Volume- 6
Issue- I
June-2021

Rajarata University Journal

ISSN: 2362-0080
www.ruj.ac.lk/journals/

Category: Research Article

A Keyword Driven Hybrid Model for Web Application Testing

[1]Madurapperuma, Dilan, [*2]Walisadeera, Anusha Indika & [3]Goonetillake, Jeevani

[1]*Axiata Digital Labs, Parkland Level 11, 33 Park St, Colombo 2, Sri Lanka*

[*2]*Department of Computer Science, University of Ruhuna, Matara, Sri Lanka*

[3]*University of Colombo School of Computing, Colombo 07, Sri Lanka*

ARTICLE DETAILS

Article History
Published Online: 30 June, 2021

Keywords
Software Development, Testing Tools, Automated Testing, Manual Testing, Web Testing.

[*]**Corresponding Author**
Email:waindika@dcs.ruh.ac.lk

ABSTRACT

Web development is one of the fast-growing techniques in the IT field. To ensure error and bug-free web products, they should be tested thoroughly. If both Manual and Automated Testing Techniques (MATT) are used concurrently, this goal can be accomplished. When MATT is considered individually, there are positive and negative factors. In this paper, a keyword-driven technique is proposed which helps to bring MATT together to solve the problem of *"How to ensure the high quality of modern web projects with highly redundant tasks at lower cost and less amount time?"* According to the current literature, manual testing is essential to ensure error and bug-free web products, while automated testing fulfills the time target of the quality assurance process. The main objective of this research is to propose *a keyword-driven web testing method* to ensure error and bugs free web products. The proposed method was tested with modern websites. Results showed that the proposed method considerably minimizes the average time while identifying the errors of test scripts. The proposed method should work for every modern web development technique; it also should be a sustainable solution to be adapted to future changes in the web development industry.

1. **Introduction**

Before starting the software development, we should manage three factors. They are *Quality*, *Time,* and *Cost,* the most important factors in software development [1]. One way to ensure the quality of a web product is through testing. Testing a web page means that checking whether the server's response contains what the client requested. Web application development is one of the rapidly growing areas in the IT field. Since web development is frequently modified, the main goal of any quality assurance engineer is to create sustainable tests [2]. Two approaches are used to carry out the web application testing process mainly. They are: *manual testing* and *automated testing*.

Manual testing is a testing process where all the activities are done by humans manually. "Manual testing can be described as a process where a person initiates each test, interacts with it, and interprets, analyses and reports the results" [3]. Automated testing is the testing process where human initiates the testing, but, all the other activities are done by software. Automated testing is a system that uses different data sets, again and again, to test without human interaction [4]. Both types have positive and negative factors. Reliability, repeatability, comprehensibility, reusability, quality, speed, low cost (it reduces the long-term costs), and high return on investment are the advantages of automated testing [5]. It has also been mentioned that the disadvantages of automated testing such as high initial cost, inability to automate some test cases, the requirement of high skills to write the automation test cases, and the necessity to change the whole test case with respect to a small change in API [5]. They further have also specified the advantages of manual testing. They are: if the test case only needs to run only a few times, it is better to use manual testing, and as such, the cost can be reduced. It allows the tester to perform more random testing, and then more bugs can be found. Short-term costs could be reduced, and when testers spend more time testing a module, then the odds of finding bugs are high. Some disadvantages of manual testing are also mentioned, such as very time-consuming; the necessity to repeat the same testing you previously done for every release; not suitable for large projects; some test cases can be missed by repeating because the tester forgets it; and human errors can happen.

There are some scenarios in the web development industry that need both manual and automated testing features. For example, in **short-term projects with highly redundant tasks**, the tester needs a combination of manual and automated testing. Many research studies in this field are trying to address this issue. However, it still is remaining as a problem in the industry. The main objective of this research study is to propose a hybrid model that is compatible with every scenario. The testing can be classified into two parts, as mentioned above. The question is: "When to automate and when not to automate?" Although the authors of [6], [7], and [8] have answered the above question up to a certain extent, some situations still need to be addressed. For example, there are short-term projects with many redundant tasks. According to the research studies mentioned in ([6], [7], and [8]) manual testing is recommended for short-term projects. On the other hand, if there are many redundant components, then automated testing is preferred.

When we consider the cost of testing time for manual and automated testing (see Figure 1), automated testing is rather beneficial, although its starting cost is high. This initial cost would not be a burden when considering the benefits of automated testing [4].

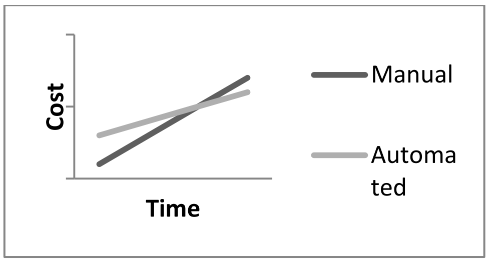


Figure 1: The cost of testing vs. time [4]

Some researchers have proposed some approaches to incorporate manual and automated testing or to mitigate the disadvantages of automated testing techniques, however, with some drawbacks. There is also a new concept that is keyword-driven test automation technique which is one of the main adopted frameworks in software industries [1]. The keyword-driven testing technique is similar to programming but is not as complex as programming. In the keyword-driven testing technique, the steps required to be performed in testing are given as a sequence of instructions. These instructions consist of a sequence of keywords. This approach is not hard to understand, and as such, it is easy to train testers. Keyword-driven automation technique consists of three major parts: *Test data*, *Test libraries*, and *Test automation framework* [1]. Test data represents the input and corresponding output [1]. Test libraries represent the interface of the system under test and the whole framework [1]. Test automation framework is for reading the test data, executing the test, and handling the errors when the tests are under execution [1]. The main reason to use the keyword-driven test automation technique is, it has some key benefits that make the testing process easier [9]. Some benefits are:

- easy automation process - a manual tester with zero scripting knowledge can quickly automate a test case
- easy maintenance of the scripts
- efficiency - the overall testing process takes less time than manual testing due to the availability of open-source
- cost-effectiveness - it will reduce the cost of testing by using the open-source tools
- easily understandable - business users can also join to test and analyze test execution results with the help of a keyword-driven test automation framework because it is easy to understand the process
- possibility of iterative testing - at the end same test case can be performed again and again to test the consistency of the application, and also the tester can perform various actions (like recursive method callings) [9].

Despite the above advantages, keyword-driven test automation does not combine manual and automated testing and instead tries to mitigate some of the disadvantages of automated testing. In Jain and Sharma's approach, they did not implement for testing domain-specific tasks [1]. By using this approach, it is not possible to handle short-term projects with highly redundant tasks. Jain and Sharma's approach does not implement to combine both manual and automated testing [1]. As mentioned above, in the literature, we can find research work that attempts to combine manual and automated testing by proposing a set of guidelines. To this end, two separate teams are proposed to be allocated for manual and automated testing. Both teams carry out their tasks separately. SmartBear Software introduces a new set of best practices in uniting manual and automated testing [10]. In this approach, although they combine manual and automated testing techniques, the disadvantages of both techniques would remain the same since two separate teams would perform the two techniques by following the corresponding procedure applicable to each technique. For example, while developer testing is performed, they also propose to start the automated testing process [10]. "Developer testing is a type of testing where developers test their code as they write it, as opposed to testing done by a separate quality assurance organization" [11], [12]. This approach enables to obtain better accuracy and achieve time targets [10], even if the disadvantages of the two techniques remain the same.

We propose a new hybrid method of manual and automated testing with a low cost through this research work. If we have a tool while automated test result is being prepared, we can check problematic scenarios manually, then, most of the questions are solved. Because of the lack of such methods and tools, we introduce this method as a free and open-source tool. Therefore, we can customize this tool to facilitate organizational needs.

2. Methodology

We have selected the action design methodology in this research since it allows designing the research iteratively [13]. In the action research design, we have identified four phases. First, we need to understand the problem, plan, and develop a solution for the problem identified in the corresponding iteration. Then, we need to evaluate and validate the implemented solution correctly addresses the problem.

Understand the problem and requirements: We analyzed the problem and requirements based on the meta-Analysis approach [14]. Meta-analysis is an analytical methodology designed to systematically evaluate and summarize the results from previous research, books, and other domain-related materials. Meta-analysis aims not only to summarize existing knowledge but also to develop a new understanding of the research problem using scientific reasoning. We analyzed several research dissertations, online articles, and other domain-related materials to analyze the problem and gather requirements. As a result of the meta-analysis, we identified the existing problems in the domain of web testing, and we explored the proposed solutions to this problem, which then lead to the design of the solution using the keyword-driven approach.

Design a solution: Following the analysis of the problem and requirements, we designed a plan to implement the solution. Based upon the knowledge that we acquired through Meta-Analysis, we propose a solution, which can be adopted in existing automated testing tools. The basic idea of this design is to combine manual and automated testing where human testers, who are going to test the web pages using this approach, need to be present even when an automated test is executed. It means the tester should check the progress of the automated test. For that, the proposed solution should ask for confirmation from testers similar to when manual testing is performed, but here simultaneously with automated testing.

The outline of the solution design consists of several core activities. They are as follows:

1. Starting of the testing process
2. Initial steps of keyword-driven testing: script conversion

   2.1. Manual confirmation

   2.2. Automated confirmation
3. Report generation according to both result

Starting of the testing process refers to the initial tasks that should be performed during the keyword-driven testing. For example, before the test is begun, testers should create a test script. *The initial step of keyword-driven testing* is the script conversion process. It means script language needs to translate the test activity sequence. Then, we come to the most crucial part of the research. During the test execution, the web page should be *manually and automatically checked and confirm*. Finally, the result of the tested scenario should be written into a report.

Develop the solution: When we analyzed the capabilities of implementing the solution, we identified that the java runtime environment to run selenium has full capabilities to implement our solution. To implement the proposed solution, we use the empirical method. The proposed solution is to combine the two main testing approaches. Therefore the development of the solution focus on implementing the activities described previously while the existing testing tools deal with the automated testing activities. After reviewing the current literature on the field, we selected the selenium testing tool to implement our tool. Selenium is a free and open-source testing tool [15], [16]. So, it does not impose a cost to acquire the source and enables the implementation of the proposed solution as a free and open-source tool. More importantly, being an open-source tool, selenium helps the implementation of the proposed hybrid approach on top of it. Since this tool is developed using Java technologies, the activities of the proposed solution are developed using Java programming language and integrated with the selenium tool.

Evaluation and Validation: We have decided to validate and evaluate the approach by using the tool that we produced. We accomplish that task by testing the web applications using our tool. We selected few test scenarios of some web applications to test the tool based on different types of operations that frequently occur in them. The requirements of those test scenarios are known. The selected web applications include a management information system, a web search engine and dynamic websites. Following are the list of websites and web applications that we used for the evaluation:

1. Management Information System of Faculty of Science, University of Ruhuna (http://paravi.ruh.ac.lk/fosmis/)
2. Google search engine (https://www.google.lk/)
3. University of Ruhuna web site (http://www.ruh.ac.lk/)
4. Faculty of Science web site (http://www.sci.ruh.ac.lk/)

Requirements of the above web applications are known. So we can perform the test cases on them and validate the results with the actual requirements of those websites. Using this approach, we evaluate and validate our solution.

We did the implementation of our research in two iterations. In the first iteration, we implemented the tool without the report generation part. Then, we identified the pitfalls of that millstone. Then, we analyzed the problems and started the next iteration. While addressing those identified pitfalls, we also developed the report generating part in the second iteration.

3. Results and Discussion

As a result of our research, we have introduced a keyword-driven testing technique. This technique has both manual and automated testing features. Figure 2 shows the workflow of the proposed method. In the figure, we can see the web applications are checked in two stages (see 5, 5.1, 5.2). One is automated and the other is manual. Therefore, in this approach, we can identify the errors of the test scripts. Because of this approach, the result of automated testing is checked manually.

The activities that need to be performed in this approach are listed below:

1. Start of the testing process
2. Program reads the script and takes activities one by one
3. Program converts it into javascript
4. Then the program runs those javascript on the selenium server
5. Program check for success condition
   5.1. If it fails, then ask the tester to confirm, and the program will write the error report
   5.2. If it passes, then ask the tester to confirm, and then the program will check for the next testing activity
6. If there is a conflict between automated results and manual confirmation, then report it and break the test.
7. Take the next activity and repeat from '2' with the until the last testing activity
8. Program writes 'pass' details to buffer
9. Program writes 'fail' details to buffer
10. Program writes details of the buffer in to report

To demonstrate these activities, we developed a tool. The activities that perform by the tool and tester can map with the above workflow as below.

Activity 1 – Start of the testing process: This belongs to a set of initial activities. The tester should initiate the automated testing by creating a test script. Then tester should start the proposed tool and select the test script (you can choose a new test script by file-> new test). The following three activities belong to the automation tool, which performs those activities automatically.

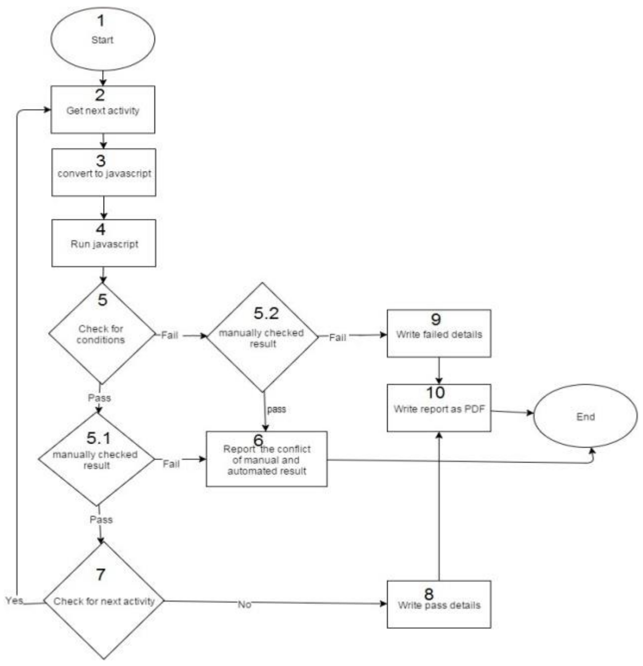


Figure 2: The Workflow Diagram of the Proposed Method

Activity 2 – Read the script and take activities one by one: According to Milad's and the team [17], the test script can be considered as the data source of automated testing. Test scripts consist of a sequence of activities that need to be performed by an automation tool during automated testing. What test tools do is they get those activities one by one and run it. The tool automatically fetches the testing activities one by one.

Activity 3 – Convert test script into javascript: This activity is also done by the tool. Each activity in the test script that is written using keywords is converted to javascript and pass to the next activity.

Activity 4 – Run the javascript on the selenium server: It is in this place where the testing activities actually started to be performed. After running one activity, the tester can see the result in his/her eyes (see Figure 3: open Browser Firefox http://paravi.ruh.ac.lk/fosmis/).

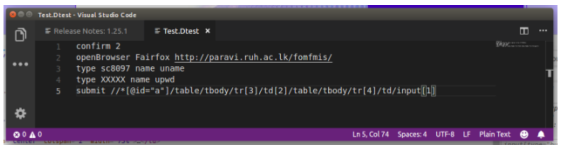
Activity 5 – Check for success condition: After the previous activity, the tool automatically checks the 'pass' or 'fail' condition. In most of the case, the pass condition is the ability to move to the next activity. For example, the pass condition of the second activity of Figure 3 is "there should be an HTML element (text box) with the value of its name attribute as 'uname.'" If there is no HTML element (text box) in the resulting page, the tool considers it a failed activity.

Activity 5.1 and 5.2 – Manual checking: As there can be errors in test scripts, the tool asks the tester to confirm the results using a yes/no popup window. So the tester will select if he/she is confident about the correctness of test scripts. Otherwise, the tester should check it manually.

Activity 6 – Conflict between automated results and manual confirmation: If there is a conflict between the automated and manual testing results, the tool will stop the execution and report that conflict of manual and automated test results. Here, the tester needs to check the test script for errors.

Activity 7 – Checking for the next activity: The tool fetches each activity one by one until the last activity.

Activity 8, 9, and 10 – Writing 'pass' or 'fail' details: If one of the activities in the test script fails or all the activities are passed in both manual and automate results, then the tool writes the passed or failed details to buffer before it writes to report. Then the tool automatically writes the passed or failed result to PDF (Portable Document Format) file.


Test.Dtest - Visual Studio Code
Release Notes: 1.25.1
Test.Dtest
Ln 5, Col 74 Spaces: 4 UTF-8 LF Plain Text

```
confirm 2
openBrowser Fairfox http://paravi.ruh.ac.lk/fomfmis/
type sc8097 name uname
type XXXXX name upwd
submit //*[@id="a"]/table/tbody/tr[3]/td[2]/table/tbody/tr[4]/td/input[1]
```

Figure 3: Sample Test Script

When using this tool to test their web products, people need to create test scripts. These scripts should save with ".Dtest" extension. The scripting language consists of only eight keywords. These keywords are selected to support most of the actions that occur when testing a web application. In the frontend web application, all common activities involve with a web browser and the html elements in a particular web page. Combining these keywords, one can accommodate most of the test scenarios. Usage and the details of the keywords are given in Table 1.

Table 1: The Keywords of the Proposed Method

Keyword	Usage	Arguments and details
Confirm	Get user confirmation after a given number of activities	Takes one integer argument. Informs the tool the activities where the program need to get the manual confirmation. Example: confirm 2
OpenBrowser	Open browser with given URI openBrowser <URI>	Takes one string argument. Informs the tool to open the browser with a given URL. Example: openBrowser http://paravi.ruh.ac.lk/fosmis/
Type	Enter a given value to a given target element	Takes three arguments: the first argument gives a text value, followed by a supportive keywords and the value of either the name or xpath of the element. Indicates that the value given by first argument should be set to the element specified by the latter arguments. Example, type sc8097 name uname
Name	Name of the target element	Takes one string argument: a string value to indicate the name of a target element. A supportive keyword that does not perform any activity, if used alone.
Xpath	Xpath of the target element	A supportive keyword similar to name. Takes one string argument to indicate the xpath of a target element. ("*XPath uses target expressions to select nodes or node-sets in an XML document*" [18])
Click	Click the given element	Takes two arguments that specify the target element: either of the supportive keywords name or xpath followed by a string value to give name or xpath of the element. Example: click name searchButton
Submit	Submit the form by clicking the given element	Takes two arguments that specify the target element: either of the supportive keywords name or xpath followed by a string value to give name or xpath of the element. Example: submit name submitElement

The reason for getting wrong results in automated testing is the complexity of the test automation process [3], [5], [19]. Due to this complexity, it is highly probable that testers make mistakes. Scripting languages like javascript are hard to understand to inexperienced persons and non-programmers. As a solution to this, we proposed the above keyword-driven technique. It is easy to understand, and no need to have deep knowledge of scripting. The primary issue in test automation is debugging. If there are any errors in the test script, it may cause severe consequences [7]. In our approach, a test case is a sequence of activities. We ask from tester to confirm whether the test case is successful or not. Both automated the result and manual confirmation should match. Otherwise, it will break the test case and report the situation so that the above problem can be solved. Consider a scenario where a short-term project has many redundant components. It would be more beneficial if manual testing is used [4], [5]. It is better to use automated testing to test redundant components [4], [5]. Because the proposed scripting language is elementary, we can create test scripts easily and can get both manual and automated testing features to test the project.

4. Validation and Evaluation

To validate our approach, we performed some test cases on four websites (with known requirements) using our tool. The test scenarios that correspond to those websites are as follows:

- Logging and logout of Management Information System web site of Faculty of Science, University of Ruhuna
- Functional workability of the search button of Google search engine
- The search facility of the University of Ruhuna website
- The correctness of link "Learning Management System (LMS)" of the University of Ruhuna - Faculty of Science, website.

We have recorded the following factors of each testing:

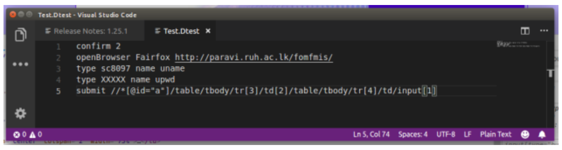
- *Time for testing:* Time that the tool took for a test run given test script
- *Accuracy:* To measure the accuracy, we used a marking system. That is, each activity that executes correctly was assigned with one (1) mark, and for each activity that runs wrongly, it was assigned with a minus one (-1) mark.

Test 1: Login and logout of Management Information System web site of Faculty of Science, University of Ruhuna

Following are the requirements (sequence of activities that users should be able to do) to login to the Management Information System website of the Faculty of Science, University of Ruhuna (http://paravi.ruh.ac.lk/fosmis/).

- The user should be able to open the browser with the URL of http://paravi.ruh.ac.lk/fosmis/
- The user should be able to fill the user name (for example, sc7987) in the textbox that the value of uname in the name attribute.
- The user should be able to fill the password (for example, xxxxxxx) in the textbox that the value of upwd in the name attribute.
- The user should be able to click the button that xpath value is
- //*[@id="a"]/table/tbody/tr[3]/td[2]/table/tbody/tr[4]/td/input [1]

Figure 4 shows the test script that belongs to evaluate this site. The tool assumes the correctness of each activity by the capability of doing the next activity. For example, after the execution of "openBrowserFirefox." http://paravi.ruh.ac.lk/fosmis/tool" activity, if it is successfully completed, there should be a text field to fill user name.



```
confirm 2
openBrowser Fairfox http://paravi.ruh.ac.lk/fomfmis/
type sc8097 name uname
type XXXXX name upwd
submit //*[@id="a"]/table/tbody/tr[3]/td[2]/table/tbody/tr[4]/td/input[1]
```

Figure 4: Test script for test FOSMIS (Test 1)

Test Results are as follows:

Number of activities (n) = 5

Time taken to test (t) = 35 seconds

Average time per activity (T) = t/n = 35/5

= 7 second/activity

Accuracy = 5 (5 out of 5 activities correctly)

In order to demonstrate what happens when we make a mistake in a test script, we change the test script incorrectly and follow the same test. Then, we are able to detect the error of the test script.

The values of above parameters using basic selenium tool are:

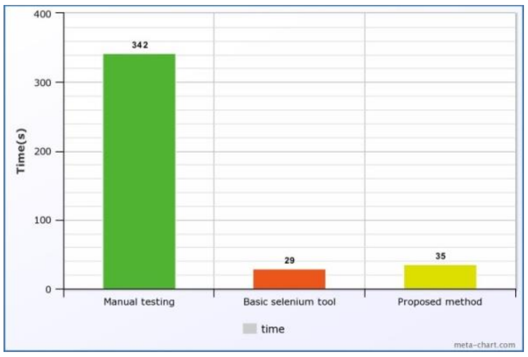
Time taken to test (t) = 29 seconds

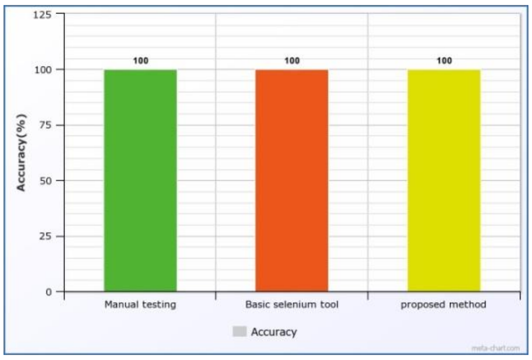
Average time per activity (T) = 29/5

= 5.8 second/activity

Accuracy = 5 (5 out of 5 activities correctly done)

In order to demonstrate a scenario when there is a mistake in a test script, we change the test script incorrectly and follow the same test. But here, the tool reports it as a bug of the website.





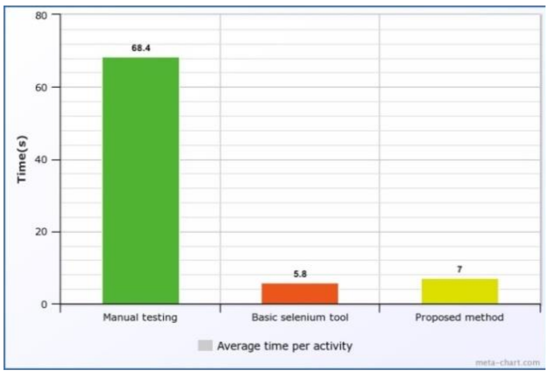


Figure 5: The summary of the results of Test 1.

The value of the above parameters using the manual testing technique is:

Time goes to test = 342 seconds (Here, nearly 250 seconds is taken to create the report).

Average time per activity = 342/5 = 68.4 second/activity (T)

Accuracy = 5 (5 out of 5 activities correctly done)

In Figure 5, these results of Test 1 are shown as graphs.

We did the same procedure for the other three scenarios as well and obtain the similar results. Due to time restrictions, we only evaluated the new approach for short test scenarios. In this research, we did not test the proposed solution for extended test scenarios. In industry, there can be test cases that might be very long and need to be executed regularly. Although we received 100% accuracy, in industry, actual scenarios might get lesser accuracy due to the complexity of the test cases.

5. Conclusion

The main goal of this research study is to combine manual and automated testing together to overcome the existing problems in software testing in short-term web projects with highly redundant tasks. The manual testing gives more short-term benefits, while automated testing gives more long-term benefits. As mentioned above, we can gain several benefits from the proposed hybrid approach. As we check twice (both manually and automatically), we can achieve more accuracy in this approach. The causes for delays in this tool are user input delay (as the tool waits for user confirmation) and delays in converting test scripts to runnable test commands. In this approach, it is easy to identify the errors in test scripts. Here we avoid the use of any commercial tool, and also, the test engineers do not require to have additional skills. Therefore, the total cost with the time will be similar to the manual testing graph in Figure 1. More importantly, since we reduced the testing time and the proposed method will save the cost than the manual testing. The keyword-driven testing technique is one of the leading modern testing techniques that can reduce the complexity of the test automation process. As this approach is based on keywords, it is easier to write and understand the scripts than other scripting languages. Using this tool would be more beneficial when the testing team consists of inexperienced testers as they tend to make more mistakes as well. It is also easy to get familiar with this tool. We have implemented the tool based on the selenium test tool. It is one of the leading web test automation tools that are freely available. We have planned to continue this research to implement a tool combining MATT without depending on other testing tools. Then, we can increase the neutrality of the proposed keyword-based language to support the web testing industry as well as customers of the web development industry to test their web products.

References

1. Jain A, Sharma S. an Efficient Keyword Driven Test Automation Framework for Web Applications. International Journal of Engineering Science & Advanced Technology. 2012; (3):600–4.

2. Ranstrom R. Automated Web Software Testing with Selenium [Internet]. Department of Computer Science and Engineering, University of Notre Dame, Notre Dame; 2010. Available from: https://www3.nd.edu/~veoc/resources/Papers/SeleniamPoster.pdf (Accessed 06 Jun 2021)

3. Chmurciak D. Automation of regression testing of web applications. Masaryk University, Czech Republic; 2013.

4. Zalavadia S. When to Use Manual Testing vs. Automated Testing [Internet]. 2015. Available from: https://dzone.com/articles/when-use-manual-testing-vs (Accessed 06 Jun 2021)

5. Hoback F, Jouda A. Automated, deterministic testing versus stochastic testing using Quickcheck. Master's Thesis. KTH Royal Institute of Technology, Stockholm, Sweden; 2007.

6. Hayes LG. The Automated Testing Handbook. Software Testing Institute; 2004. 182 p.

7. Exforsys. Automated Testing Advantages, Disadvantages and Guidelines. 2005. [Internet] http://www.exforsys.com/tutorials/testing/automated-testing-advantages-disadvantages-and-guidelines.html (Accessed 06 Jun 2021)

8. Maurya VN, Kumar R. Analytical Study on Manual vs. Automated Testing Using with Simplistic Cost Model. ViXra 2012; 2(1):23–35.

9. SmartBear Software. Your Guide to Keyword-Driven Testing, A White paper. Test Complete. 2014. [Internet] https://smartbear.com/resources/white-papers/your-guide-to-keyword-driven-testing/ (Accessed 06 Jun 2021)

10. SmartBear Software. Uniting Your Automated and Manual Test Efforts, A White Paper. Test Complete. 2010. [Internet] https://static1.smartbear.co/support/media/resources/sp/uniting-automated-and-manual-tests.pdf (Accessed 06 Jun 2021)

11. Xie T, Tillmann N, de Halleux J, Schulte W. Future of developer testing. In: Proceedings of the FSE/SDP workshop on Future of software engineering research - FoSER '10. New York, New York, USA: ACM Press; 2010. p. 415.

12. Xie T, de Halleux J, Tillmann N, Schulte W. Teaching and training developer-testing techniques and tool support. In: Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion - SPLASH '10. New York, New York, USA: ACM Press; 2010. p. 175.

13. Mullarkey MT, Hevner AR. An elaborated action design research process model. Ågerfalk P, editor. European Journal of Information Systems. 2019 Jan 2; 28(1):6–20. Available from: https://doi.org/10.1080/0960085X.2018.1451811

14. Gurevitch J, Koricheva J, Nakagawa S, Stewart G. Meta-analysis and the science of research synthesis. Nature. 2018 Mar 8;555(7695):175–82. DOI:10.1038/nature25753

15. Holmes A, Kellogg M. Automating Functional Tests Using Selenium. In: AGILE 2006 (AGILE'06). IEEE; 2006. p. 270–5. DOI: 10.1109/AGILE.2006.19

16. Kaur H, Gupta G. Comparative Study of Automated Testing Tools: Selenium, Quick Test Professional and Testcomplete. International Journal of Engineering Research and Applications. 2013; 3(5):1739–43.

17. Hanna M, El-Haggar N, Sami M. A Review of Scripting Techniques Used in Automated Software Testing. International Journal of Advanced Computer Science and Applications. 2014; 5(1):194–202.

18. W3School. http://www.w3schools.com/ , (Accessed 06 Jun 2021)

19. Zambelich K. Totally Data-Driven Automated Testing, A white paper. 1998; 17. Available from: http://www.oio.de/public/softwaretest/Totally-Data-Driven-Automated-Testing.pdf