past execution of services.

T is set of Throughputs of different web services.

T = { t1, t2 , t3,…… , tn }

Throughput is the number of requests executed per unit time.

Rp is Report contains request and response in the test case of each web service.

Rp = {X1, X2}

Where

X1 is set of request XML from the user

X1={x11, x12, x13, … x1n }

X2 is Response from the server

X2 ={ x21, x22, x23,……, x2n}

Success State:

S = Successful web services testing by the framework .

Failure State:

F = Failure of at least one scenario tests of framework.

3.2 TFCWS ARCHITECTURE

Figure 1 shows the system architecture for Testing Framework by which user will upload the WSDL of WS to be tested.

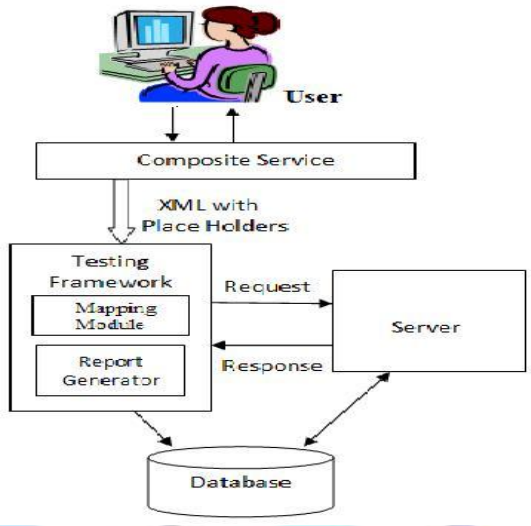

Fig 1: TFCWS Architecture

From these WSDL request XMLs with place holders are created automatically. We have taken the example of flight booking web service. The request XML with place holder is as shown below. ##From##, ##To##, ##Name## etc. are placeholders.

<soapenv: Envelope xmlns:q0="http://flightmain.bpel.com"

xmlns:soapenv="http://schemas.xmlsoap.org/soap/e

nvelope/"

xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <soapenv: Header>

</soapenv: Header> <soapenv: Body>

<q0: FlightMainRequest>

<q0: From>##From##</q0: From> <q0: To>##To##</q0: To>

<q0: Name>##Name##</q0: Name> <q0: ID>##ID##</q0: ID>

<q0: Email>##Email##</q0: Email>

<q0: Phone No>##PhoneNo##</q0: Phone No> <q0: Date>##Date##</q0: Date>

</q0: FlightMainRequest> </soapenv: Body>

</soapenv: Envelope>

The above XML file is parsed and user interface is created where user enters data to test web services. User interface created for above request XML with data validation is shown as Figure 2. Mapping module is used to map data and place holders. In this framework Graphical User Interface (GUI) is provided to the user from which user will enter data which is stored in the database. The advantage of using this GUI is to provide data validation. Data required for mapping is taken from xls sheets. xls sheets contain data which can be hard coded or obtained from properties file or database.

ISSN 2277-3061

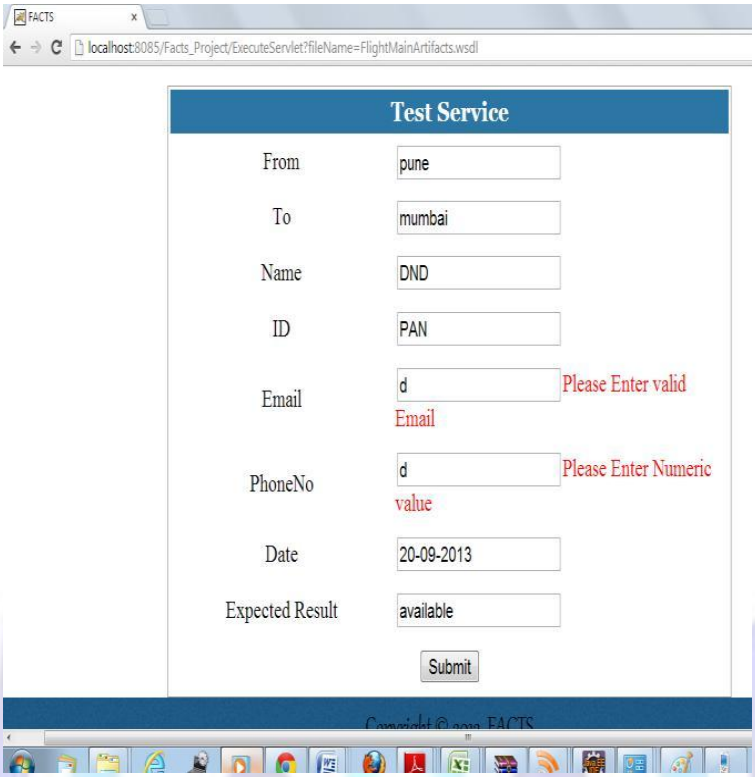

Fig2: TFCWS GUI

Placeholders in request XMLs are replaced by actual data then it becomes Request XML with actual data. We have used xls sheets to store data. While Mapping names of placeholders are matched with column names in xls sheet and placeholders are replaced with data.

`After mapping of data and place holders Request XML is represented as follow

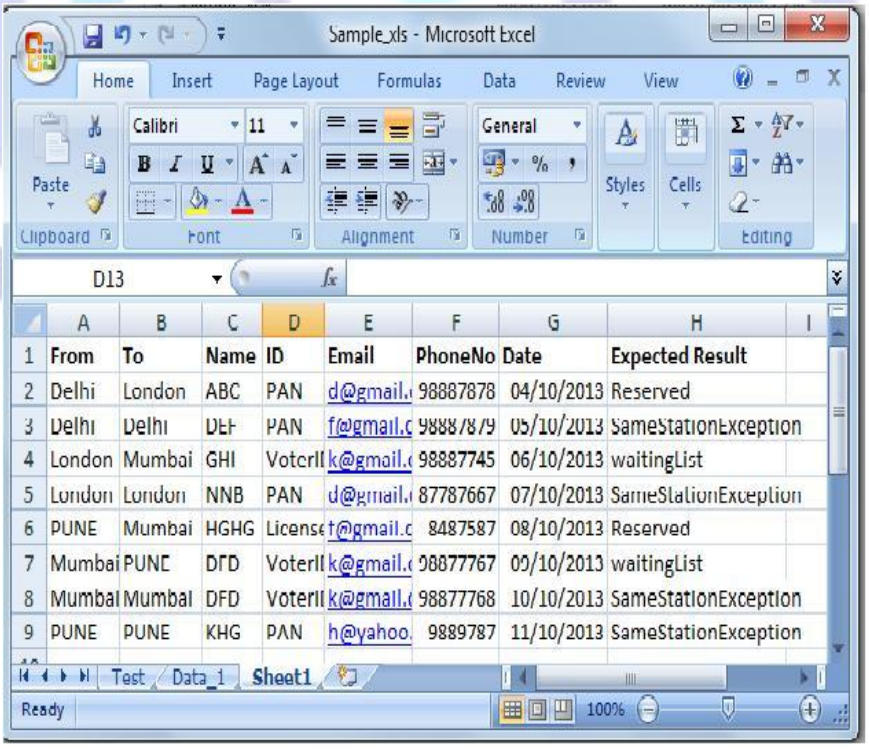

Fig3: xls sheet containing data and assertions given by the user for WS testing.

```
<soapenv: Envelope xmlns:q0="http://flightmain.bpel.com"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"

xmlns:xsd="http://www.w3.org/2001/XMLSchema"xmlns:xsi ="http://www.w3.org/2001/XMLSchema-instance">
<soapenv: Header>

</soapenv: Header> <soapenv: Body>
<q0: FlightMainRequest>
    <q0: From>Delhi</q0: From> <q0: To>London</q0: To> <q0: Name>ABC</q0:
     Name> <q0: ID>PAN</q0: ID>
      <q0:Email>d@gmail.com</q0:Email> <q0: Phone No>9876543212</q0: Phone No>
     <q0: Date>03-10-2013</q0: Date> </q0: FlightMainRequest>
</soapenv: Body> </soapenv: Envelope>
```

This XML is request XML with actual data which is sent to server and response from server is obtained. The response obtained from server is matched with assertion. The assertion is nothing but expected results given by the user which is shown in data xls sheet. The report generator module is used to generate reports in XML format. It shows the total number of test cases executed, number of pass and fail test cases with response time required to execute each request. Following Figure 4 shows the xls sheet containing reports. It will store the user's request and response obtained from the server. At any time these reports are accessible to the user. There is no need to test these test cases again.

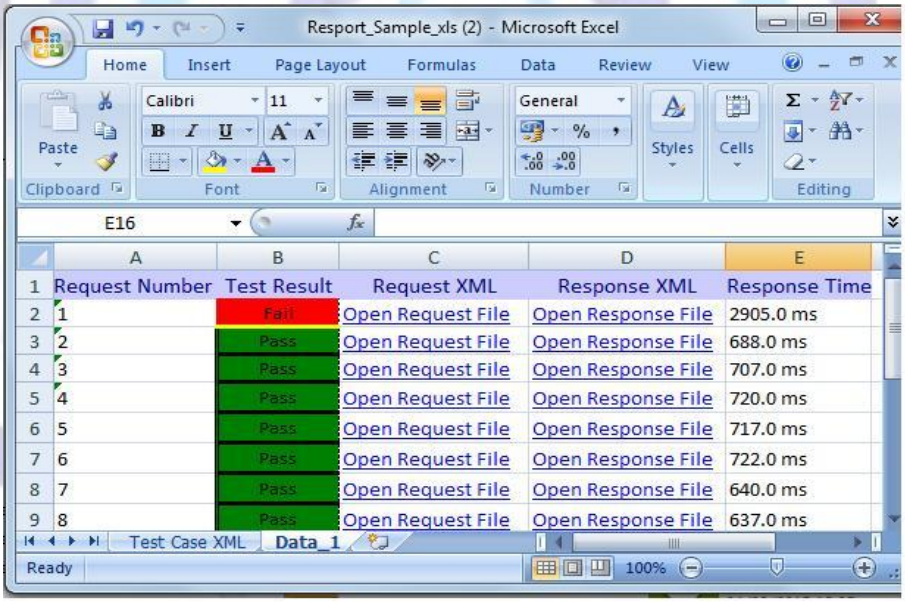

Fig 4: test cases results

While execution of web service user will upload WSDL. WS to be tested is selected by the user. From WSDL Request XML with place holders is created. Mapping module will map place holders in XML with data. In this way request xml with actual data is created and it is sent towards the server. The response obtained from server is matched with assertions given by the user. Finally the report generator is used to generate reports.

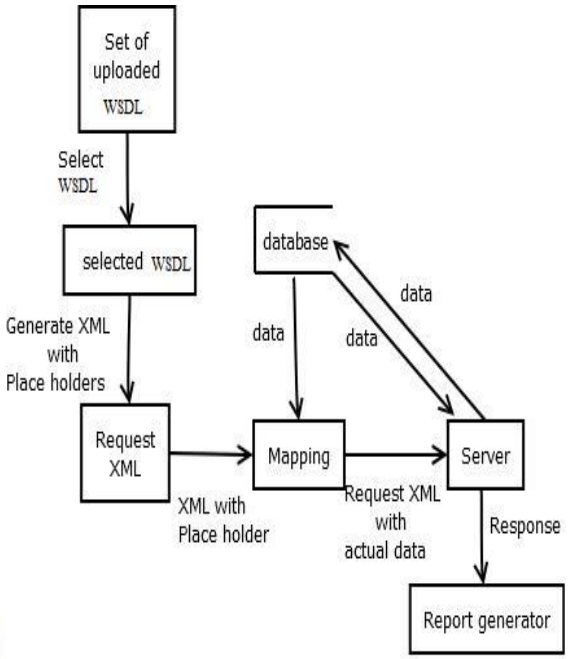

Fig 5: Execution process for proposed system

RESULTS AND DISCUSSIONS

We have developed a web services testing framework. The first part of the framework is a graphical user interface through which user will provide data to test cases. Second part is to execute test cases. We first develop several composite WS through eclipse and deployed them on the Apache ode server. WSDL of these WS are used in the framework. This framework is accessible by other PCs which are connected in LAN. In the experiments, we choose 5 composite services denoted by w1, w2, w3, w4, w5.For each web service 100 test cases are created and used for comparison of TFCWS, SOAP UI, storm .Steps for configuration of each tool includes installation of testing tool, test data collection, setting up test environment, selecting test parameters and report analysis.

We run the test cases on an Intel Core i3, 2.20 GHz processor machine with 4GB RAM, Microsoft Windows 7 Home basic. At regular interval tests were conducted to get fair and transparent results. There are many factors which affect on the performance of the internet such as traffic, users, etc.

TFCWS, SOAP UI and storm were tested by invoking sample web services. The results were collected for analysis as shown in Table1.Comparative results with testing tools are described as follow: Each tool have different internal processes to perform tasks. Tools architecture and internal processes are basic factors to compare tools in terms of response time. Response time having Minimum and maximum values at different time intervals are shown in Table 1. From Table 1 we observe that at 6.00 PM for all tools values of response time are less. This shows that Internet connection performance can be reflected in response time values. Further, Average response time for each test tool and for each web service is calculated by using above test results. Table 2 shows the average response time for each web service. It is also presented in the form of graph as shown in Figure 6.

ISSN 2277-3061

Table 1:Sample web services response time for testing tools

Testing Tools	Web Service	Response Time (ms)							
		12 PM		3 PM		6 PM		9 PM	
		Min	Max	Min	Max	Min	Max	Min	Max
TFCWS	W1	661	902	659	897	657	894	660	896
	W2	643	787	640	785	637	784	642	788
	W3	649	818	648	815	647	810	653	813
	W4	671	885	670	881	667	878	670	880
	W5	682	892	680	894	678	890	683	896
SOAP UI	W1	671	912	668	911	668	902	671	906
	W2	654	798	652	796	647	797	657	798
	W3	661	829	662	819	650	814	655	824
	W4	682	887	684	885	671	883	673	886
	W5	693	904	691	897	687	898	685	898
Storm	W1	683	923	678	921	676	914	683	918
	W2	667	807	665	809	663	807	668	809
	W3	674	841	676	819	653	817	657	829
	W4	694	896	697	889	674	886	677	890
	W5	705	921	705	899	683	902	688	889

Table 2: Sample web service average response time for testing tools

Web Service	Average Response Time in ms		
	TFCWS	SOAP UI	Storm
W1	778	789	800
W2	713	725	737
W3	732	739	746
W4	775	781	788
W5	787	794	799

From the results, we observe that TFCWS is taking less time in responding to web services as compared to other two tools. Hence the proposed system is the fastest tool in terms of response time.

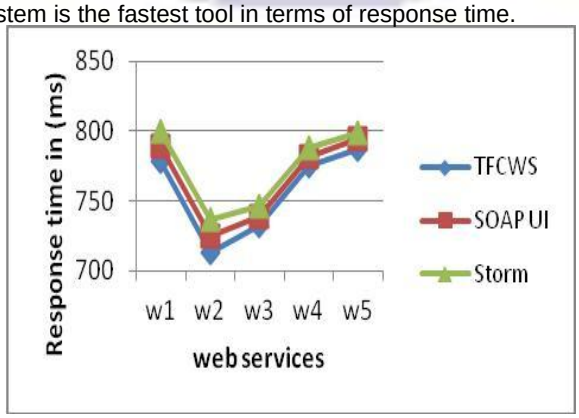

Fig 6: Sample web services Vs Response time.

Next comparison parameter is throughput. Throughput is the number of requests handled per unit time. We have calculated throughput as the number of requests handled per second.

Table 3: Throughput of sample web services for different tools.

Web Service	Average Response Time in ms		
	TFCWS	SOAP UI	Storm
W1	1.2853	1.2674	1.2500
W2	1.4025	1.3793	1.3568
W3	1.3661	1.3531	1.3404
W4	1.2903	1.2804	1.2690
W5	1.2706	1.2594	1.2515

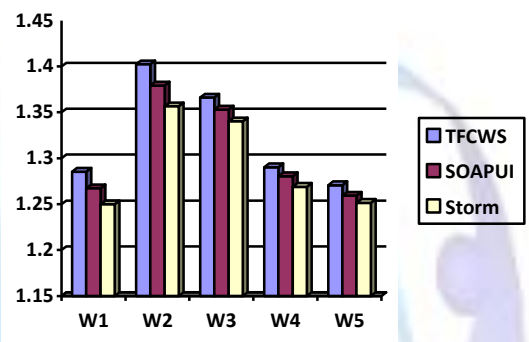

Fig 7: Sample web services Vs Throughtput

As TFCWS takes less amount of time to execute web services. It can execute a number of web services in unit time, hence throughput of the proposed system is more. Based on functionality we can compare these tools as follow.

Table 5: Technical overview of web services

Technology	TFCWS	SOAP UI	Storm
Technology Support	Web- HTTP, HTTPS SOAP Database via JDBC	Web- HTTP, HTTPS SOAP Database via JDBC JMS	SOAP
Programming Language	Java	Java	F#
Requirement	JRE 1.6+	JRE 1.6+	. NET Framework 2.0 F# 1.9.3.14
Operating System Support	Cross platform	Cross platform	Microsoft windows

Table 5 shows a comparison of testing tools based on technology and platform.

Table 4: Functionality overview of web Services testing tools

Functionality	TFCWS	SOAP UI	Storm
GUI	YES	YES	YES
Data Validation at GUI	YES	NO	NO
Data access from the database or properties file	YES	NO	NO
Report generation	YES	YES	YES
Works as web application	YES	NO	NO
Testing of multiple web services simultaneously	YES	NO	YES

The above table shows that all these tools have GUI. TFCWS and Storm have GUI which is easy to use. SOAP UI has attractive GUI which provides multiple testing functionalities. TFCWS provides data validation at GUI in which directly we can give data or access it from database or properties file. These tools generate reports of testing to which user can refer to any time.

CONCLUSION

In this paper, we have proposed TFCWS testing framework for composite web services. By using this framework we can test multiple test cases of different web services. TFCWS generates report which shows the total number of test cases executed, pass or fail status of each test case. This report is saved and the user can check it according to requirements. In this paper, a comparative study of open source tools for web service testing is presented. Response time and throughput these quality factors are used for comparison of tools. To evaluate tools sample web services and their test results are used. TFCWS have simple and user friendly interface, less response time and more throughput than SOUP UI and Storm. Throughput increased by TFCWS is 1.14% and 2.22% than SOAP UI and Storm respectively.

REFERENCES

[1] W. T. Tsai, R. Paul, W. Song, Z. Cao, “Coyote: An XML-Based Framework for Web Services Testing,”Proceedings of the 7th IEEE International Symposium on High Assurance Systems Engineering (HASE’02).

[2] H. Huang, W. T. Tsai, R. Paul, Y. Chen, “Automated Model Checking and Testing for Composite Web Services,” Proceedings of the Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC’05).

[3] X.Bai, W.Dong, W.Tsai, Y.Chen, “WSDL-Based Automatic Test Case Generation for Web Services Testing,”Proceedings of the 2005 IEEE International Workshop on Service-Oriented System Engineering (SOSE’05).

[4] M. Karam, H. Safa, H. Artail, “An Abstract Workflow-Based Framework for Testing Composed Web Services,” 1-4244-1031-2/2007 IEEE.

[5] Structural Testing Approach for Web Service Compositions,” 2008 IEEE International Symposium on Service-Oriented System Engineering.

[6] S. Noikajana, T. Suwannasart, “An Improved Test Case Generation Method for Web Service Testing from WSDL-S and OCL with Pairwise Testing Technique,” 2009 33rd Annual IEEE International Computer Software and Applications Conference.

[7] C.Bartolini, A.Bertolino, E.Marchetti,“WS-TAXI: a WSDL-based testing tool for Web Services,” 2009 International Conference on Software Testing Verification and Validation.

[8] B. Yang, J. Wu, C. Liu, L.Xu, “A Regression Testing Method for Composite Web Service,” 978- 1-4244-5316-0/2010 IEEE.

[9] T.D.Cao, R.Castanet, P.Felix, G. Morales, “Testing of Web services Tools and Experiments,” 2011 IEEE Asia Pacific Services Computing Conference.

[10] F. Belli, M. Linschulte, “Testing Composite WebServices - An Event-Based Approach,” IEEE International Conference on Software Testing Verification and Validation Workshops.

[11]A. Sharma, T.D. Hellmann, F. Maurer, “Testing Of Web Services – A Systematic Mapping,” 2012 IEEE Eighth World Congress on Services.

[12] Y. Zheng, J. Zhou, P. Krause, “An Automatic Test Case Generation Framework for Web Services,” Journal of software, vol. 2, no. 3, September 2007.

[13] B. Stepien, L. Peyton, P. Xiong, “Framework Testing of web applications using TTCN-3,” Int J Softw Tools Technol Transfer (2008).

[14] H. Zhu, Y. Zhang, “Collaborative Testing of Web Services,”IEEE Transactions on services Computing, Vol. 5, no. 1, January 2012.

[15] M. Yale, R. Shanbhag, “Building An automation Framework around open source technologies,” STC Conference 2009 .