Efficient Agent Based Testing Framework for Web Applications

Ms.Neha Saluja Prof.Amit Kanskar

Abstract- Now days, making use of web based applications becomes crucial for the success of businesses worldwide. But as they are open and built on Internet, this kind of applications is imposing the new challenges to the developers and researchers such as such as dynamic behaviors, heterogeneous representations, novel control flow and data flow mechanisms, etc. In the previous studies, the agent based approach provided for web application testing in order to reduce the complexity of such applications. A four-level data flow test approach can be employed to perform structure testing on them. In this approach, data flow analysis will be performed as Function Level Testing, Function Cluster Level Testing, Object Level Testing, and Web Application Level Testing, from low abstract level to high abstract level. But that approach was limited because only the basic features of such framework are implemented.
Therefore, in this research thesis we are further extending that framework with more specific features implement like specific test agents for each particular type of Web document or object. Moreover, integrating more testing approaches, such as navigation testing, object state testing, statistical testing, etc., is still necessary for a systematic testing approach for Web applications.

Index Terms—Testing, Web Applications, Function cluster level testing, Object level testing, Function level testing.

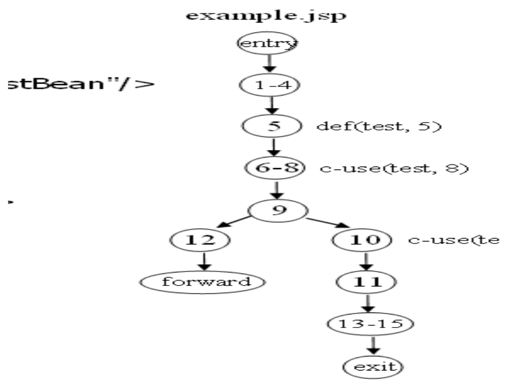
1 INTRODUCTION

In the last few years, web-based systems as a new genre of software systems have found their way into many different domains like education, entertainment, business, communication, and marketing. Parallel to this interest in development of web-based systems, many needs arise due to the importance of assessing the quality of these systems. Software testing is the traditional mechanism for this purpose and it has long been used in the software history. Web-based systems, due to their special characteristics and inherent complexities are more difficult to test, compared to traditional software. These complexities increase the cost of testing web-based systems. Test automation is the main solution for reducing these costs. Considerable effort has been dedicated to the development of tools, techniques and methods that automate different tasks in the testing process, but they are usually limited to one part or activity of the test process (e.g. test case generation, test execution). In addition to these limited solutions, some works have focused on presenting an integrated test framework that can be used to perform the whole test process with as much automation as possible. The complexity of web-based systems dictates that a systematic test framework, which is suitable for their architecture, is needed rather than a set of in- dependent tools.
In this project, an agent-based framework is presented for testing web-based systems and a prototype of this framework is developed. The main design goals have been to develop an effective and flexible system that un-like most of the existing test frameworks are capable of supporting different types of test with as much test automation as possible. The framework is designed to be capable of utilizing different sources of information about the System under Test (SUT) in order to automate the test process.
To meet these goals, the proposed framework is a multi-agent system consisting of a set of agents. Different agents, collaborating with each other, perform the activities involved in the test process. Therefore, one of the main issues in the design of the framework is the identification and separation of different parties and roles that are involved in the test process. From this point of view, a reasonable design helps to improve the extendibility and flexibility of the framework.
.

2 PREVIOUS WORKS

Wireless Ad-hoc networks are required where a fixed in the previous studies, we observed that agent based framework is proposed for web applications testing. However this framework having very limited functionalities for evaluating the complexity of such web applications. Following are the drawbacks of existing approach:

- Limited Agents for limited features.
- Only basic features implemented.
- No specific test agents for every kind of web document.
- Limited Testing approaches implemented.



3 Proposed Approach

- Thus in the project we are further extending the previous basic agent based testing framework for web applications into a novel multi-agent framework for automated testing of web- based systems. In this project we are presenting an effective and flexible framework that supports different types of tests and utilize different sources of information about the system under test to automate the test process. In this multi agent framework, we address the limitations of agents in previous approach as well as more specific features are considered and implemented in the framework. In this approach we are also consider integration of more testing approaches to become more flexible and efficient framework.
-

The existing web application test frameworks have two main characteristics in common. First, all of them are somehow limited both in terms of the test strategy they use (white-box, black-box, gray-box) and the types of tests they are designed for. For instance, some framework addresses only white-box strategy and session-based test case generation, while some framework is addresses only security tests. The second point is that, despite their differences, the way they finally execute a test is almost similar. In other words, regardless of whether a security test is being executed, or a functional test generated from TTCN-3 specifications, in both cases the test execution is performed by a set of HTTP interactions with the target system. Therefore, it can be concluded that it is possible to have a framework that supports different types of tests. The reason is that a test, whether a security test or a load tests, finally is executed in terms of a set of HTTP interactions with the SUT. So, if there is a formal format for test specification, then it is possible to develop different modules, each of which generates the specification of a special type of test. In addition, a single module can be developed for execution of all types of tests. All that is needed is that the tests are represented in a format that the executer module understands, and the executer module is able to behave like a web browser and perform HTTP-based interactions. The proposed framework relies on this point to support different test types.

Our goal was to design a test framework for testing web applications. The main design goals were effectiveness and flexibility. By effectiveness we mean that the framework is useful for automated execution of different types of tests, such as functional, load, stress, security or regression test. By flexible we mean that the framework should be designed in a way that adding new functionalities can be achieved with some reasonable level of effort, i.e. the architecture of the framework is open to future changes and improvements. To meet these goals, it was decided to design a multi-agent architecture for the framework. By analyzing the system from a more abstract point of view, different concepts (e.g. test script, test code) and roles (e.g. test script generator, test executer) involved in the test process were identified.

In the proposed framework, different kinds of agents responsible for performing different tasks and playing different roles are defined. This separation of concerns is helpful in achieving the desired goals. As each agent is responsible for performing almost a single task, it reduces the complexities of implementing the agents and also enables new agents to be added in the future. Another benefit of using multi agent architecture is that different agents can be distributed across a network and provide a distributed framework for testing web-based systems that are themselves inherently distributed. This distributed architecture can increase the effectiveness of the framework because it facilitates some tests to be performed in a more actual style. The main drawback of using a multi-agent architecture for the framework is that it imposes some communication overhead because of the messages that must be transferred between different agents to perform their activities. In addition to the communication overhead, the definition of interfaces through which different agents collaborate with each other is important. 2.2 Multi Agent System.

The critical difference between multi-agent systems and individual agents focuses on the patterns of communication.

A multi-agents system communicates with the application and the user, as well as with the other agents in the system to achieve their objectives. However, in the Individual agent, communication channels are only open between the agent and the user. The key characteristics in multi-agent environments are:

• Multi-agent systems provide the infrastructure for inter-agent communication.

• Multi-agent systems are usually designed to be open concept without any centralized designer.

• The agents within a multi-agent system are autonomous
and may be cooperative or competitive in
nature.
The most important aspect of multi-agent systems is
the communications between the agents. Many protocols have been developed that give the agents the ability to both receive and send information to each other.The overall architecture of the framework and its parts are illustrated in Figure 1. Different parts of the system are discussed in the following sections.

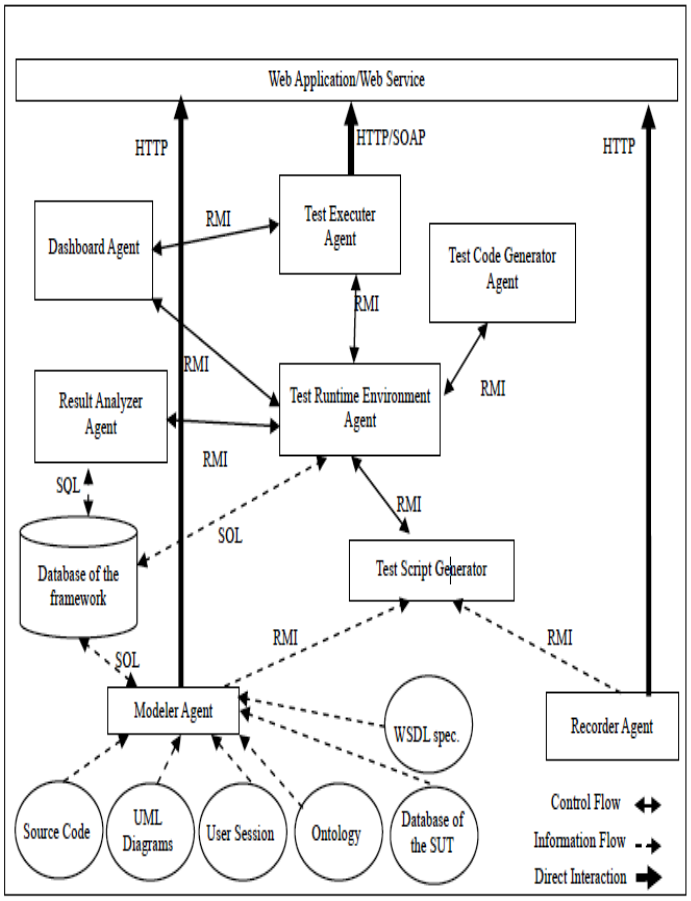
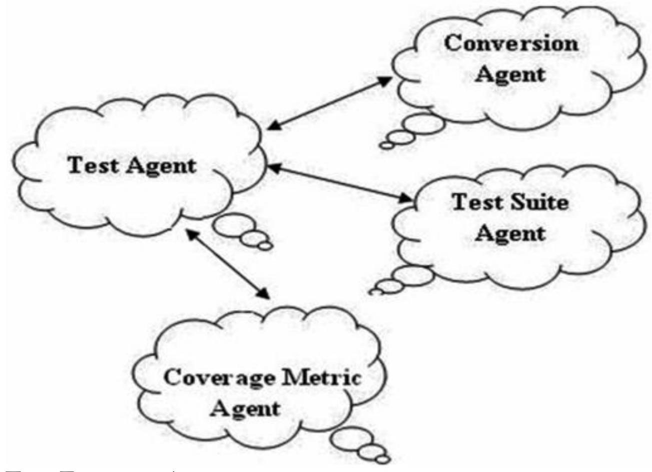


Figure 1. The architecture of the proposed framework.

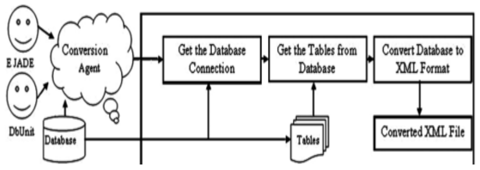
4 BASIC TERMS

Test Script: A test is composed of a set of actions or steps (e.g. opening a web page, entering some values in the fields of the page, submitting the page…). Each action has a type (e.g. open, submit, fill, assert Title) and it may require some parameters (e.g. the URL of the page to be opened). Therefore, having an appropriate set of actions defined, a test can be specified in a text file which we call it test script. It is worth mentioning that a test script contains test criteria and information needed to judge about the test result. In other words, there is no separate part as a test oracle.
Test Code: Test code is a piece of program written in a programming language which is logically equivalent to a test script. A test code is generated by performing some transformations on a test script
Test Case: Test cases are data items used in performing different steps of the test. For instance in the login scenario presented earlier, the values used as the user-name and password are some test cases.
Test Runtime Environment Agent
Test Runtime Environment (TRE) agent is the central part of the system. It communicates with other agents in order to manage the setup and execution of different activities of the test process. TRE is also responsible for providing suitable interfaces for the user. TRE uses Test Script Generator (TSG) agent for creating test scripts. When TSG has created the test script, it sends it to TRE. Receiving the test script from TSG, TRE passes it to a Test Code Generator (TCG) agent, which creates the test code from the test script, compiles it and returns the com- piled test code back to TRE. Then, TRE allocates some Test Executer (TE) agents for executing the test, and sends the compiled test code to them to be executed. TRE is also responsible for allocating a Dashboard agent and introducing it to the TE agents executing the test. TE agents communicate with the Dashboard agent to provide real-time information about the test process.
Test Script Generator Agent
TSG agent is responsible for providing facilities through which the user can create a test script. Using TSG, the user can select how the test script is generated. There are two possible choices in the framework: using a Recorder agent, or using a Modeler agent. Based on the user's choice, TSG calls the recorder agent or the modeler agent to create a test script. These agents, after generating the test script, return it back to TSG. TSG enables the user to view the test script and to edit it if required. After all, TSG sends the test script to TRE and TRE continues the test process.
Test Code Generator Agent
Test Code Generator (TCG) agent generating a test code from a test script, compiles it and sends the compiled code to TRE.



Test Executer Agent

A TE agent receives the executable code (generated by TCG agent) from the TRE. It then executes the received code. In addition, during the test execution, it is in communication with a Dashboard agent and sends the partial results to it. After the execution of the test is completed, the TE agent sends the total result to the TRE. It is important to note that it is possible (and even sometimes required) that multiple TE agents be involved in running a test. For instance, in case of load test, multiple agents can be created on different machines and execute the test from that machine to simulate concurrent users of the system


E JADE
DbUnit
Conversion Agent
Database
Get the Database Connection
Get the Tables from Database
Convert Database to XML Format
Tables
Converted XML File

Dashboard Agent

When a TE agent is executing a test, it sends the partial results of the test to a Dashboard agent. Dashboard agent uses such data to provide a real-time display of the test execution status and test results.

Result Analyzer Agent

When the TRE receives the test results from TE agents, it sends them to a Result Analyzer agent to perform user- specified analysis on them. It is possible that different types of Result Analyzer agents, with different capabilities, exist in the system. Such agents can create reports in different formats and generate different kinds of graphs and tables presenting the test results in more comprehensible forms.

Recorder Agent

Recorder agent is responsible for generating test scripts by recording the user interactions with the SUT. It provides a browser-like facility for the user to perform some interactions with the SUT and it captures these interactions as a test script.

Modeler Agent

Modeler agent, which enables model-based testing, is used to generate a test script based on some formal or semi-formal model of the SUT. If such models are avail-able, they can be utilized to generate test scripts. Different types of Modeler agents can be implemented, each of which uses different source of information as a model to create the test script. We have identified these types of models or information sources:

• Navigation model: The simplest case for a Modeler agent is to create a test script from the navigation model of the SUT. Navigation model represents a web application in terms of its composing pages and navigation links.

• UML Diagrams: a modeler agent can use the UML diagrams of the SUT to create test scripts. Such test scripts can be used for functional tests for instance. Especially if OCL (Object Constraint Language) is used in the UML diagrams to specify restrictions on concepts of the system, they can improve the performance.

• Session Data: session data can be used by a modeler agent to generate test scripts.

• Ontology: Ontologies can also be used as a source of information to generate test cases required for a test script.

• Source code: in case that the source code of the sys-tem is available, it can be utilized to generate test scripts, for instance test scripts that cover all the execution paths. Techniques like Java annotations can be used to add useful metadata to the source code to ease such test script or test case generation.

• Database of the SUT: Although it is not a model of the system, but the database can contain useful in-formation about the concepts and entities present in the SUT.

• Security: A modeler agent for security testing gene-rates a model for the system from the perspective of evaluating its security. The result of this test provides some useful information about the degree of security of the system based on ASVS standard.

5 IMPLEMENTATION

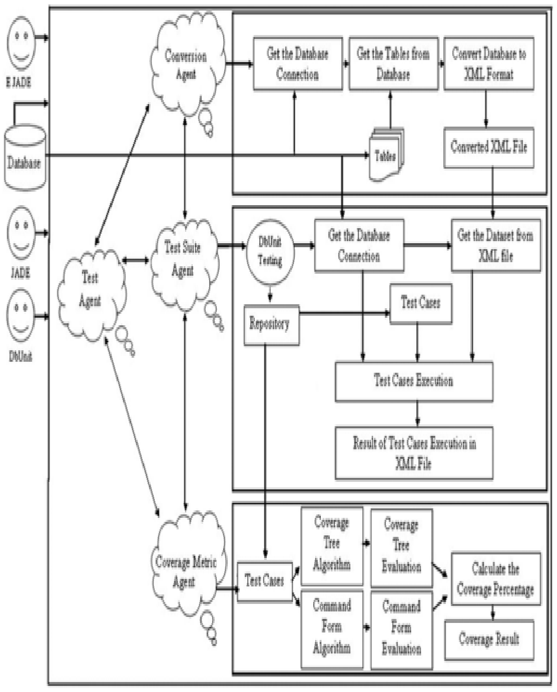
Prototype of the proposed framework was implemented in Java. In this section some issues about the implementation of this prototype are briefly discussed, since a comprehensive discussion of the implementation details is beyond the scope of this paper. JADE2 is used as the under- lying infrastructure of the framework. It provides the essential services for developing a multi-agent system and hides many low level complexities and implementation details. TRE, TE, Dashboard, Result Analyzer, TSG, TCG and Modeler agents have been developed. A Recorder agent is developed which uses Selenium. Selenium is an open source tool that provides the recording functionality through a plug in for Fire-fox browser.

In the prototype system, the format of the test scripts was chosen to be the same as that of Selenium. A Selenium test script is a simple XHTML file, containing a table. Each row of this table (except the first one) indicates one step of the test. Each step represents one action. The first row indicates the title of the test script (i.e. its name). Other rows have three columns. The first column includes the name of the action. Other two columns are used for the parameters of that action (e.g. the URL to be opened, the field name to be filled with the input value, the expected title).

Test scripts can be created manually or automatically by the framework. Since test scripts are simple text files, they later can be edited easily by human testers. In the current implementation of the framework, there are different possibilities for creating a test script: Using recorder agent, and using modeler agent. Different types of modeler agents are implemented: based on navigation model, based on session data, based on both the navigation mod el and the database of the SUT and also the notion of on- technologies. In addition, another modeler agent is developed for web service testing.

A TCG agent is implemented which translates the test scripts into Java source code. The generated source code uses Selenium HTML Unit class (from the Selenium tool API) to simulate behaviour of the browser. TCG compiles the generated Java class and sends the created .class file to the TRE. When a TE agent receives this compiled test code from the TRE agent. Then it creates a new object from the received .class file (using Java reflections). We call this object the test code object. Then is starts execution of the test by calling the action methods on the test code object. Each action method executes one of the test steps. Dashboard agent receives the test results from TE agents during the test execution and generates diagrams representing number of failed and passed action. A simple result analyzer agent is developed in the framework. Currently, in addition to computing the average number of failed steps among all executers, the result analyzer agent computes 'functional adequacy' and 'accuracy to expectation' defined based on ISO/IEC 9126 standard.

In order to perform security tests, a modeler agent was implemented that focuses on generating test scripts for security tests. This Security agent uses w3af4, which is a Python-based tool. Based on the user configurations, Security Agent creates a simple test script. This test script is defined using a set of new actions we added to actions defined by Selenium. These actions are specific to w3af it means that TCG agent translates theses actions to specific Java code which enables running w3af plug ins from Java.



6 PERFORMANCE EVALUATION

A comparison of the proposed framework with similar works discussed in this paper is presented in Table 1 (given below). This comparison is performed based on these factors:

• Supported Test Types: The more test types are supported by a test framework, the more powerful is that framework.

• Test Strategy: Generally there are three test strategies. Black-box testing imposes the least requirements for the test to be performed. It does not re-quire the source code or internal information about the SUT. White-box strategy is on the other end. It requires that the source code of the system to be available. Gray-box strategy resides in the middle. It requires some information about the internal structure of the system or its details, for instance the database structure, but not the source code. A framework that is limited to white-box strategy has less applicability than one that uses black-box strategy, because it may not be possible to ask the providers of a system to make the source code of the system accessible in order to test the functionality of public interface of the system.

• Information Sources: This item indicates the types of information sources that are utilized by the framework to automate the test process. A framework that is able to use different sources (e.g. UML models, session information, source code…) is clearly more effective than a framework that works only in the presence of a single source.

• Human Manual Intervention: The less human intervention is needed in the execution of a test pro- cess, the more effective is the underlying framework.

• Test Applicability Time: In which phases of SDLC the framework can be used? Is the framework applicable only when the system is deployed or it can be used during the whole development cycle?

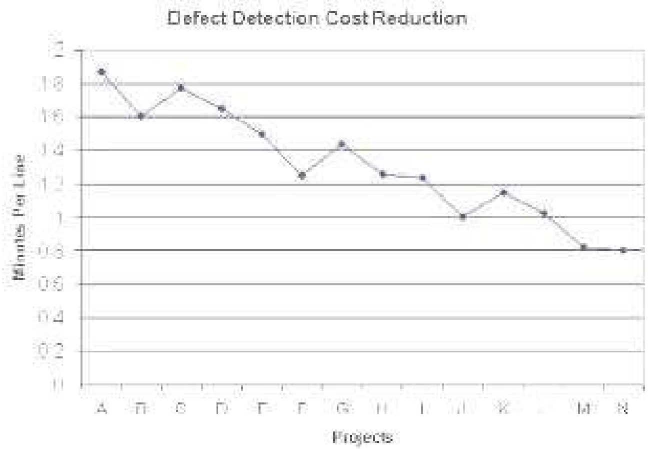
• Framework Architecture: As mentioned before, a distributed framework is more powerful and flexible in the testing web applications, because it copes better with the characteristics of these systems.

• Target Type: What type of systems can be tested using the framework? Does it support web services or only traditional web application?

Here, we concentrate on discussing the proposed framework with regards to these factors. The framework supports functional, load, stress, security, and performance tests. All of these tests are possible through appropriate test scripts. For instance if a test script for assessing SQL injection is available then this test script can be used to per-form a security test. Therefore, the main issue is how to represent the logic of a test in a test script. After such a test script is available, it is executable. Fortunately, all of the test types mentioned above, can be represented in Selenium test script, because they do not need anything more than a sequence of HTML interactions with the SUT. Selenium HTML Unit API declares methods for handling dynamic behaviour of web pages, but these methods are not yet completely implemented. Our point of view is that our framework will be capable of testing dynamic aspects of web pages (e.g. Ajax) if required such functionality is provided by Selenium. Currently the framework is used for performing some load, stress, functional, performance, and security tests. If a valid test script representing the logic of the test is available, the test process can be performed automatically. Therefore, the main issue is the way a test script is generated. As mentioned before, the framework provides facilities for automatic test script generation based on the user session logs and navigation model of the SUT. It also provides semi-automatic test script generation using the recorder agent. The framework supports all three test strategies. Based on the presence or absence of different information re-sources, different functionalities of the system might be available or unavailable. At least, the black-box strategy is available and the system requires no access to the internals of the SUT. But if some sources like user sessions or system models are available, the frame can well utilize them. The manual intervention in the framework is at an acceptable low level. The framework provides automatic and semi-automatic facilities for creating test scripts. After a test script is created, it can be run automatically with little human intervention (e.g. specifying some parameters). Also as mentioned in section 3, the level of automation gained by the framework is much more in case of distributed tests. The framework is useful in testing operational systems. Therefore, it does not support tests like unit test and integration test. Although some of these tests can be performed by functional tests. The framework is a distributed one, consisting of different agents collaborating with each other.

7 Conclusion

In this project, a multi-agent framework was introduced for testing web-based systems. Different agents are designed with specific roles and they collaborate with each other to perform the test. The main design goals have been to develop an effective and flexible framework that supports different types of tests and utilize different sources of information about the system under test to automate the test process. One of the novelties of this work is the use of test code which is based on the idea of mobile code. It provides benefits like increasing the performance, and decreasing the complexity of test executer agents. Another novelty of the work is the modeler agents that use different in-formation sources for automatic test script generation. A prototype of the proposed framework has been implemented and is used to perform some experiments.



These results are promising and verify the overall design of the framework.

The cost of defect detection has dropped dramatically.

8 REFERENCE

[1] A. G. Lucca and A. R. Fasolino, "Testing Web-Based Applications: The State of the Art and Future Trends," Information and Software Technology, Vol. 48, No. 12, 2006, pp. 1172-1186.

[2] A. G. Lucca and A. R. Fasolino, "Web Application Testing," Web Engineering, Springer, Berlin, Chapter 7, 2006, pp. 219-260. doi:10.1007/3-540-28218-1_7

[3] S. Murugesan, "Web Application Development: Challenges and the Role of Web Engineering," J. Karat and J. Vanderdonckt, Eds., Web Engineering, Modelling and Implementing Web Applications, Springer, Berlin, 2008, pp. 7-32.

[4] A. G. Lucca and M. Penta, "Considering Browser Interaction in Web Application Testing," Proceedings of the 5th IEEE International Workshop on Web Site Evolution, IEEE Computer Society Press, Los Alamitos, 2003, pp. 74-83.

[5] F. Ricca and P. Tonella, "Web Testing: A Roadmap for the Empirical Research," Proceedings of the Seventh IEEE International Symposium on Web Site Evolution, Budapest, 26 September 2005, pp. 63-70.

[6] S. Sampath, V. Mihaylov, A. Souter and L. Pollock, "Composing a Framework to Automate Testing of Operational Web-Based Software," 20th IEEE Conference on Software Maintenance, Chicago, 11-14 September 2004, pp. 104-113. doi:10.1109/ICSM.2004.1357795

[7] S. Elbaum, S. Karre and G. Rothermel, "Improving Web Application Testing with User Session Data," Proceedings of the 25th International Conference on Software Engineering, Portland, 3-10 May 2003, pp. 49-59. doi:10.1109/ICSE.2003.1201187

[8] S. Elbaum, G. Rothermel, S. Karre and M. Fisher, "Leveraging User-Session Data to Support Web Application Testing," IEEE Transactions on Software Engineering, Vol. 31, No. 3, 2005, pp. 187-202. doi:10.1109/TSE.2005.36

[9] H. Zhu, "A Framework for Service-Oriented Testing of Web Services," 30th International Computer Software and Applications Conference, Chicago, Vol. 2, 17-21 September 2006.

[10] P. Dhavachelvan, G. V. Uma and V. Venkatachalapathy, "A New Approach in Development of Distributed Framework for Automated Software Testing Using Agents," Knowledge-Based Systems, Vol. 19, No. 4, 2006, pp. 235-247. doi:10.1016/j.knosys.2005.12.002

[11] B. Stepien, L. Peyton and P. Xiong, "Framework Testing of Web Applications Using TTCN-3," International Journal on Software Tools for Technology Transfer (STTT), Vol. 10, No. 4, 2008, pp. 371-381.