VOL. 7, NO. 11, NOVEMBER 2012 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

UML BASED WEB SERVICE REGRESSION TESTING USING TEST CASES: A CASE STUDY

Rajani Kanta Mohanty[1], Binod Kumar Pattanayak[2] and Durga Prasad Mohapatra[3]
[1]Department of CSE, Synergy Institute of Technology, Bhubaneswar, Odisha, India
[2]Department of CSE, ITER, Siksha 'O' Anusandhan University, Bhubaneswar, Odisha, India
[3]Department of CSE, National Institute of Technology, Rourkela, Odisha, India
E-Mail: rkm.bbs@gmail.com

ABSTRACT

Web services represent the class of Service Oriented Architecture (SOA) based applications with a hugely diversified domain. Web service regression testing presents a set of challenges to the tester which need to be overcome in order to provide a reliable performance of the desired application. Code based regression testing approaches present a lot of difficulties as the tester needs to know the code which is in most cases not possible. In this paper, we address a UML based regression testing method independent of the code using test cases generated from use cases in the context of a case study.

Keywords: web service, regression testing, use case, test case.

1. INTRODUCTION

With the increasing popularity of global internet, numerous web applications have been designed in variety of areas such as communication, information distribution, parallel and distributed search across World Wide Web and many other applications. However for a web application to deliver desirably and correctly, it needs to undergo testing. Testing is the process that identifies correctness and the quality of the application. At the same time, modifications incorporated into the application during testing, must not adversely affect its basic functionality, which can be achieved with the help of regression testing. Thus, regression testing is the process of selective retesting of a system or component in order to ensure that modifications do not lead to undesired effects and that the system still conforms to its desired functionalities [1].

Unlike traditional applications, design of web applications need to take into consideration a set of quality attributes such as security, scalability, reliable transfer of information and optimal distribution of functionalities between client side and server side. Frequently changing business goals make testing of a web application more challenging. However, traditional regression testing must be adapted to conform to the characteristics, needs and nature of the web application.

Regression testing represents testing of the modified version *s'* of an original system *s* with respect to a given test set *t*. Different appropriate test cases from *t* can be applied to *s'* in different possible ways. A large spectrum of regression-testing methods is available in the literature.

In this paper, we use a UML based regression testing technique based on test cases generated from use cases for on line item purchase system, a class of web services. The rest of the paper is organized as follows. Section 2 details an extensive literature survey on web service regression testing. Web service architecture is discussed in Section 3. Section 4 covers the approach to web service modeling. UML based web service regression testing is described in Section 5. A case study for web service regression testing is elaborated in Section 6 and Section 7 concludes the paper with probable future extensions to the current work.

2. RELATED WORK

Numerous research works have been presented related to web service regression testing; a class of SOA based applications. A safe regression testing selection technique for web services based on Event Driven Graph (EDG) is proposed by authors in [1]. Authors in [2] present a regression testing method for retesting modified web applications, where only necessary test sequences are chosen to ensure the correctness of the modified application. A test case prioritization technique based on dependence analysis is proposed [3], where first of all, dependence relationship is analyzed on the basis of control and data flow information using orchestration language WS-BPEL, and then a weighted graph is constructed to perform impact analysis in order for identification of elements affected by the modification. A Control Flow Graph (CFG)-based approach for application of a safe regression test selection (RTS) technique to web services is presented in [4]. A regression testing approach for composite web services is proposed [5], which makes it possible for the tester to locate the fault in the respective service, ultimately differentiating test data from test behavior. In order to verify if the functional and nonfunctional requirements of a web service are maintained over time, a test suite, in the form of a contract between the service provider and the user of the service, can be run repeatedly [6]. A model based test case prioritization technique is presented by the authors in [7], where the test cases are ordered in such a manner that the most desired one is executed first. A safe regression technique for verification of web services in end-to-end manner is proposed in [8]. An automated regression test selection technique for web services has been presented in [9].

Coyote and XML based object-oriented testing approach is detailed in [10]. Authors in [11] present a test case generation frame work for operation sequences of semantic web services using WSDL interface and IOPEs information changes and impacts. As claimed by authors, specification of values during development of web pages and verification of the results that are displayed on the web pages can be achieved with the automated regression testing of a web service on the server side [12]. An approach for optimization of cost of regression test selection is discussed in [13]. A regression test method for web applications using automated repair of user session data is addressed, the effectiveness of which is established with experimental results [14]. A model driven approach to development of automated script testing in order to validate a web application is proposed in [15], where a meta-model is defined using UML 2.0 for the purpose. A model based approach to regression testing is discussed in [16]. An integrated and scalable regression testing method for IP based application is addressed in [17]. Authors in [18] propose a safe regression test strategy for Java web services using a code transformation approach. An extension of WSDL in order to facilitate web service regression testing is implemented [19], where the extensions are incorporated in input-output dependency, invocation sequence, hierarchical functional description and concurrent sequence specifications. In an attempt to automate web service regression testing, the WSDL file can be parsed generating SOAP requests with the corresponding values required for those requests which can be queried from different data bases [20]. In [21], authors present specific requirements, challenges and benefits in providing web service regression test as a service (RTaas).

From our intensive investigation across the spectrum of literature on web service regression testing, we could infer that, most of the approaches tend to code based testing, which becomes a tedious job, and a little emphasis is given on UML based regression testing using test cases, which can be performed independent of the code. We have made an attempt to address this issue in the form a case study in the context of On Line Item Purchase System.

3. WEB SERVICE ARCHITECTURE

A web service can be defined as a set of self-contained component based applications, implemented on different servers that communicate other applications by exchanging XML messages using Simple Object Access Protocol (SOAP) interfaces [2]. Web service architecture represents a service-oriented architecture (SOA) that incorporates three basic components: Service Provider, Service Broker and Service Requester [Figure-1].

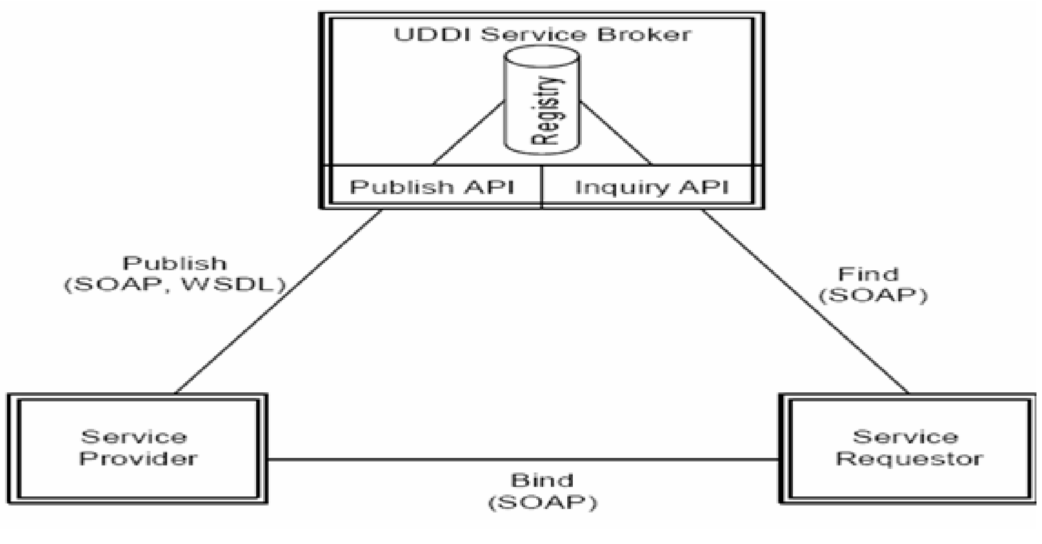

Figure-1. Web service architecture.

Services of a service provider are published to the service broker. The service requester finds the desired service with the service broker and binds to itself [22]. Functionalities of the web service are supported by the standards such as Web Service Description Language (WSDL), Universal Description, Discovery and Integration (UDDI), the Extensive Markup Language (XML) and SOAP. The service provider creates a web service, generates the required WSDL file and then publishes it on the global internet. Elaboration of access methods of the corresponding service and the set of operations associated with the service are incorporated in the ESDL file. UDDI registry that includes the specification of services and the URL, pointing to the respective WSDL file of the services are incorporated with the service broker. A desired web service can be searched by the service requester in the UDDI registry, which is then bound to it, following which XML messages and data are transmitted using SOAP interfaces.

4. WEB SERVICE MODELING

Web applications are designed by integrating together a set of interactive components which are invoked by events, and then, such applications can be viewed as event driven applications or services. Such interactions are associated with data dependence, control dependence and call dependence [1]. Two statements are data dependent if the value of a variable computed by one statement is used by another statement. If there is a flow of program control from one statement to another, then control dependence exists between them. The call dependence persists between a calling procedure and a server program to be invoked. The call dependences are categorized as inheritance and internal call dependences. Further, event based dependences are classified as link, visible effect and invisible effect dependences [1]. When one page requests for another page, then link dependence exists between them. If one page requests a second page and modifies it, then the requested page opens with the modified information and such dependence is known as visible effect dependence. If the modification incorporated by the requesting page is not displayed, then it is called as invisible effect dependence. In addition, semantic dependence exists between an informative object and a page or even another informative object, where the information flows from the former to the later. A UML based web service modeling combines all the dependencies discussed above among the interacting components of the desired web service.

5. UML BASED WEB SERVICE REGRESSION TESTING

Many application software testing accounts for 30 to 50 percent of software development cost. Delaying the start of testing activities until the entire development is accomplished represents a high risk approach. Hence, if we begin testing from the requirement design phase, the defect/error propagation will be significantly less in the later phases of application development. This is very well supported by UML based design method that depicts requirement and design in a lucid manner. Use cases are drawn at the design phase (before the code is written). Use cases tell the customer what to expect; the developer what to code; the technical personnel what to document and the tester what to test. Creating test cases is the first and fundamental step of testing. Hence, creating test cases from use cases makes an emphatic approach of testing much before the code is written, substantiating the fact of saving tester's time, labor and other tangible resources. Deriving functional test cases has the following benefits:

a) Avoiding duplicate testing;
b) Better test coverage;
c) Easier monitoring of testing process;
d) Easier regression testing;
e) Even work load balancing between testers;
f) Early discovery of missing requirements;
g) Decrease in project time by moving some tasks from construction to elaboration.

UML based regression testing has the following advantages over code based testing [23].

Traceability: Changes are easily traceable from design rather than from code, since differentiating a minute modification in the modified code from the original code is much difficult, and is mostly ignored by the developer in the implementation phase.

Scalability: Scalability represents the major bottleneck in code based regression testing as it is applicable only in small scale, whereas UML based regression testing can be performed at all levels and applicable in a large scale, i.e., large software applications.

Understandability: UML design approach is easily understandable and provides an explicit insight on requirements and specification, whereas in code based testing, the tester needs to understand the code which is a tedious job.

Language dependence: Codes written in different language pertain to different regression testing methods. But, UML based regression testing is language independent.

Cost: Huge amount of cost is involved in code based regression testing as errors/faults may be identified in later stages of software life cycle and these errors/faults need to be rectified which is cost consuming. In UML design, errors can be spotted in the design phase that reduces the overall cost.

Code dependence: In component based software, there may exist some dependence among the different components. To perform code based regression testing for such software, the tester needs the source code which is not practical. UML based regression testing can be carried out without such a dependency in the code.

Complexity: Retrieval of information from different static and dynamic components of a UML diagram is easier, whereas it is difficult to extract information about dynamic bindings from the code.

Executable UML: Executable forms of UML such as Executable UML and the UML virtual machine can be easily validated through UML based regression testing.

6. CASE STUDY

UML based regression testing can be performed by generating test cases from use cases [24]. The most important part of a use case for generating test cases is the flow of events. Two important components of flow of events are: basic flow of events and alternate flow of events. Basic flow of events should cover what normally happens when the use case is performed. The alternate flow of events covers the behavior of an optional and unexceptional characteristic relative to normal behavior. It may also be the variations in normal behavior. We have chosen the On Line Purchase application [Figure-2], a web based service, to address web service regression testing.

VOL. 7, NO. 11, NOVEMBER 2012 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

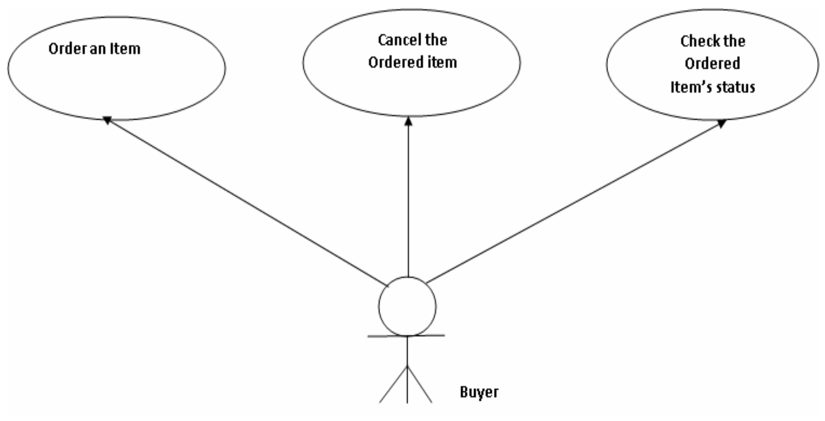

Figure-2. Use case diagram of an on line item purchase system.

Each use case also requires a significant amount of text in order to describe it. The text should be formatted as shown in Table-1.

Table-1. Format for a use-case textual description.

Use case section	**Description**
Name	An appropriate name for the use case
Brief description	A brief description of the use case's role and purpose
Flow of events	A textual description of what the system does with regard to the use case (not how specific problems are solved by the system). The description should be understandable to the customer
Special requirements	A textual description that collects all requirements, such as non-functional requirements, on the use case, that are not considered in the use-case model, but that need to be taken care of during design or implementation
Preconditions	A textual description that defines any constraints on the system at the time the use case may start
Post conditions	A textual description that defines any constraints on the system at the time the use case will terminate

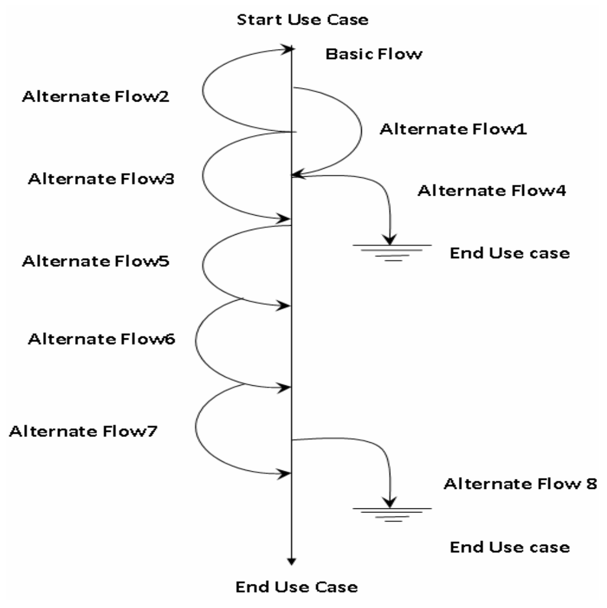

Figure-3. Basic flow and alternate flows of the events of the use case of on line item purchase system.

6.1. Basic flow

In the on line purchase of an item, the basic flow (B) of the use case is detailed below [Figure-3].

a) **Step-1.** User enters the web site address in the browser, following which the system shows the *log in* page.
b) **Step-2.** User enters an email address and a password, and in anticipation to it, the system confirms correct *log in*, shows main page and prompts for input.
c) **Step-3.** User either enters a search string or selects an item from a list of choices. Then system shows the items available for the selected category.
d) **Step-4.** User selects a category and then, the system presents detailed information about the selected category of item.
e) **Step-5.** User puts the item into the shopping cart. Shopping cart contents are displayed.
f) **Step-6.** User selects *"Buy now"* option. The system asks for a shipping address.
g) **Step-7.** User enters the shipping address. The system prompts shipping options and charges if any.

h) **Step-8.** User confirms shipping option. The system asks the method of payment (Credit Card/Cash on Delivery).
i) **Step-9.** User confirms Credit Card/Cash on Delivery. The system asks for the final confirmation to purchase the item.
j) **Step-10.** User buys the item. The system returns a confirmation number.

6.2. Alternate flows

The alternate flows (A) involved in the use case of on line purchase system are detailed below [Figure-3].

i) **A1:** For unregistered user/buyer the system displays a message.
ii) **A2:** The system prompts *"Invalid Password"*.
iii) **A3:** Mismatch of a searched item is prompted.
iv) **A4:** An item is declined.
v) **A5:** Shopping of another item continues after an item into the shopping cart.
vi) **A6:** Enter a new address.
vii) **A7:** Enter a new credit card.
viii) **A8:** Cancel the order.

6.3. Use case scenarios

A use case scenario is an instance of a use case or a complete "path" through the use case. Users of the completed system can go down many paths as they execute the functionalities prescribed in the use case. Following the basic flow would be one scenario and following the basic flow and the alternate flow would be another. Basic flow plus another alternate flow would be a third scenario and so on. The scenarios detailed in Table-2 can be used to generate test cases.

Table-2. Scenario for use case shown in Figure-2.

Scenario1	**Basic Flow**			
Scenario2	Basic Flow	Alternate Flow A1		
Scenario3	Basic Flow	Alternate Flow A2		
Scenario4	Basic Flow	Alternate Flow A3		
Scenario5	Basic Flow	Alternate Flow A4		
Scenario6	Basic Flow	Alternate Flow A5	Alternate Flow A6	Alternate Flow A8
Scenario7	Basic Flow	Alternate Flow A5	Alternate Flow A6	Alternate Flow A7
Scenario8	Basic Flow	Alternate Flow A5	Alternate Flow A8	
Scenario9	Basic Flow	Alternate Flow A5		

6.4. Generating test cases

A test case is a set of test inputs, execution conditions expected results developed with a particular objective; to exercise a particular program path or verify compliance with a specific requirement. Test cases are necessary to verify successful and acceptable implementation of the product requirements (use cases).

A three step approach is taken to generate test cases from a fully detailed use case:

a) Generate a full set of use case scenarios for each use case.
b) For each scenario, identify at least one test case and the conditions that will make it execute.
c) For each test case, identify the data values with which to perform the test.

6.4.1. Generate use case scenarios

By reading the textual description of a use case, one can identify each combination of basic and alternate flows. Table-3 shows a partial scenario matrix for the on line item purchase system. Here, some simple scenarios are taken into consideration.

Table-3. Partial scenario matrix for on line item purchase system.

Scenario	Starting Flow	Alternate Flow
Scenario1: Successful login	Basic Flow	
Scenario2: Unidentified Buyer	Basic Flow	A1
Scenario3: Invalid Password	Basic Flow	A2
Scenario4: No Item Found	Basic Flow	A3
Scenario5: Decline an Item	Basic Flow	A4
Scenario6: Change of Address	Basic Flow	A6

VOL. 7, NO. 11, NOVEMBER 2012 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences
©2006-2012 Asian Research Publishing Network (ARPN). All rights reserved.
www.arpnjournals.com

6.4.2. Identify test cases

After the full set of scenarios has been identified, the next step is to identify the test cases. This can be achieved by analyzing the scenarios and reviewing the use case textual description of the use case as well. There should be at least one test case for each scenario. There probability might be more. The next step in detailing the test cases to read the textual description of the use case and find the conditions or data elements required to execute the various scenarios. For example, for registering as an on line buyer, the required conditions may be valid buyer ID, password, correct address and some other attributes.

To document the test cases effectively, the test case matrix format (Table-4) can be used. The first column of this Table holds the test case ID, the second column contains a brief description of the test case including the scenario being tested and all other columns except the last one hold the data elements that are useful for implementing the tests. The last column specifies description of the output of the test case.

Table-4. Test case matrix for use cases of items in on line purchase system (I: invalid; V: valid; N/A: not applicable).

Test case ID	Scenario/ condition	Buyer ID/ login	Password	Item selected	Address correct	Mode of payment	Credit card input	Expected result
OP1	Scenario1 successful login	V	V	N/A	N/A	N/A	N/A	User home page displayed with a hello message
OP2	Scenario2 unidentified user/ buyer	I	N/A	N/A	N/A	N/A	N/A	Back to login screen with an error message
OP3	Scenario3: valid buyer quits	V	V	N/A	N/A	N/A	N/A	Login screen reappears
OP4	Scenario4:Search Item not available	V	V	V	N/A	N/A	N/A	Error message and back to OP1
OP5	Scenario5: decline an item	V	V	V	N/A	N/A	N/A	Error message and back to OP1
OP6	Scenario6: address change	V	V	V	V	N/A	N/A	Confirmation message of new address
OP7	Scenario7: purchase selected item	V	V	V	V	V	V	Purchase confirmation with “thank you” message

This particular matrix (Table-4) is a good intermediate step that clearly shows what conditions are being tested for each test case.

6.4.3. Identify data values

After all the test cases are identified, they should be reviewed and validated in order to ensure accuracy and to find redundant and missing cases if any. Then, once they are approved, the final step is to substitute invalid I and valid V (but not for N/A) options incorporated in Table-4 with actual data values, and thus Table-5 results. Without test data, test cases/test procedures cannot be executed as they are descriptions of conditions, scenarios as well as paths. Hence, it is essential to identify actual data values to be used while implementing the final tests. Table-5 shows a test case matrix with values substituted for I and V options. This is a document given to the tester, where the tester will fill the last three columns in a real environment.

VOL. 7, NO. 11, NOVEMBER 2012 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

Table-5. Test case matrix for use cases of items in on line purchase system with data values.

Test case ID	Scenario/ condition	Buyer ID/ login	Pass word	Item selected	Address correct	Mode of payment	Credit card input	Expected result	Actual result	Pass/rail	Comments
OP1	Scenario1 successful login	RKM007	rkm123007	N/A	N/A	N/A	N/A	User home page displayed with a hello message			
OP2	Scenario2 unidentified user/buyer	RKM0007	N/A	N/A	N/A	N/A	N/A	Back to login screen with an error message			
OP3	Scenario3: valid buyer quits	RKM007	rkm123007	N/A	N/A	N/A	N/A	Login screen reappears			
OP4	Scenario4:Search Item not available	RKM007	rkm123007	Nokia lumia 610	N/A	N/A	N/A	Error message and back to OP1			
OP5	Scenario5: decline an item	RKM007	rkm123007	Nokia lumia 610	N/A	N/A	N/A	Error message and back to OP1			
OP6	Scenario6: address change	RKM007	rkm123007	Nokia lumia 610	A-12, Pallash palli, bhubaneswar	N/A	N/A	Confirmation message of new address			
OP7	Scenario7: purchase selected item	RKM007	rkm123007	Nokia Lumia 610	Yes	Credit Card	5001201033121002	Purchase confirmation with "thank you" message			

As the current practice shows, use cases are implemented at the front end of the software development lifecycle, whereas test cases are implemented at the later part of the lifecycle. By generating test cases from use cases, the tester can execute its actions much earlier in the software development lifecycle, which makes it possible for the tester to identify and repair the faults that are costly to fix in the later part, ship on time and ensures the reliability.

7. CONCLUSION AND FUTURE WORK

As practice shows, regression testing of web services presents a set of challenges to the tester, which need to be addressed in order to guarantee reliable performance of the desired application. With a huge domain of web services, the issues and challenges may vary from application to application. Code based regression testing becomes a difficult job for the tester as the code might not be known to him/her. We present in this paper a UML based regression testing approach using test cases generated from use cases that is conducted absolutely independent of the code. We have addressed the regression testing of On Line Purchase System, a category of web service. However, the complexity of regression testing may vary from application to application with respect to underlying software architecture. In future, we intend to use the test case approach to perform regression testing of more complex web services.

REFERENCES

[1] IEEE. 1990. IEEE Standard 610.12-1990, IEEE Standard Glossary of Software Terminology, Juval Lowy, Programming WCF Services, O'RELLY.

[2] Abbas T., Zahi I. and Nashat M. 2008. Regression testing of web applications. Proceedings of the International Conference on Advanced Computer Theory and Engineering. pp. 902-906.

[3] Abbas T., Hacene F. and Nashat M. 2006. Regression testing web services-based applications. Proceedings of the IEEE International Conference on Computer Systems and Applications AICCSA'06. pp. 163-170.

[4] Lin C., Ziyuan W., Lei X., Hongmin L. and Baowen X. 2010. Test case prioritization for web service regression testing. Proceedings of the 5th IEEE International Symposium on Service Oriented System Engineering. pp. 173-178.

[5] Michael R. and Shengru T. 2007. A safe regression test selection technique for web services. Proceedings of the 2nd IEEE International Conference on Internet and Web Applications and Services (ICIW'07).

[6] Bo Y., Ji W., Chao L. and Luo X. 2010. A regression testing method for composite web service. Proceedings of the International Conference on Biomedical Engineering and Computer Science (ICBECS). pp. 1-4.

[7] Marcello B., Gerardo C. and Massimiliano D.P. Regression testing of web services. www.rcost.unisannio.it/mdipenta/papers/regression-ws.pdf.

[8] Athira B. and Philip S. 2010. Web services test case prioritization Proceedings of the IEEE International Conference on Computer Information Systems and Industrial Management Application (CISIM). pp. 438-443.

[9] Michael R. and Shengru T. 2010. Towards automating regression test selection for web services. Proceedings of the 16th International Conference on World Wide Web (WWW'07). pp. 1265-1266.

[10] Tsai W.T., Ray P., Weiwei S. and Zhibin C. 2002. Coyote: An XML-based framework for web services testing. Proceedings of the 7th IEEE International Symposium on High Assurance Systems Engineering (HASE'02). pp. 1-2.

[11] Tao Z., Qing Y., Xiaoning Z., Chunyan M. and Haipeng W. 2011. An approach of end user regression testing for semantic web services. Proceedings of the IEEE International Conference on Management and Service Science (MASS). pp. 1-4.

[12] Takao S., Kenji I. and Muneo T. 2009. Synchronization of multi-window requests for server-side regression test of web applications. Proceedings of the 9th IEEE International Conference on Quality Software. pp. 129-134.

[13] Michael E.R. and Shengru T. 2008. Empirical studies of decentralized regression test selection framework for web services, TAV-WEB'08. pp. 8-14.

[14] Nadia A. and Mark H. 2008. Automated session data repair of web application regression testing. Proceedings of the IEEE International Conference on Software Testing, Verification and Validation. pp. 298-307.

[15] Yanelis H., Tariq M.K., Jairo P. and Peter J.C. 2009. A meta-model to support regression testing of web applications. Proceedings of the 47th Annual Southeast Regional Conference (ACM-SE 47).

[16] Tamim A.K. and Reiko H. 2009. A methodology for model-based regression testing of web services. Proceedings of the 2009 Testing: Academic and Industrial Conference-Practice and Research Techniques, IEEE Computer Society. pp. 123-124.

[17] Tiziana M. Oliver N. and Bernhard S. 2003. A practical approach for the regression testing of IP-based applications, IP Application and Services 2003: A Comprehensive Report, Industrial Engineering Consortium, Chicago, USA.

[18] Michael R., Feng L. and Shengru T. 2006. Applying safe regression test selection technique to Java web services. International Journal of Web Services Practices. 2(1-2): 1-10.

[19] Tsai W.T., Ray P., Yamin W., Chun F. and Dong W. 2002. Extending WSDL to facilitate web service testing. Proceedings of the 7th IEEE International Symposium on High Assurance Systems (HASE '02). pp. 1-2.

[20] Ashok K.S., Gokul P.K., Ankur D. and Andhe D. 2009. Automated regression suite for testing web services. Proceedings of the IEEE International Conference on Advances in Recent Technologies in Communication and Computing. pp. 590-592.

[21] Sheng H., Zhong J.L., Ying L., Jun Z, Yang H.X. and Wei W. 2011. Regression testing as a service. Proceedings of the IEEE International Conference on Web Services, IEEE Computer Society. pp. 484-491.

[22] 2004. IBM Web Services Architecture Team, Web Services Overview, In IBM.

[23] Mohammad F. and Aamer N. 2008. A survey of UML based system testing. Proceedings of the International Federation for Information processing. 288: 200-210.

[24] Jim H. 2001. Generating test cases from use cases. Rational Software, IBM.