Automated Software Testing Tools

Conference Paper · September 2020

DOI: 10.1109/ICCIT-144147971.2020.9213735

Shakeel Ahmed
Taylor's University
71 PUBLICATIONS **785** CITATIONS

Automated Software Testing Tools

Saja Khalid Alferidah
College of Computer Sciences and Information Technology
King Faisal University
Alahsa, Saudi Arabia
saja.alferidah@gmail.com

Shakeel Ahmed
College of Computer Sciences and Information Technology
King Faisal University
Alahsa, Saudi Arabia
shakeel@kfu.edu.s

***Abstract*— In the recent years, the researcher tried to find tools that could help in improving the quality, reliability, and performance of the software systems. Automated software testing tools are the solution in performing a complete system test without an error that would have led to the loss of money. Most of the tools of automated software testing reduce the time, increase the reliability and performance, and allow the tester to repeat and reuse the final product. This paper presents a description of automated software testing tools and specifies which tools are the best and more efficient.**

Keywords—*automated testing, automated testing tools, software system, software testing, verification, validation*

I. INTRODUCTION

Software testing is a software engineering activity to check whether the actual results of the software being developed matches with those of the expected results and to ensure that the software system is defect free. It involves the execution of a software component or system component to evaluate one or more properties of interest. Software testing is not a new field and it has appeared many years earlier, in software, we need to create a good and efficient software product or services, but this can be difficult because software must be tested before by stakeholders. A simple definition of software testing is the process of investigating software to check if it satisfies the requirement and detects errors that can happen in any software. “Another definition of software testing is the process of testing, verifying and validating the user’s requirements” [1]. Figure 1 shows the verification and validation activity of software testing.

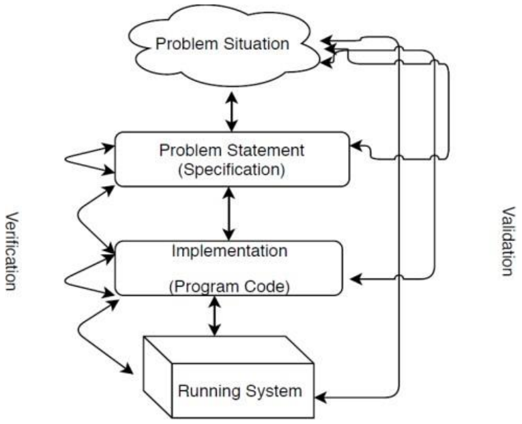


Fig. 1. Verification and validation activity of software testing [2].

Software validation and verification are software quality assurance activities. The meaning of validation is the answer to the question “are we building the right product” and the meaning of verification is the answer to the question “are we building the product right”. The goal of software verification and validation is to provide tools to check the correctness of the software. The amount of time required for testing software can be infinite. The goal of software testing is to get information about the quality, risks, error, defects, and bugs that can be present in the software.

Software testing can be done manually or automatically [1]. Figure 2 shows software testing. Manual testing is created by the testers. Automated testing or automated software testing is a test that can be done by software and run automatically. The goal of automated testing is to reduce the error that may be done by the user because users are slow and could make an error, so this leads to reduce the cost of software. There are many automated testing tools that can perform an automated test for software. A simple example of the automated testing tool in Java is JUnit.

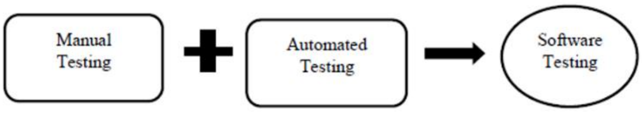


Fig. 2. Software testing [1].

The testing techniques that can be done by the developer could be divided into two types. First, the static testing that is done earlier in the development life cycle, it can be called a verification test. The static test could include inspection which is a formal type of review where the checklist is created to review documents works. The main goal of the inspection is to find the defects. The walkthrough is a meeting conducted by the author to explain the product. The participants can ask questions and a scribe is assigned to make notes. Technical reviews are created to check if the code created depends on technical specification and standards. In general, the test plans, test strategy, and test scripts reviewed in technical reviews. Informal reviews it is an informal review of the document and the informal comments are provided. Second, the dynamic testing that is done in the runtime environment; it can be called a validation testing. The dynamic test includes both functional testing and non-functional testing. The dynamic test could include unit testing which tests for individual modules by developers. The source code is tested in unit testing. Integration testing which in different modules tests the interface between them then they are joined. System testing provides a test for whole system. Acceptance testing which is done form the user's point of view at user's end. The testing techniques could divide into different types, which are a white box, a black box, and a gray box. The white box is a test which is useful at exposing coding mistakes. The black box is a functional test which is useful at exposing conceptual faults. The gray box is a test has little information about the structure, code, and system. The gray box commonly used in the integration test.

II. BACKGROUND

These days, globalization of automation is increasing especially for safety-critical applications for example, in the military system if an error occurs on the system it causes a major concern of losing more money and innocent human lives. There are many challenges for designing and development of automated testing tools such as it needs high performance, it must be easy to learn and effective tool. In addition, there is no standard or universal language for automated testing tools.

The automated software tools can be classified into three categories which are classification based on the mode of operation, classification based on the development phase, and classification based on the function of the tools [3]. There many software testing such as functional testing, load testing, and unit testing. Each type of test has it is own tool and some tools can be used for two types of testing.

Automated testing tools consist of two major steps to perform the testing and analysis of software. First, is to get all the software under test (SUT) and identify all the various files such as header file, input, output, functions and variables. Second, the user should select the desired modules or the function to perform the test and select the required test [4].

III. AUTOMATED SOFTWARE TESTING TOOLS

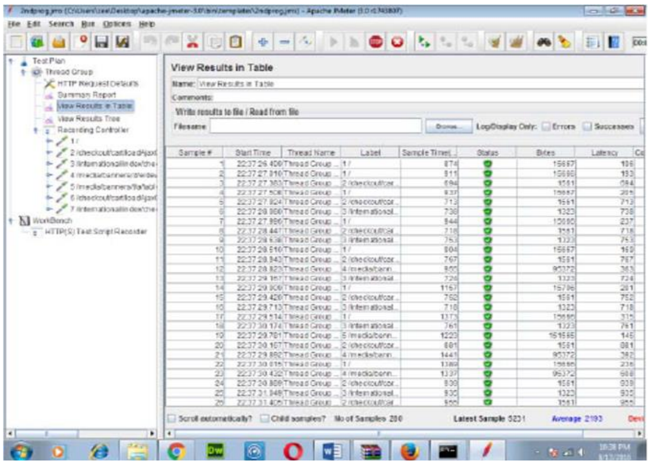
Software automated testing tools became available in the stores and ready to be used by people. There are many studies that are done related to automated testing tools, this section presents the study of some of the most common automated testing tools for software.

a. Load Testing Tools

Load testing is a type of testing that allows the tester to analyze the performance of the systems [1]. Automated load testing aims to find the functional and performance issue of any system under a load. Load testing is suitable for the websites and the frameworks. The load testing tools:

1) *Apache JMeter:* "Apache Software Foundation (ASF)" developed Apache JMeter which an open-source testing tool. The main job for this tool is to provide a load test of the client/server also, it is used for regression testing [1]. JMeter performs offline reporting. Apache JMeter contains many features for testing, gives an accurate result, performs GUI and very good for different tests [1]. On another hand, it required to set up [1]. Figure 3 shows a test report of JMeter.

2) *LoadRunner:* "Hewlett Packed enterprise developed HPE LoadRunner" which an automated testing tool. The main job of this tool is to identify bottlenecks that helps to detect and prevent issues of software performance [1]. LoadRunner is the best one for checking the performance and also check the network. It can run for different users at the same time [1]. On another side, it may have issues with the installation or configuration across firewalls [1]. Figure 4 shows a report summary of LoadRunner.


2ndprog.jmx (C:\Users\zee\Desktop\apache-jmeter-3.0\bin\templates\2ndprog.jmx) - Apache JMeter
File Edit Search Run Options Help
Test Plan
Thread Group
HTTP Request Defaults
Summary Report
View Results in Table
View Results Tree
Recording Controller
1 /
2 /checkout/cart/loadAjax
3 /international/index/che
4 /media/banners/slider
5 /media/banners/flatlabl
6 /checkout/cart/loadAjax
7 /international/index/che
WorkBench
HTTP(S) Test Script Recorder

View Results in Table

Name: View Results in Table

Comments:

Write results to file / Read from file

Filename: Browse... Log/Display Only: □ Errors □ Successes

Sample #	Start Time	Thread Name	Label	Sample Time(...	Status	Bytes	Latency
1	22:37:26.409	Thread Group ...	1 /	874	✓	15667	186
2	22:37:27.910	Thread Group ...	1 /	911	✓	15690	193
3	22:37:27.383	Thread Group ...	2 /checkout/car...	694	✓	1561	694
4	22:37:27.508	Thread Group ...	1 /	837	✓	15667	285
5	22:37:27.824	Thread Group ...	2 /checkout/car...	713	✓	1561	713
6	22:37:28.980	Thread Group ...	3 /international...	738	✓	1323	738
7	22:37:27.986	Thread Group ...	1 /	944	✓	15690	237
8	22:37:28.447	Thread Group ...	2 /checkout/car...	718	✓	1561	718
9	22:37:28.638	Thread Group ...	3 /international...	753	✓	1323	753
10	22:37:28.510	Thread Group ...	1 /	904	✓	15667	169
11	22:37:28.943	Thread Group ...	2 /checkout/car...	767	✓	1561	767
12	22:37:28.823	Thread Group ...	4 /media/bann...	955	✓	95372	363
13	22:37:29.157	Thread Group ...	3 /international...	726	✓	1323	724
14	22:37:29.909	Thread Group ...	1 /	1167	✓	15706	281
15	22:37:29.420	Thread Group ...	2 /checkout/car...	752	✓	1561	752
16	22:37:29.713	Thread Group ...	3 /international...	718	✓	1323	718
17	22:37:29.514	Thread Group ...	1 /	1173	✓	15695	315
18	22:37:30.174	Thread Group ...	3 /international...	761	✓	1323	761
19	22:37:29.781	Thread Group ...	5 /media/bann...	1223	✓	151565	145
20	22:37:29.167	Thread Group ...	2 /checkout/car...	881	✓	1561	881
21	22:37:29.892	Thread Group ...	4 /media/bann...	1441	✓	95372	382
22	22:37:30.015	Thread Group ...	1 /	1189	✓	15686	236
23	22:37:30.432	Thread Group ...	4 /media/bann...	1337	✓	95372	608
24	22:37:29.869	Thread Group ...	2 /checkout/car...	939	✓	1561	939
25	22:37:31.949	Thread Group ...	3 /international...	930	✓	1323	935
26	22:37:31.405	Thread Group ...	2 /checkout/car...	955	✓	1561	955

□ Scroll automatically? □ Child samples? No of Samples 280 Latest Sample 5231 Average 2193 Devi

Fig. 3. Test reporting of JMeter [1].

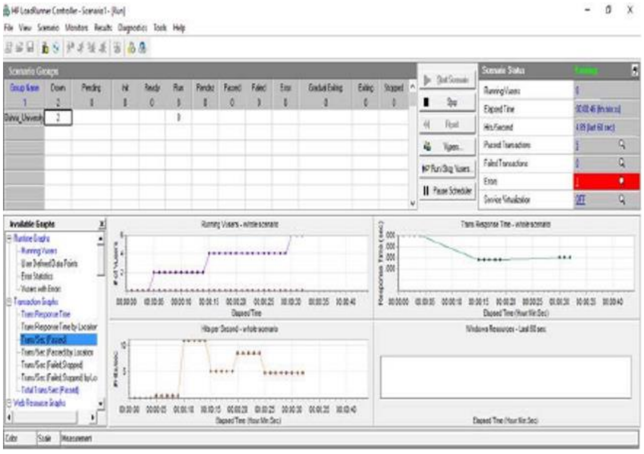
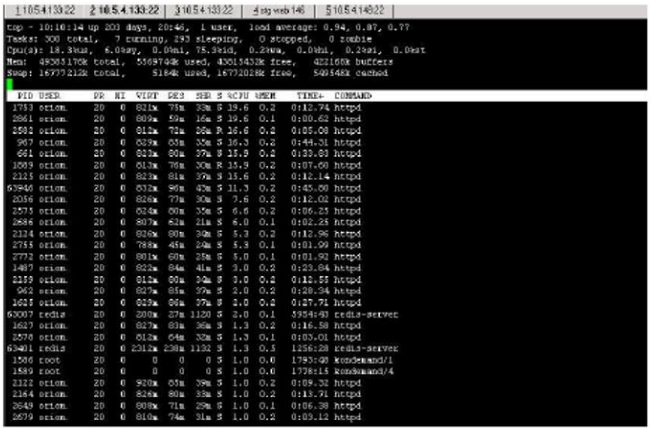


Fig. 4. Report summary of LoadRunner [1].

3) *Siege:* "Jeffery Fulmer" developed and implemented Siege. The main job for Seige is to detect the performance of the system when the load exists [1]. Siege is faster to set up and gives a quick result. On another hand, it may give an inaccurate result and it has limited options. Figure 5 shows the report summary of Siege.



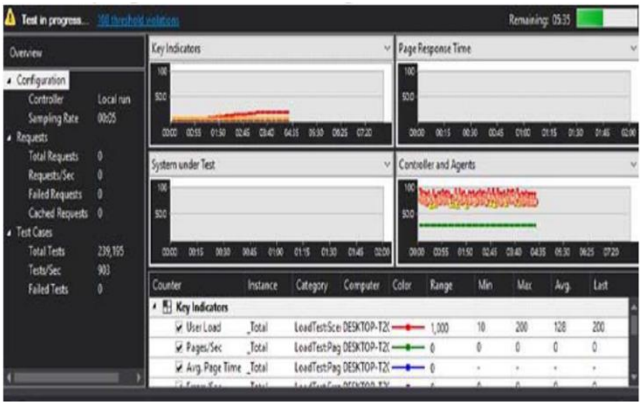
```
top - 10:10:14 up 203 days, 20:46,  1 user,  load average: 0.94, 0.87, 0.77
Tasks: 300 total,   7 running, 293 sleeping,   0 stopped,   0 zombie
Cpu(s): 18.3%us,  6.0%sy,  0.0%ni, 75.3%id,  0.2%wa,  0.0%hi,  0.2%si,  0.0%st
Mem:  49385176k total,  5569744k used, 43815432k free,   422168k buffers
Swap: 16777212k total,     5184k used, 16772028k free,   599548k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1753 orion     20   0  821m  75m  33m S 19.6  0.2   0:12.74 httpd
 2861 orion     20   0  809m  59m  16m S 19.6  0.1   0:00.62 httpd
 2582 orion     20   0  812m  72m  26m R 16.6  0.2   0:05.08 httpd
  967 orion     20   0  829m  65m  35m S 16.3  0.2   0:44.31 httpd
  661 orion     20   0  823m  80m  37m S 15.9  0.2   0:33.83 httpd
 1089 orion     20   0  813m  76m  30m R 15.9  0.2   0:07.60 httpd
 2125 orion     20   0  823m  81m  37m S 15.6  0.2   0:12.14 httpd
63946 orion     20   0  832m  96m  43m S 11.3  0.2   0:45.80 httpd
 2056 orion     20   0  826m  77m  30m S  7.6  0.2   0:12.02 httpd
 2575 orion     20   0  824m  80m  35m S  6.6  0.2   0:06.25 httpd
 2686 orion     20   0  807m  62m  21m S  6.0  0.1   0:02.25 httpd
 2124 orion     20   0  826m  80m  34m S  5.3  0.2   0:12.96 httpd
 2755 orion     20   0  788m  45m  24m S  5.3  0.1   0:01.99 httpd
 2772 orion     20   0  801m  60m  25m S  5.0  0.1   0:01.92 httpd
 1487 orion     20   0  822m  84m  41m S  3.0  0.2   0:23.84 httpd
 2159 orion     20   0  812m  80m  34m S  3.0  0.2   0:12.55 httpd
  962 orion     20   0  827m  85m  37m S  2.0  0.2   0:28.34 httpd
 1625 orion     20   0  829m  86m  37m S  2.0  0.2   0:27.71 httpd
63007 redis     20   0  200m  27m 1120 S  2.0  0.1   3954:43 redis-server
 1627 orion     20   0  827m  83m  36m S  1.3  0.2   0:16.58 httpd
 2578 orion     20   0  812m  64m  32m S  1.3  0.1   0:03.01 httpd
63401 redis     20   0 2312m 238m 1132 S  1.3  0.5   1256:28 redis-server
 1586 root      20   0     0    0    0 S  1.0  0.0   1793:40 kondemand/1
 1589 root      20   0     0    0    0 S  1.0  0.0   1778:15 kondemand/4
 2122 orion     20   0  920m  65m  39m S  1.0  0.2   0:09.32 httpd
 2164 orion     20   0  826m  80m  33m S  1.0  0.2   0:13.71 httpd
 2649 orion     20   0  808m  71m  29m S  1.0  0.1   0:06.38 httpd
 2679 orion     20   0  810m  74m  31m S  1.0  0.2   0:03.12 httpd
```

Fig. 5. Report summary of Siege [1].

4) *Microsoft Visual Studio (TFS):* Team Foundation Server (TFS)" is a tool for load testing that performs help to the code, project, requirement, lab management, reporting, testing capabilities, and automated builds [1]. It is very simple for use, but it supports Windows Operating

System (OS) only [1]. Figure 6 shows the report summary of TFS.



Test in progress... 398 threshold violations — Remaining: 05:35

Overview	
Configuration	
Controller	Local run
Sampling Rate	00:05
Requests	
Total Requests	0
Requests/Sec	0
Failed Requests	0
Cached Requests	0
Test Cases	
Total Tests	239,195
Tests/Sec	903
Failed Tests	0

Counter	Instance	Category	Computer	Color	Range	Min	Max	Avg.	Last
Key Indicators									
☑ User Load	_Total	LoadTestSce	DESKTOP-T2(		1,000	10	200	128	200
☑ Pages/Sec	_Total	LoadTestPag	DESKTOP-T2(		0	0	0	0	0
☑ Avg. Page Time	_Total	LoadTestPag	DESKTOP-T2(		0	-	-	-	-

Fig. 6. Report summary of TFS [1].

b. *Acceptance Testing Tools*

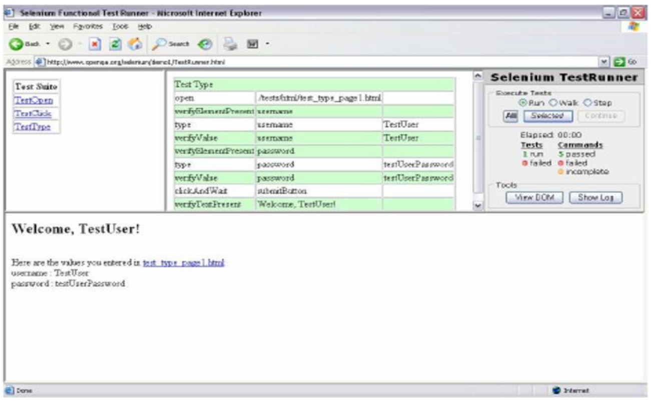
Acceptance testing is the type of test provided to check if the software system meets the requirement they are used to validate if the system delivers the required functionality or not [5]. There are many tools for acceptance testing, but the most commonly used is the "FIT (Framework for Integrated Test)" [5].

1) FIT: FIT tool depends on JUnit, it is created by clients and developers. The clients write the test cases and classes, then automatically generates the acceptance testing [5].

c. *Functional Testing Tools*

Functional testing is the type of test to validate if the website or web application correctly perform all the required functions. Functional testing tools such as Selenium which is the most popular tool used and the other tool is FitNesse.

1) Selenium: Selenium is an automated functional testing tool. It uses simple scripts to run the test directly in the browser [6]. It can run on different platforms. It allows the tester to edit, record and debug tests [7]. Selenium is an easy setup tool. Figure 7 shows the Selenium tool windows.



Selenium Functional Test Runner - Microsoft Internet Explorer

Address: http://www.openqa.org/selenium/demo1/TestRunner.html

Test Suite
TestOpen
TestClick
TestType

Test Type		
open	/tests/html/test_type_page1.html	
verifyElementPresent	username	
type	username	TestUser
verifyValue	username	TestUser
verifyElementPresent	password	
type	password	testUserPassword
verifyValue	password	testUserPassword
clickAndWait	submitButton	
verifyTextPresent	Welcome, TestUser!	

Selenium TestRunner

Execute Tests: ● Run ○ Walk ○ Step

Elapsed: 00:00

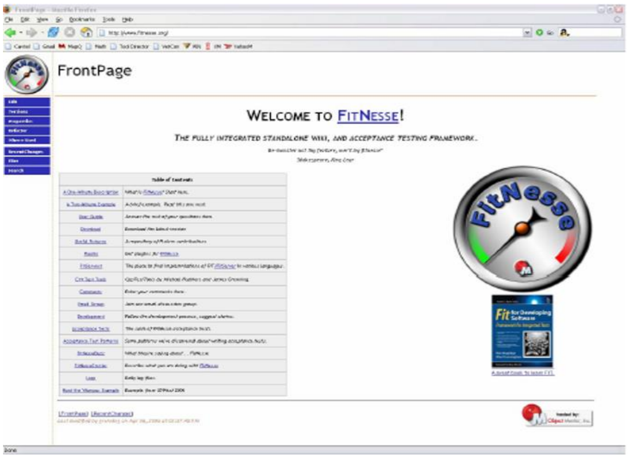
Tests	Commands
1 run	5 passed
0 failed	0 failed
	0 incomplete

Welcome, TestUser!

Here are the values you entered in test_type_page1.html
username : TestUser
password : testUserPassword

Fig. 7. Selenium [6].

2) FitNesse: FitNesse an automated testing tool, wiki, and web server all rolled into one application [6]. It is used for functional and acceptance testing. It allows the tester, programmer or customer to know that their software should do and provide an automatic comparision to what it does [6]. FitNesse saves a copy for all pages and of every version. Figure 8 shows the FitNesse tool window.



FrontPage

WELCOME TO FITNESSE!

THE FULLY INTEGRATED STANDALONE WIKI, AND ACCEPTANCE TESTING FRAMEWORK.

Fig. 8. FitNesse [6].

d. *Regression Testing Tools*

An automated regression test is similar to an automated functional test that validates the functionality of the system and checks if the new added functionality of the system does not make an error or bugs to the system. Selenium can be a tool for regression testing.

1) IBM Rational Functional Tester: "IBM Rational Functional Tester" is an automated regression and functional testing tool [7]. It uses Java language and Excel to store data and use them in a web application to be tested [7].

2) Quick Test Professional (QTP): QTP it provided by HP/Mercury Interactive [7]. It is an automated regression testing tool. It uses visual basic (VB) language. QTP can be used in both manual testing and automated testing [7].

3) Sahi: Sahi also is an automated testing tool. It uses a web application. It was developed in Java and Javascript and hosted on SoundForge [7]-[8]. It applies to record and plays back the scripts [7].

e. *Graphical User Interface(GUI) Testing Tools*

GUI testing is a process to validate the functionality of GUI of the system and check if it related to system specification. Testing the GUI can be very complex. Automated GUI test through capture and relay technique (CR) [9]. There is a sequence of CR tools that are used to perform automated GUI testing include the "popular Quick Test Professional (QTP), Abbot, Selenium, and Rational Functional Tester (RFT), Win runner, SilkTest, and IBM Rational Robot" [9].

f. *End to End Testing Tools*

End-to-End automated testing allows to discover regressions at the beginning of the development and provide a solid basis for the future modifications of the systems [10]. This means the End-to-End testing is various type of testing that provides a check the correctness of application the application flow from the users' point of view [10]. Selenium is a tool for End-to-End another example is Robotium [10].

1) Robotium: Robotium is an automated End-to-End testing tool. It provides steps for the developer on how they can solve testing tasks [10].

IV. DISCUSSION

In an automated software testing tool, many popular open-source testing tools can be used for two types of test. For example, load testing includes two types of testing performance and stress. Each tool has it is own function, feature, and usability although the foundation function of these tools is the same. These tools are different only in architecture. Selenium and SAHI are the most tools using for testing.

V. CONCLUSION

In the software development life cycle, software testing is the most important part. In every organization improve the quality and performance of the software. It is becoming a goal for most of the organization and this relay on software development because it saves both the time and cost of software development also help to perform good satisfaction for the customers, increase efficiency, effectiveness and the coverage of software testing. A good automated testing tool is that can be easy to use and learn by the users. In the future, the use of automated software tools will increase especially if the system runs at maximum capacity.

ACKNOWLEDGMENT

We would like to express our special thanks to King Faisal University for all their assistance and support.

REFERENCES

[1] R. Abbas, Z. Sultan, and S. N. Bhatti, “Comparative analysis of automated load testing tools: Apache JMeter, Microsoft Visual Studio (TFS), LoadRunner, Siege,” 2017 International Conference on Communication Technologies (ComTech), pp.39-44, 2017.

[2] Vishawjyoti, and P. Gandhi, “A survey on prospects of automated software test case generation methods,” 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), pp. 3867–3871, 2016.

[3] C. V. Ramamoorthy, and S. F. Ho, “Testing large software with automated software evaluation systems,” in IEEE T. Softw. Eng., Vol. SE-1, No. 1, pp. 46–58, 1975.

[4] R. Khalid, “Towards an automated tool for software testing and analysis,” 2017 14th International Bhurban Conference on Applied Sciences and Technology (IBCAST), pp. 461–465, 2017.

[5] N. Negara, and E. Stroulia, “Automated Acceptance Testing of JavaScript Web Applications,” 2012 19th Working Conference on Reverse Engineering, Kingston, pp. 318–322, 2012.

[6] A. Holmes, and M. Kellogg, “Automating functional tests using Selenium,” in Proceedings of the conference on AGILE 2006 (AGILE ’06), pp.270–275, 2006.

[7] L. Nagowah and P. Roopnah, “AsT-A simple automated system testing tool,” 2010 3rd International Conference on Computer Science and Information Technology, pp. 301–306, 2010.

[8] T. J. Naidu, N. A. Basri, and S. Nagenthram, “SAHI vs. Selenium: A comparative analysis,” 2014 International Conference on Contemporary Computing and Informatics (IC3I), pp. 967–970, 2014.

[9] Y. Miao, and X. Yang, “An FSM based GUI test automation model,” 2010 11th International Conference on Control Automation Robotics and Vision, pp. 120–126, 2010.

[10] A. Vasilyev, I. Paramonov, and S. Averkiev, “Method and tools for automated end-to-end testing of applications for sailfish OS,” 2017 20th Conference of Open Innovations Association (FRUCT), pp. 471–477, 2017.