A SELENIUM BASED APPROACH TO AUTOMATE SOFTWARE TESTING

THANUJA MURTHY S¹ & DR. CHETHANA R MURTHY²

¹Masters of Technology in Software Engineering, RV College of Engineering, Bangalore Karnataka – 560059

²Assistant Professor, ISE Department, RV College of Engineering, Bangalore Karnataka - 560059

ABSTRACT

Software testing is viewed as the most significant advance in SDLC (Software Development Life Cycle). The principle goal of testing procedure is comparing, acquired outcomes and the results for the software expected by the client. Automation testing disentangles crafted by the tester via automating the test-script execution by utilizing superior software. This study centers around the utilization of Selenium Web driver for testing a web-application and also to exhibit the utilization of the tool in mix with different tools such as the Test NG, Maven, and so on., for increasingly simpler way to deal with testing and thus increasing the testing process quality.

KEYWORDS: Selenium Web Driver, Maven, TestNG

Received: Jun 01, 2020; **Accepted:** Jun 20, 2020; **Published:** Jul 22, 2020 **Paper Id.:** IJMPERDJUN2020415

INTRODUCTION

Web testing is considered as a crucial group of testing programs that focuses for the most part on testing web-applications. Testing of a web-application is fundamental with a particular ultimate objective to perceive and change the mistakes, before it is revealed to everyone. There are different web application execution gadgets available. They are used to test the web-related interfaces and web-applications. Despite web-application testing, test web servers, destinations and further web-interfaces can be tested by utilizing of web-automation testing [1].

With the improvement of Internet and versatile innovation, Web-applications are continuously famous than customary win. Since Web programs dependent on standards of CSS3 and HTML5 became popular. It turns out to be increasingly more effectively to actualize the stunning interfaces for User-PC in Web-applications. However present web advancement procedures are progressively experienced, the examination of programming designing hypothesis despite everything should be improved practically speaking, for example, extreme programming and agile programming. In spite of the fast advancement in the field of technology, if a program is manually written rather than automation, the codes should be refactored, which has a requirement of being improved in terms quality of the software. Be that as it may, the similar issues with multi browser compatibility, despite everything inconvenience's the test engineers and developers, when structuring and Web pages are coded with JavaScript strategy, in light of the fact that every program perhaps has its particular analyzer for HTML/CSS and Engine for JavaScript modules. Along these lines, as a major test for programming quality affirmation and for maintenance, it is vital for a Web-application to refactor the codes written in JavaScript when moving the Web-application amongst different browsers.

The utilization of distinct software for automating the execution process of the test cases is referred to as automation-testing. Indeed, the text-cases in the test automation ought to be manually scripted [7], yet the executing these test-cases are automated by using tools. Test Automation [8] is helpful when test-cases ought to be essentially

re-used commonly. For example, during regression testing, same test-cases are executed over and over again on an application, at whatever point a change is done in the application. In such circumstances, manually executing the test-cases will be exceptionally tedious and there is a high danger of making blunders. So, automation testing ends up being helpfully productive when the requirements of an organization are changing time to time or when few test-scripts should be developed over and over. The tools for testing utilized for this reason may be costly. In this way, tools for automation testing are normally utilized in incorporation with the process of manual testing. In this way, as clarified above test automation ends up being progressively valuable when compared to manual testing process in different circumstances. In situations where a same test ought to be executed more than once, test automation is quicker and dependable. The human resource (HR) investment is likewise to decrease with automation testing. There are many open-source, free test automation tools in the market. Choosing the tool that is best for the product, can help in getting progressively precise, dependable outcomes in an increasingly reasonable way and likewise be financially savvy.

Present day web-applications have numerous clients in common that send numerous simultaneous requests. To guarantee that a web-application can deal with the quantity of simultaneous requests (load), the framework must be completely tested with the load. Load-testing is performed in-order to decide the behavior of an application under different load settings [7]. Dismissing a load test will lead to cataclysmic results. For instance, in 2016 - Statistics Canada site was broken for three hours since it couldn't deal with the traffic [2].

PROBLEM STATEMENT

Testing is considered as a major bit of programming progression since it allows you to measure quality of programming. On Lack of testing, there are no other genuine methods to affirm that a product is working precisely and bugs might be left redundant all over the application and are left undiscovered. Testing is performed now and then manually, by test engineers executing activities, systematically, and demonstrating whether a particular step is successful or a failure. Regardless, manual-testing is an around the clock effort, unnecessary and every now and again associated with human errors. Automation test-scripts should be revised consistently in the midst of progress cycles to ensure quality. Every time when the source-code is being changed the automation test-scripts should be revised respectively. For each new release of the respective project, the test-scripts may be broken for all maintained working systems. This task is truly reiterating this test is absurd and dull.

BACKGROUND

Automation Testing of Web-Application

Across the field of product business, quality confirmation depends on the testing a software similar to the Web-applications. Regardless of whether countless methods are not being used currently, the exploration on automation testing despite everything has come to blockage. Testing computer software despite everything relies upon manual control even in some propelled IT enterprises. Likewise, manual test engineers are bound to commit errors in almost every step in testing of Web-application. Along these lines, it expands test expense, which results in reduction of the work productivity. The above marvel particularly happens in the case of web-application improvement and also the testing.

In order to defeat the issues referenced over, the study on Web-application test automation method is broadly investigated in bunches of fields. As demonstrated by Shauvik Roy, a sort of test fixes technique for Web-application, which was named as WATER [5]. In the field of test-driven improvement, Michal and Marian product the pioneers of

building up a automation acknowledgment testing apparatus for Web-applications [6]. According to research in the MBT field, Tommi, Mika and Tuomas has executed a system working by the catchphrase (keyword) driven instrument, which consolidates the theoretical model with test-scripts the first run through [7]. As of late, Selenium is a prevalent automation tool, so an ever-increasing number of remote researchers utilize it. Rosnisa and Antawan dealt with Selenium in functional-testing process [8] and selenium in agile methodology [9]. Then, residential analysts are additionally constant chipping away at it. In Baosight-Corporation, an automated testing system which is dependent on Fit Nesse and Selenium [10] is applied during the time spent programming designing.

Truth be told, Web-application automation testing infers the UTP that is under test program which can be run consequently are determined by testing tools, these tools collects the outcomes of the execution, then perform the comparison of this result and the predefined results. In this way, test driving device is critical in programmed test. The elements of these apparatuses are comparable. Other than recording clients' activities, more highlights should be actualized, for example, executing activity contents and creating testing contents. In any case, how to understand the normal test result from conceptual experiments have been zero potential arrangements due to problem with Test-Oracle. Hence, current programmed Web-application testing is just utilized to relapse testing in industry so as to execute expenditure decrease [4].

Automation Tool - Selenium

Selenium was first released in 2004. "Jason Huggins" was the person behind this idea. He is the creator and architect of selenium. He started with the work during his attempt on inside application genuinely at thought works and he comprehended that he can improve use of the time rather than truly experiencing similar advances. He started to work with java-script and developed a library with java scripting which will interface with the website and can execute a similar test normally ordinarily against various projects. That java content library at that point got the chance to be Selenium Core, which consists of all the properties of Selenium IDE [5] and Selenium Remote Control (RC) [4]. Presumably Selenium initially was a huge gadget, but furthermore had many highlights [2].

Security element in Selenium, previously it was not too exact as a result of which a bit of the things was hard to implement in Selenium [3]. Web-applications were ending up being increasingly successful and complex bit by bit, so it was ending up being amazingly difficult to deal with it by utilizing Selenium. Engineer at Google Simon Stewart, started working on an endeavor then named it as Web Driver. Google was making use of Selenium lot; be that as it may, test engineers were going after the imperatives of the thing. Simon Stewart was in search of a web-testing gadget that will talk explicitly to the web program using a technique that is a locally available for the program and a working system. The Web Driver turned into a basic-factor with a purpose of overcoming the limitations of Selenium in the year 2008[4].

Figure 1: Selenium Framework Architecture.

As shown in the above Figure, selenium is a framework for Web-application automation testing which incorporates Selenium Core, Selenium RC, Selenium WebDriver and Selenium IDE. It recreates a progression of client activities as well as playbacks the occasions the test engineer worked during the process of testing.

Selenium Core is referred to as kernel-engine of the complete framework, it is a library of functions, written using JavaScript. All other remaining modules of Selenium relies upon it. In the above-mentioned design, Selenium-Core is referred as the base and establishment, which infuses into the under test Web pages in order to control the execution of the tests.

Figure 2: Selenium -Core Work Flow.

Selenium IDE is a program module in Firefox browser. Selenium IDE records and replays commands. Commands of Selenium can be effectively changed over into all bolstered programming dialects, which help's the test engineers to alter and reuse the existing test cases.

Figure 3: Workflow of Selenium IDE.

Selenium RC, abridged for Remote Control, it is a structure supporting propelled dialects to conjure the Selenium Core, to be specific referred as APIs. As indicated in the documentation, RC will have a Selenium Server that accepts and constantly monitors the Selenium Client's instruction. In the wake of getting the orders, Selenium RC will change over the guidelines into JavaScript code for various programs. As appeared in Figure 4, Selenium RC summons the piece capacities by infusing code into Web pages. Remote Control bolsters lattice registering. Critically, analyzers can run equal experiments simply like distributed computing idea. In addition, Selenium testing is constrained by server nearby, in which equal orders can be sent to servers utilizing equal calculation [12] when it is sent to multi-servers.

Figure 4: Workflow of Selenium Remote Control.

Selenium WebDriver is the most recent rendition (v2.0) and it is not same as RC. Since WebDriver doesn't infuse the contents into the Web pages, conjuring the program need the assistance of local APIs [12]. It is suggested by current authority documentation since its dependability and execution is more consistent compared to Selenium RC. Also, test engineers could quit any pretense of utilizing Selenium Server in programmed testing without equal.

Figure 5: Workflow of Selenium Web Driver.

DEFINITION AND DESCRIPTION OF TESTCASE

Before presenting a method for test-scripting in the generation of automation technology, we initially present some associated essential ideas. Numerous assignments in the field of testing have to be automated. For example, displaying, test-scripts age and execution. Furthermore, before executing the test-scripts, the key advance of automation testing is to utilize a sensible adaptable approach to officially depict the test-script, which is all the more near primitive language. This articulation can create and execute the test-scripts naturally just as confirmation [13]. This strategy should be very less intricate than customary in which the test-scripts are coded manually. Hence, it needs formal articulation and a test engineer. Indeed, automation testing is likewise not the same as model-based testing. However, our examination can coordinate with it.

As topic 3 explains, the Refactoring of the JavaScript code should be on the basis of “Functional testing”. For creating test-scripts inevitably, the current test-cases will be viewed as the establishment. Subsequently, test-cases must be structured by analyzers or created with respect to model-based testing before scripts production [14]. The current data may be portrayed by characteristic language, semi-formal or a formal technique, yet a language which is characteristic and also semi-formal can barely be acknowledged by PC so it ought to be changed over into the proper uniform so as to be gotten by the contents parsing apparatus. Also, to get ready test-cases for refactoring the JavaScript codes, these test-cases should be executed once in any event alongside the normal outcomes must be gotten as of now [15]. Along these lines, our apparatus can be utilized to achieve automation testing errands from contents age and execution stage to the outcome assessment stage.

Defining XML Schema for Test Cases

Selenium structure has a unique language structure to characterize the test-cases. Each Selenese speaks to a test-step and a rundown of them can be spared as a test-cases as HTML. Regardless, the data contained in Selenese isn't sufficient. The main factor is the depiction with Selenium just point in contradiction of a test-step and single test-case, while the statement of relationship between test-cases is marginally less. Additional factor is for a single test-case, there is zero unequivocal confinement, for example, limitations about the planning of approval. As it were, the check orders can vanish or show up in any situation with no reasonable measure in Selenium, yet in the genuine test circumstance, the test setting must be

confirmed and approved when each test-step. In light of these reasons, a progressively solid and adaptable linguistic structure ought to be theorized, which consolidates the upsides of XML information organization and Selenese. It very well may be called as ESCL (Extension Selenium Command Line) or also as X Selenese.

The highlights of XML give's a superior arrangement. The explanations behind expanding the Selenium order articulations with XML rather than HTML is as per the following:

- Use the tag nest feature and a reference that indicates the relation between the test-cases and reuse verification.
- Use XML Schema to normalize the data type of the XML and also to validate the XML files

The two highlights express that client can tweak their own linguistic structure for various test necessity. Furthermore, only a couple of rules and setups ought to be changed when reusing test-cases that the age framework despite everything works. The XML Schema for a single test-case is characterized as follow:

Figure 6: Test Case XML Tag Structure for Root Element.

Portrayal of experiment has a “root tag” is called as test case. Its “attribute name” recognizes each test-case by its name. The tag will have three “children” tags, which are “pre-check”, “steps” and “post-confirmation”, separately. The 1stchild tag is called as “pre-confirmation” is utilized to check the past setting condition. The 2nd child is called as “steps”, which portrays each test step subtleties individually. 3rdchild is called as “post-check”, which comprises the data about how to approve the outcome after test-steps are been executed.

Figure 7: Test Step Tag Structure and Steps Node.

Particularly, all of the key focuses are in the 2nd child. As shown in the Figure-7, the “step-tag”, as its name suggests, portrays the subtleties of each test-step. Each step-node ought to contain a few selenese labels, which are utilized to portray a couple of clients, activities before approving the outcome. It is critical here the confirmation hub with an attribute called "ref". It could connection to other check node characterized as of now in a similar test framework for reuse. Furthermore, the selenese-node additionally has its “childrens”, which are command-tag, locator-tag and “value” separately. The 3child hubs (nodes) relate to the three pieces of the Selenese coordinated. For example, if the input validated code is a test-step, then leave the command "type" or the "input" and give the locator as the selector of validated code component in the webpage like "id=v-code". The last contention statement could be a genuine legitimate code. Here is a legitimate XML design experiment record created with help of schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<cpmiss-ts:test-suite
  xmlns:cpmiss-ts="http://www.shu.edu.cn/CPMISS/TestSuite"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.shu.edu.cn/CPMISS/TestSuite TestSuite.xsd">
  <cpmiss-ts:test-case name="register"></cpmiss-ts:test-case>
  <cpmiss-ts:test-case name="login" reverse-case="logout"></cpmiss-ts:test-case>
  <cpmiss-ts:test-case name="change-password" reverse-case="this"></cpmiss-ts:test-case>
  <cpmiss-ts:test-case name="logout" reverse-case="login"></cpmiss-ts:test-case>
</cpmiss-ts:test-suite>
```

Figure 8: XML Demo of a Test-Case.

Near to the test-case as described in the above Figure-8, there are just two test-steps only. The initial step is written to open the webpage of CPMISS which is Condominium Property Management Information Server Secretariat, CPMISS. According to the test-case showed, the login structure containing username, secret word and a code for validation will be entered and then the login-button will be clicked. For the most part, login is considered as a top test-case in entire test-suites so its “pre verification” is precluded in the XML report. The pieces of verification-nodes have no child nodes. However just have a lot of "ref" property estimation. It shows that the check node is alluded to another current confirmation-node.

Defining a XML Schema for Test-Suites

The XML based portrayal of test-case as described above have few deformities that, how to show the dependencies among test-cases is certain. In particular, each and every test-case class is independent and isolated. That is a smart thought to develop a “register table” in test suite, in any case the wellspring of verification-node reference cannot be found in existing test framework. Subsequently, any other XML document is utilized to show the test-suites.

Figure 9: XML Schema of Test Suite.

As shown in the aboveFigure-9, test-suite consists of many test-cases. Be that as it may, it presents a new parameter which is "reverse-case". As shown in the Figure-10, this is a basic test suite created by XML-Schema, the

request for test-cases is arranged by “test-case nodes”. The test-suite contains only 4 test-cases, they are "register", "logout", "login" and "change-secret key". The attribute called “name” of each test-case tag is the identifier.

```

- <cpmiss:test-case name="login" xsi:schemaLocation="http://www.shu.edu.cn/CPMISS/Testcase TestCase.xsd">
  - <cpmiss:pre-verification>
    <cpmiss:verification ref="null"/>
  </cpmiss:pre-verification>
  - <cpmiss:steps count="2">
    - <cpmiss:step id="1">
      - <cpmiss:selenese>
        <cpmiss:command>open</cpmiss:command>
        <cpmiss:locator>http://220.248.23.234:8888/CPMISS/</cpmiss:locator>
      </cpmiss:selenese>
      <cpmiss:verification ref="login-page-verification"/>
    </cpmiss:step>
    - <cpmiss:step id="2">
      - <cpmiss:selenese>
        <cpmiss:command>type</cpmiss:command>
        <cpmiss:locator>id=username</cpmiss:locator>
        <cpmiss:value>*****</cpmiss:value>
      </cpmiss:selenese>
      - <cpmiss:selenese>
        <cpmiss:command>type</cpmiss:command>
        <cpmiss:locator>id=password</cpmiss:locator>
        <cpmiss:value>*****</cpmiss:value>
      </cpmiss:selenese>
      - <cpmiss:selenese>
        <cpmiss:command>type</cpmiss:command>
        <cpmiss:locator>id=vericode</cpmiss:locator>
        <cpmiss:value>cookie("v-code")</cpmiss:value>
      </cpmiss:selenese>
      - <cpmiss:selenese>
        <cpmiss:command>click</cpmiss:command>
        <cpmiss:locator>id=login</cpmiss:locator>
      </cpmiss:selenese>
      - <cpmiss:verification ref="null">
        <cpmiss:assert locator="url" value="http://220.248.23.234:8888/CPMISS/main.jsp"/>
      </cpmiss:verification>
    </cpmiss:step>
  </cpmiss:steps>
  - <cpmiss:post-verification>
    <cpmiss:verification ref="main-page-verification"/>
  </cpmiss:post-verification>
</cpmiss:test-case>

```

Figure 10: XML Demo of a Test-Suite.

It merits talking about the reverse-case attribute, which is utilized to deal with the converse test-case of current test-case. It's not quite the same as the figuring out in programming advancement. The innovative idea of "Reverse Test Case" is stated below:

Reverse Test Case—Normally on executing the Test-cases one can alter the test condition. On the off chance that there is alternative test-case that is executed after this, which makes the test condition come back to the first express, this test-case type is referred to as “Reverse Test-Case”. On the off chance that and just if the experiment is self-invert, its test condition won't change.

The reverse test-case for an instance of "login" test-case is the "log-out". On the off chance that the logout case is simply executed once log-in case is executed, the webpage status is equivalent to the previous log-in. Some framework might be storing the login bombed data; however, it doesn't have a place with the Webpage setting. The "change-secret key" case above is a self switch case. Obviously, it relies upon the information. The motivation behind why presenting the converse experiment is that experiments are frequently run over and again. For this situation, the test condition will change persistently with no opposite case. Test situations change will expand the trouble of the accompanying testing, particularly refactored JavaScript testing.

TEST SCRIPT EXECUTING AUTOMATICALLY

It is normal to change the test-suites into “test-scripts”. In this manner, how to remove the information and also parse the information arrangement to comparing codes have been factor to be worried. The test-case is necessary to communicate the littlest test-code unit, and also each test-suite ought to be appropriate to the extent of code creating. For the exhibition in past segment, there are 3 inward classes in any event in the produced programmed contents, those are exploring to the primary page of the CPMISS, contributing the client login data, click the login catch and afterward confirming the setting after login. The class chart is as shown below:

Figure 11: Generated Test Case Class Diagram.

Clearly in the Class-Diagram appeared over, each test-case is viewed as an independent class in test-script, each test-step is an inward class of the case. Utilizing this class one can improve the intelligibility and reuse pace of the code. The "test-suite.xml", "test-case.xml" in Class Diagram as shown in Figure-8 shows the two XML design documents. Its investigation objects are to change over XML tag into content codes. APIs of Selenium RC is simpler than WebDriver if test engineer might want to conjure on the grounds that there is inward static items where API will be able execute the Selenese with no change. Actually, the primary employment is to separate the order contentions received from XML setup and set them into the experiment classes all together. In the wake of breaking down all design documents of experiments and suites, the connections and references ought to be removed from the enrollment information construction of test-suites and "verification-nodes" in XML.

According to the example shown in Figure-9, class "Test Login CPMISS" which is a test-case executable test-script created by the strategy above. All experiments contents consequently created by the apparatus have regular highlights as follows:

- All the test-case classes are stretched out from the abstract super-class called "Default Test Case".
- All of the test-steps are inward classes and each progression has its own sort to recognize whether it's the verification-step or not.
- Each verification-step is an internal class, which must have been enrolled in the test-suite class.
- Initialize the test-steps performing vector as indicated by the progression id characteristic in XML.

Presently the pseudo code of test-scripts programmed change is given as following:

Algorithm 1: Analyze the XML and set script data

Input: TestCaseNode (the test case tag of XML file)
Output: TestCaseScript (the script data)

```

PreVerify © getPreVerification(TestCaseNode); IF
PreVerify.ref ≠ Null THEN
    SearchVerificationRef(PreVerify);
ELSE
    RegisterVerification(PreVerify);
END
TestCaseScript.parseScript(PreVerify);
WHILE hasNextStep(TestCaseNode.steps) DO
    CurrentStep © nextStep(TestCaseNode.steps);
    Selenese © parseCommand(CurrentStep);
    IF Selenese.isVerify THEN
        RegisterVerification(Selenese);
    END
    TestCaseScript.parseScript(Selenese); END
PostVerify © getPostVerification(TestCaseNode); IF
PostVerify.ref ≠ Null THEN
    SearchVerificationRef(PostVerify);
ELSE
    RegisterVerification(PostVerify);
END
TestCaseScript.parseScript(PostVerify);
RETURN TestCaseScript;

```

According to the algorithm explained above, the way toward parsing an XML based test-cases into test-scripts which is coded by JAVA. XML test-cases can be isolated into 3 sections. Each section maps to a tag in the XML file in experiment record. When the Selenese content examination, the past confirmation and post check should be taken care of and question the reference target in the event which has the attribute for reference purpose.

In the second section of the above algorithm, a loop that emphasizes the hub list under the “steps” to get guidance, also the parameters of each and every Selenese. The second section in the algorithm of code is to examine the order information from XML in light of the fact that the design documents contains entire data of test-scripts creation and afterward, the relating data about Selenium order line will be removed. In the event that the Selenese has, Selenium gives a progression of APIs to change over the substance into homologous code areas. The significant focuses are to make a mapping table and then search it rapidly.

But these highlights, the individual substance of every progression are legitimately changed over from the Selenium orders contentions, which extricated by XML reader utilizing DOM. At that point, if the script should be executed, some *.jar libraries and JDK condition should be arrangement. As a matter of fact, the Selenium independent server isn't vital, in light of the fact that the most recent structure prescribe to utilize the WebDriver present inside it. Which can conjure the program local APIs.

The parsing outcome of Log-in Case is as follows.

decreased and the less manual cooperation the couple of slip-ups will be made. As the future works, the UI, a few highlights merit considering. We intend to structure and actualize an overshadowing module. So as to raise the test proficiency and lessen the mediation of analyzers, we will concentrate on the best way to create the test contents naturally with the model-based testing method.

REFERENCES

1. Martin Fowler, Kent Beck, John Brant, *Refactoring: Improving the Design of Existing Code*, U.S., 1999, pp.11-21.
2. Martin Fowler, Kent Beck, John Brant, *Refactoring: Improving the Design of Existing Code*, U.S., 1999, pp. 2 –20.
3. Antawan Holmes and Marc Kellogg, *Automating Functional Tests Using Selenium*, *Proceedings of AGILE 2006 Conference*, 2006.
4. McMahon, C, *History of a Large Test Automation Project Using Selenium*, 2009.8.
5. Shauvik. R. Choudhary, Dan Zhao, *WATER: Web Application Test Repair*, ACM, 2011.
6. Marian, Michal. *Automated acceptance testing tools for Web application using Test-Driven Development*. *PzeglądElektroTechniczny*.2010-9.
7. Tuomas Pajunen, Tommi Takala and Mika Katara. *Model-Based Testing a General Purpose Keyword-Driven Test Automation Framework*. *International Conference on Software Testing. Verification and Validation Workshops*.2011.
8. Antawan Holmes and Marc Kellogg, *Automating Functional Test Using Selenium*. *Proceeding of Agile Conference*,2006.
9. Chris McMahon. *History of a Large Test Automation Project Using Selenium*. *Agile Conference*.2009.
10. Xinchun Wang, Peijie Xu, *Build an Auto Testing Framework Based on Selenium and Fit Nesse*. *International Conference on Information Technology and Computer Science*.2009.
11. Zhenghua Feng, Ju Gao, Hongwei Zeng, *Research of Web Application Test Automation*, *Computer Engineering and Design*, 2010.
12. Web Site, <http://docs.seleniumhq.org/docs/>
13. Pan Liu and Huaikou Miao, *A New Approach to Generating High Quality Test Cases*, *19th IEEE Asian Test Symposium*, 2010,pp.71-76.
14. F.Ricca and P. Tonella. *Analysis and Testing of Web Applications*. In: *Proc. of the International Software Engineering Conference*. Toronto Canada: IEEE Computer Society Press, May 2001, 25-34.
15. Andreas Bruns, Andreas Kornstädt, and Dennis Wichmann, *WebApplication Test with Selenium*, *IEEE Software*, 2009, pp.88-91.
16. C. A. R. Hoare. *An axiomatic basis for computer programming*. *Communications of the ACM*, 12(10):576–580,583 October1969.
17. R. W. Floyd. *Assigning meanings to programs*. *Proceedings of the American Mathematical Society Symposia on Applied Mathematics*. Vol.19, pp.19–31.1967.
18. Ghodeswar, Ajeet, and Y. Kulkarni. "Smeasy: A Gui Automation Framework On the Top of the Selenium." *International Journal of Computer Networking, Wireless and Mobile Communications (IJCNWMC)* 3.2 (2013): 103-112.

19. Kaur, Navneet, and Ashima Singh. "A Technique for Component Based Software Development." *International Journal of Computer Science and Engineering (IJCSE)* 2.2 (2013):123-134.
20. Aruna, Chittineni, and R. Siva Ram Prasad. "Development of an Automated Framework to Resolve Software Testing Issues." *International Journal of Computer Science Engineering and Information Technology Research (IJCSEITR)* 4.2 (2014):259-276.
21. Shingadiya, Chetan J., and Nitesh M. Sureja. "Nature Inspired Algorithms for Test Suite Optimization for Regression Testing: An Investigation." *International Journal of Computer Science Engineering and Information Techonology Research (IJCSEITR)* 8.2 (2018):61-68