Functional Tester Runner Tool

Leticia Dávila-Nicanor[1], Aldo Benhumea-Peña[1],
Carlos Omar González-Morán[1], Pedro Mejía-Álvarez[2]

[1] Centro Universitario UAEM Valle de México, CDMX, Mexico

[2] Centro de Investigación y de Estudios Avanzados del IPN, CDMX, Mexico

{ldavilan, cogonzalezm} @uaemex.mx, pmalvarez@cs.cinvestav.mx

Abstract. The sheer complexity of Web applications and inadequate approaches to analyze, design or update such projects generate incorrect environments, which are the source of potential risks. Poor applications reflect low availability, maintenance difficult and questionable efficiency in the services they provide. At present the quality assurances schemes still performed manually. A group of evaluators are based on a test matrix to run tests, and thus according to the degree of experience is that the results are interpreted by each evaluator. This process requires much time and effort, since the goal is to test the system thoroughly to identify as many potential errors in the system. This proposal focuses on the development of an assessment tool with the ability to emulate a group of virtual testers in real contexts of tests for applications on the Internet. This system generates controlled testing using virtual evaluators that emulate the activity of people atmosphere. The results have been satisfactory, currently we worked on the proposal to create a factory abstract of test cases to extend the functionality tool.

Keywords: Testing tool, web applications, reliability.

1 Introduction

Software systems are exposed to significant changes during its development, maintenance and evolution. These affect system functionality and quality you must possess a productive area. The central problem is to have functionality changes that impact directly on the software architecture and design. Functional and non-functional requirements set out in the statement of the purpose of developing a software system. Regarding functional requirements in architecture and software components of the system it is designed. When a system is undergoing maintenance and even evolving relationship has clear requirements with system design is central because it's the basis to extend or upgrade the functions of a system. However, it is up to the test phase when the functionality of the system is evaluated.

This situation is much more noticeable on Internet system, in this case the number of users to whom it may concern, emerging technologies, processes and services that cater as in the case of banking portals and business sites, now that provides a special

relevance. The complexity of applications and inadequate approaches to analyze, design or update such projects generate incorrect environments which are the source of potential risks. Poor architectures reflect low availability, maintenance difficult and questionable efficiency in the services they provide.

Companies that develop software development processes have been used systematically highlight the need for and select tools that benefit the operating costs in software testing. Thus, they have useful and efficient tools, according to their contexts of operation is a necessity that the software project leaders and administrators argue from the last decade of the twentieth century [1]. The interest of companies and organizations that develop and test software systems are focused on reducing production costs for software testing. These costs are assessed on the duration of the evaluation process; the effort hours/man is implicit in the test coverage and computing resources consumed. Thus, assessment tools have important expectations such as tangible improvements in test coverage and implementation of a greater number of test cases at lower cost. Productivity and quality thresholds are dictated by the policies of the company and the project managers, some companies require very high thresholds. Today a lot of organizations perform evaluations of their systems manually, with a group of people called (QA) that are based an array of tests to run tests manually and functionality according to the degree of experience is that the results are interpreted by each evaluator. This process is very expensive because the goal is to test the system thoroughly, to locate as many potential errors in the system.

According to studies Miranda and Jelinski the trend's still exponentially [2]. So, to test a system it is requiring thousands of evaluations. In studies that have been conducted [3] and [4], 100 tests functionality of an application 1200 lines of code carried by a single evaluator lasts about 30 days, in the embodiment of 5000 evaluations it would take approximately 1500 days, namely 4.10 years, this time lapse exceeds any economic development, human resources and the projected time in any case. If we combine this fact that having all these assessments don't guarantee that a software system operate without failure, the problem is even more critical.

This proposal is based on the development of efficient software with the ability to emulate a group of virtual testers in real contexts of reliability testing for Internet applications. In developing the tool they have combined several techniques to make assessment process reliability for Internet system. In this case turned on a group of virtual evaluators is emulated; for this process elements and statistical simulation were taken for analysis of the activity of each of the evaluators elements of analysis used by the scheme compilers high-level languages.

The paper is organized as follows; in section 2 of the theoretical framework and related work is discussed. Section 3 operation executing functional test is described. Section 4 architecture and the main executor of test algorithm is presented and finally in Section 5 conclusions and future work are described.

2 Related Work

Web applications possess unique characteristics that make web testing and quality assurance different from the corresponding traditional techniques. Web applications can be characterized by the following aspects [5]. Massive Access of users, this

simultaneous access of the users in these applications is part of the essence of systems. Web applications provide cross-platform universal access to web resources for the massive user population. For the users it should be transparent that these web applications provide this service to millions of other users. Difficulty of establishing causes of the errors. Since web applications may be accessed by millions of users, errors have a big impact. Finding the origin of errors in web Applications may be difficult and its recovery time may not be immediate, given the great number of software elements that intervene.

The integration of diverse software elements for an application on the Internet, Web users employ different hardware equipment's, network connections, operating systems, middleware and web server support. In a web application, two main components are always required: the backend and the frontend. The backend is the software required for an application on the Internet to operate. Among the most important software found in the backend are: the database servers (MySQL, Oracle, Informix, DB2, among those most important), Web Servers (Apache, Netscape Enterprise Server, Netra of Sun, etc.), and the interface programming languages (HTML, XML, PHP, Servlets-Java, Live - Wire, etc.). The frontend is the software required on the part of the client to allow the systems to access the Web. Among the most important software found in the frontend are: Navigators (Explorer, Netscape), which contain plug-in software such as presentations of Macromedia, and languages like JavaScript.

Diversity of frameworks are develop to operate and to maintain a Web site. The development of a Web site requires of a great team of people with different profiles and backgrounds. These teams include programmers, graphic designers, and usability engineers, specialists in information integration, network experts and database administrators. This diversity of personnel profiles makes reliable web applications development difficult and sometimes unpredictable. Because of the above characteristics web-based systems tend to evolve rapidly and undergo frequent modifications, due to new technological and commercial opportunities, as well as feedback from the users. In consequence web-based systems are very sensitive to errors. Most work on web applications has involved making them more powerful, but relatively little has been done to ensure its quality. The most important quality attributes demanded by web-based systems are reliability, usability and security. Additional important quality attributes are availability, scalability, maintainability and time-to-market [6].

2.1 Automated Assessment Tools for Web Systems

In the article by Robert M. Poston and Michael P. Sexton [1] an entire study on the needs and requirements that must cover the evaluation tools from the point of view of managers and project leaders who develop software business addresses. Until then there wasn't clarity on the needs assessment tools that were developed, misinformation predominated and tools hitherto existing lacked related to the sector which evaluates software goals, this work focuses on developing a summary precisely these needs. Table 1 shows a product form requirements evaluators sectors in companies that develop software.

The companies generally want to see greater development productivity and software quality as a result of the incorporation of new assessment tools. The concern is the

potential for improving productivity and quality of the evidence, whether a tool can have tangible and substantial improvements. Productivity and quality thresholds are dictated by the policies of the company and the project managers. Some companies require very high thresholds.

Table 1. Form needs related to the testing process [1].

Test-productivity (cost) data	Present data from recent Project	Predicted data if new tools are not acquired data
Predicted cost of testing in staff months		
Predicted cost for all testing		
Predicted cost per testing activity		
Planning testing		
Defining test objectives		
Designing tests		
Constructing test environments		
Executing tests		
Evaluating tests and software		
Test-quality data		
Test coverage		
Requirements coverage		
Input coverage (valid, invalid, etc.)		
Output coverage		
Structure coverage (DU path, branch, etc.)		
Predicted data if new tools are not acquired		

Since 2000 year, companies such as IBM. Hewlett Packard and Rational Software, has been concerned about the need to develop tools that can be used in the evaluation process of software development, the latter being an international economic activity, it is very important that the software products operate with high standards [7]. However the quality of an Internet system according to Jeff Tian studies [6] has six aspects: Reliability; Security; Usability; Availability; Maintainability and Scalability. Instruments or tools that are automate the evaluation of this process with different approaches. Below is a comparison chart is presented, the most important tools available today.

Table 2 shows some of the most important characteristics of the instruments most comprehensive evaluation exist in this case most licensing are summarized; such is the case of Web Link [8], Mercury LoadRunner [9], Rational Functional Tester [10] and WAPT [11]. There are other less robust instruments that do not take licenses for its implementation [12], [13], such as: Apache JMeter (evaluates efficiency) Curl-loader

(evaluates efficiency), Selenium (simulates multiple access). Cloud-based tools: Blitz (evaluates performance), Testize (evaluates usability).

Table 2. Comparative table of tools that automate the testing process.

Name	Description	Quality attribute	Advantages	Disadvantages
Web Link (Rel software)	It is a software which is responsible for checking links and web addresses.	Availability.	– Check links – Has spell check – Make content validation – Send emails reports	Costs: 3000 bonds $145 to $1,195 (limited links).
Mercury LoadRunner (Hewlett Packard)	Testing tool for software application performance.	Efficiency.	– Emulates a lot of users interacting with a specific application at the same time. – It can measure response times of processes.	Costs
Rational Functional Tester(IBM)	Tool for functional testing and automated regression.	Functionality.	– Automated testing. – Tests based on data. – Test Script (manual sequences).	$7,017.20
WAPT (SoftLogica, 2014)	It is a tool that evaluates the performance of web applications and interfaces related by generating virtual evaluators.	Efficiency.	– Supports load specified users at all times.	Users have a Static activity.

3 Functional Tester Runner Tool Architecture

The Functional Tester Runner (FRT) tool has been based on the development of a software system with the ability to emulate a group of virtual testers in real contexts of reliability testing for Internet applications. The property to be evaluated is *Reliability*, which is defined as the *probability that a free operating system failures within a specified framework of time and under specific environmental conditions* [14]. This group of virtual testers perform tests and obtain metrics on a defined Web application in a defined time frame. The group´s activity is based on a black box testing technique,

this is combined with statistical simulation [15] and analysis of the activity of each of the evaluators used the analysis approach language compilers high level.

When designing our system architecture (see Figure 1), it can be seen that control passes from the *Arrival test evaluator* to the *Process test state*, control then forks to two concurrent flows, the enclosing object will be in the *Web application* state and the *Testing activity state*. Furthermore, while in the *Testing activity state*, the enclosing object will be in the *Test path* and *URL_conexion* state. The *analyzer* reads the Web server file answer and calls to the parser program to make a new question related with the test case. The *analyzer* instance reads the response from the Web server and calls the *parser* instance to make a new request that is related to the test case. This focus is important in the information upgrade, where the select and update activity are involved more than one form. Finally, a report suite based on programing shell, reads the log files and make the final report. This report included the metric result. The tool nowadays to make a defect density report. The defect density is a software metric.

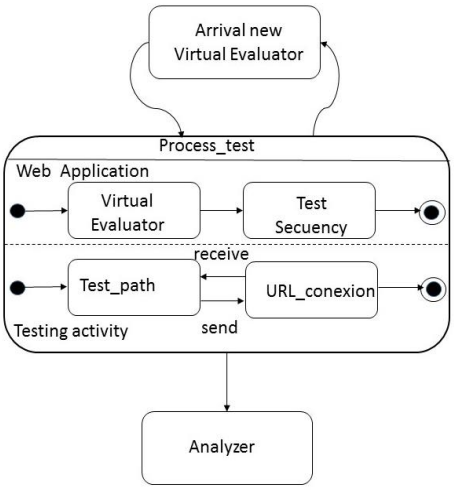

Fig. 1. System architecture to run functional tests.

Fig. 2 shows the sequence diagram for the module arrivals. In the instance initialization *(Init)*, data structures, statistical counters and the first arrival is scheduled initialized. The *new_test_application* instance aims to estimate the time of arrival of virtual evaluators check whether waiting in line at the pool of servers and if the system has capacity to serve more requests. If the architecture has the required capacity, an event access to the virtual evaluator with prior knowledge to assess the application is activated. The *process_test* instance contains the main process where a call is made to *Departure_pro()* method in turn causes a flame to its corresponding instance. In the instance Departure user activity it is established, taking into account its assessment coverage. In Statistics, the results of the evaluation process previously conducted by virtual testers and statistics are generated are stored. The simulation conditions are as follows:

– The time of arrival for evaluators to the system is determined by an exponential distribution $\mu = 5$.

- Server queues *M/M/1* type in the pool of Web servers were used.
- To determine whether the user expects the server's attention in the queue, a probability of *4/(n+1)*, where n represents the current queue size (number of users on the system) was used.
- For reasons of reliability in the operation of the system can work properly even with evaluators *k* (where *k* = 3400).
- Depending on the application and coverage of evaluation, evaluators can perform only *x* types of transactions. The operating time depends on the evaluators related transactions allocated to each evaluator coverage.

The activity of the evaluation process that makes each evaluator as shown in Fig. 3 by a sequence diagram component test process. When they arrive the virtual evaluators them is assigned a test case according to the coverage assessment by *Secuency_path* instance, the script is formed into a loop to form a list of nodes associated with the application components to evaluate this process done with the interplay between *Secuency_path* and *Component* instances. When the virtual evaluator executes a sequence of each node evaluation by *test_path* instance, the *send_receive* method sends requests to the server where the application and evaluated the responses are analyzed by the Analyzer component is housed.

Analyzer component in the activity analysis of the results obtained in the evaluation process by the sequence diagram is developed (see Fig. 4). In this case the results obtained by the virtual evaluators analyzed through *test_case_analysis* instance and its *verify_answer_test* method sends the requested response. The *scanner_html* instance examines the contents of the response and using the *parser_test_case* instance *make_next_test* method and the following request is built

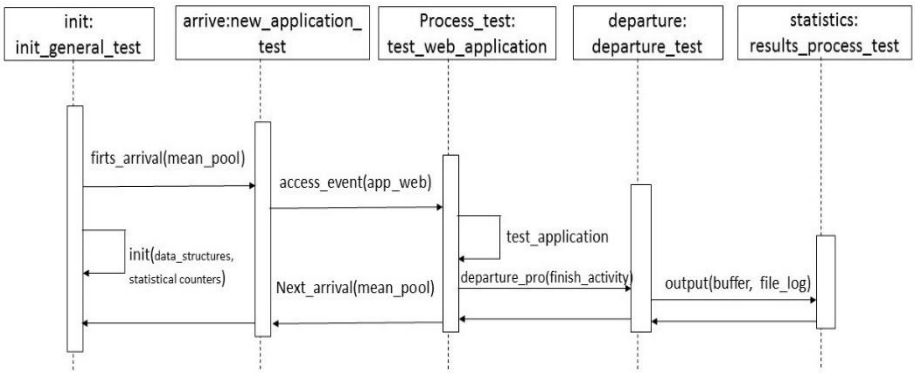

Fig. 2. Diagram for component sequenced arrivals.

This tool was used to evaluate the reliability attribute under real conditions. Evaluation was conducted by our testing tool FTR using concurrent test threads (system testers). Each test thread is responsible for executing a specific functionality test specified on its corresponding test case file. Test cases are generated randomly and test data is prepared to perform functionality tests. For each test case, a specific test profile indicates a specific path of navigation (type of test) and the view that the tester will test. The test thread has access to test cases (which contain the test data and the test profile) and the activity log files. The activity log files are files that contain the activities

performed by the test thread. The analyzer reads the activity log files and produces an error log file which contains the specific faults detected and the defect density computed.

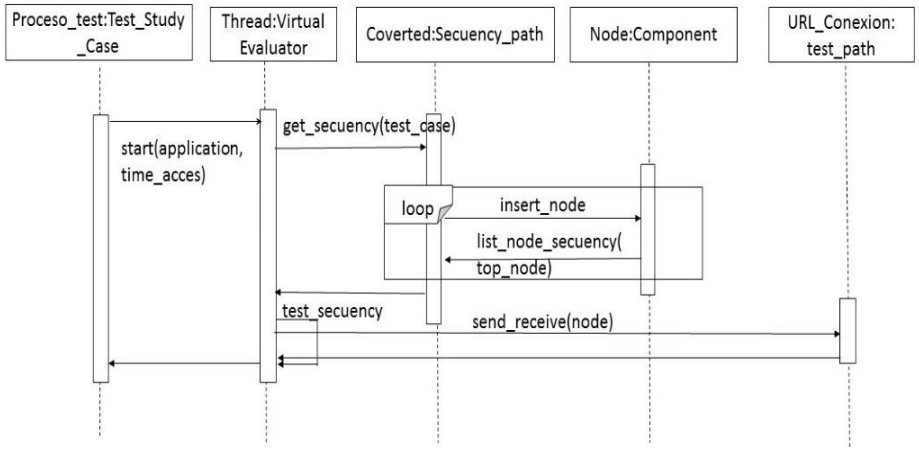

Fig. 3. Diagram for evaluation process.

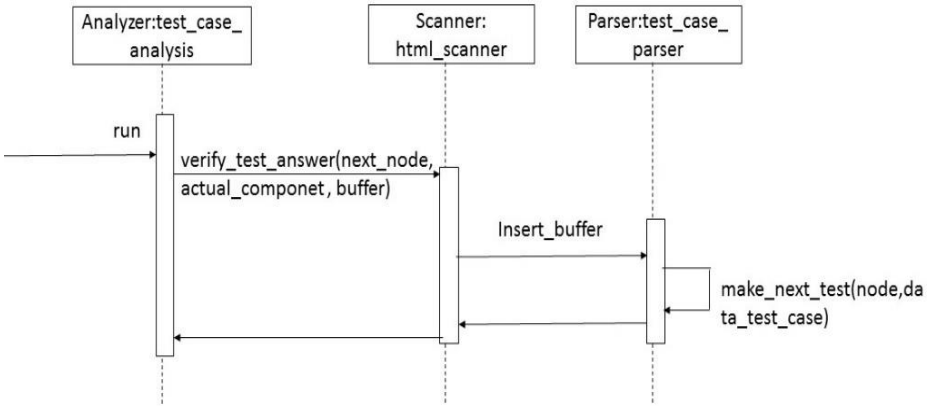

Fig. 4. Flow chart of activity Analyzer.

4 FTR Tool Implementation Results

The functional tester runner is implemented in Java programming language and has been used as an instrument of evaluation and analysis of the reliability of the systems on the Internet [3]. This tool was used to evaluate the reliability attribute on SOGU system on the publications Reliability improvement with PSP of Web-based software application [4], with the ability to emulate a group of virtual testers in real contexts of reliability testing for Internet applications. The process evaluation was conducted by our testing tool FTR using concurrent test threads (system testers). Each test thread is responsible for executing a specific functionality test specified on its corresponding test

case file. Test cases are generated randomly and test data is prepared to perform functionality tests. On each test case a specific test profile indicates a specific path of navigation (type of test) and the view that the tester will test. The test thread have access to test cases (which contain the test data and the test profile) and the activity log files. The activity log files are files that contain the activities performed by the test thread.

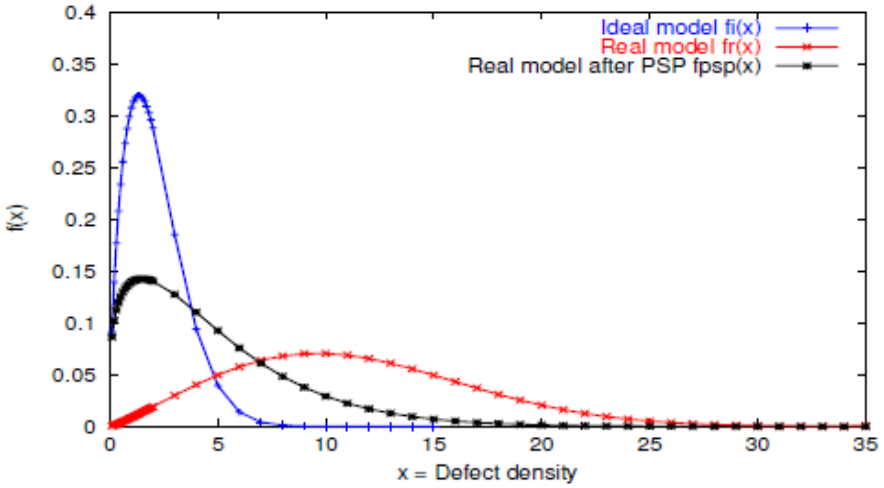

Fig. 5. Graph of SOGU system assessment before and after applying PSP - SEI CMM.

The graph in Figure 5 reports the results of the research [4] where the SOGU system was evaluated before and after applying the Personal Software Process SEI-CMM [16] and [17]. The evaluation was performed by the tool described here. In this case the model $f_r(x)$ describes the evaluation of the system before applying PSP and $f_{psp}(x)$ describes the evaluation system after applying PSP.

5 Conclusions

In conclusion we can say that today software development is an economic activity, where quality assurance is an imperative; however, the existing assessment processes are still expensive. That's why, it's appropriate to develop an instrument to support and contribute to reducing operating costs in software systems in organizations.

In this way, our research shows the development and implementation of a tool to assess the reliability of Web applications. This tool simulates a virtual group of evaluators who perform the testing process according to a scheduled coverage of specific cases. The combination of statistical simulation techniques to the process of assessing the reliability of the software is an effective scheme respect of the testing process serving virtual testers, control of variables such as weather and operating conditions can be handled efficiently in the context of concurrent processes.

For purposes of our research was relevant implement an automated assessment tool, because the approximate time 5000 for manual testing required approximately 4.16 years ; accordingly , such an approach far exceeds available resources in any organization. To evaluate system SOGU's reliability, were performed 100 tests, with a

single virtual evaluator, we need 30 days with the manual process. With the FTR tool operating on CISC architecture in a Red Hat Linux 7.0 platform only they took 5 hours to do 500 tests. In environment with a RISC processor and Solaris 2.7 platform with only they took 5 hours to do 5000 tests [3, 4].

In general, it is projected that the FTR can compete with tools developed by private companies such as Rational Functional Tester or Mercury Load Runner, but without restriction involving licensing; since the FTR is provided with mechanisms that improve its efficiency and scope in relation to the type of system to be evaluated.

6 Future Work

In the Functional Tester Runner tool, developing test cases is according to the analysis of traceability. Every test case is designed and coded. In the test execution, coverage is limited by the cases already scheduled. Currently we worked on the proposal of creates a factory of abstract test cases. With the current approach, the generation of test cases has new scope, because they are generated dynamically according to the information traceability test matrix. A very important advantage is that the test matrix has the ability to upgrade and expand. This approach improves the use of the tool from any perspective. In the evolution of software requirements, the functionality may vary according to the current context of system operation to generate new test cases you will only need to update the traceability matrix testing. In this case, the use of design patterns Abstract Factory and Builder [18] allow you to build test cases to run time.

References

1. Poston, R.M., Sexton, M.P.: Evaluating and Selecting Testing Tool. Software, IEEE, Vol. 9, Issue 3, pp. 33–42 (1992)
2. Horgan, A. H. Krauser, E.W.: Incremental Regression Testing. IEEE Proceedings of the Conference on Software Maintenance, pp. 348–357 (1993)
3. Dávila-Nicanor, L., Mejía-Alvarez, P.: Reliability improvement with PSP of Web-based software application. Computer Science & Engineering: An International Journal, Vol. 2, No. 4, pp. 106–112 (2012)
4. Dávila-Nicanor, L., Mejía-Alvarez, P.: Reliability improvement with PSP of Web-based software application. Computer Science & Engineering: An International Journal, Vol. 2, No. 4, pp. 106–112 (2012)
5. Kallepalli, C., Tian, J.: Measuring and Modeling Usage and Reliability for Statistical Web Testing. IEEE Transactions on Software Engineering, Vol. 27, No. 11 (2001)
6. Offutt, J.: Quality Attributes of Web Software Applications. IEEE Software, Vol. 19, Issue 2, pp. 25–32 (2002)
7. Software Engineering Technical Subcommittee of the IEEE Computer Society: IEEE Guide for the Use of IEEE Standard Dictionary of Measures to Produce Reliable Software, IEEE Std. 982.2-1988 (1988)
8. Li, W., Moore, A.W.: Classifying HTTP Traffic in the New Age. SIGCOMM'08: proceedings of the 2008 SIGCOMM Conference and Co-Located, pp. 17–22 (2008)
9. Stantchev, V., Schröpfer, C.: Techniques for service level enforcement in webservices based systems. 4th International Conference, GPC 2009, Geneva, Switzerland, May 4-8. Proceedings, Series Title Lecture Notes in Computer Science, Springer-Verlag, pp. 25–35 (2009)

10. Davis, C., Chirillo, D., Gouveia, D., Saracevic, F., Bocarsley, J.B., Quesada, L., Thomas, L.B., van Lint, M.: Software Test Engineering with IBM Rational Functional Tester: The Definitive Resource. Goebel, A.: Pearson plc (2010)
11. Carvalho, R.E.: Ames Res. Center, NASA, USA, Williams, J., Sturken, I., Keller, R.: Investigation Organizer: the development and testing of a Web-based tool to support mishap investigations. Aerospace Conference, IEEE, pp. 89–98 (2005)
12. Vithani, T.: BITS Pilani, Dubai, United Arab Emirates; Kumar, A. Presentation 5. A comprehensive mobile application development and testing lifecycle. IT Professional Conference (IT Pro), pp. 1–3 (2014)
13. Thanawala, P., Pareek, J., Shah, M.: OntoB Aeval: Ontology Based Automatic Evaluation of Free-Text ResponseTechnology for Education (T4E), IEEE Sixth International Conference on, pp. 189–190 (2014)
14. Mussa, J.D.: Software Reliability Engineering. McGraw Hill, (2000)
15. Law, A.M., Kelton, W.D.: Simulation Modelling and Analysis, McGraw-Hill Series in Industrial Engineering and Management Science (2000)
16. Humphrey, W.S.: Introduction to the Personal Software Process. Addison Wesley (1997)
17. Humphrey, W.S.: Using a Defined and Measured Personal Software Process. IEEE Software, Vol. 13, No. 3, pp 77–88 (1996)
18. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns. Elements of Reusable Object-Oriented Software. Addison Wesley, Pearson Education (2005)