MANTECH Publications

Journal of Software Engineering & Software Testing
Volume 6 Issue 3
ISSN: 2457-0516 (Online)

Software Testing Automation Process Optimization for Web-based Applications

Premasiri S.M.U.

Lecturer

Department of Information Technology

University of Moratuwa, Sri Lanka

Corresponding Author's E-mail: *upremasiri@uom.lk*

Abstract

In modern era enormous number of software systems are built as web-based applications which run on a web browser. This trend has been emerged to cater the high demand coming from massive population around the globe. Agile and extreme programming like software development methodologies have been introduced for the development of application within a short period of time to fulfil the rapidly changing customer requirements. As a result, software systems have been developed within short development cycles. In agile development methodology a working piece of software is being developed within an iteration which is about two weeks' time. The time remaining for testing phase is very limited in agile development methodology and required level of testing coverage cannot be fulfilled through the manual testing approach Automation testing is identified as the solution for this problem. But testers have experienced some limitations with the automation testing approach also. Faults injected in the initial phase of the test automation process will result in huge system failures and also will badly affect to the reputation of the organization which is responsible for that particular product or service. Furthermore, the initial phase of the test automation life cycle is very much time consuming and resource consuming. This will result in inability to meet the deadlines. Therefore, it has been identified that optimization of the initial phase of the software test automation process helps to optimize the entire software test automation process.

This research study is mainly focused on the optimization of the three identified sub process in the initial phase of the software test automation life cycle. Study is continued to find methodologies to optimize the test automation tool selection process, test automation framework selection process and process of selection and prioritization of the test cases for automation.

Keywords: *Automation testing tools, Automation testing frameworks, Selection and prioritization of test cases*

INTRODUCTION

The study will consider the Software testing automation process optimization for web-based applications.

Software testing stage is one of the major phases in Software Development Life Cycle (SDLC) and one of the important aspects of Software Engineering. It is an activity to evaluate the actual results and expected results and to ensure that the software system which was developed is defect free. Software testing also helps to identify errors, gaps or missing requirements in contrary to the actual requirements.

In the present-day scenario there is a huge necessity for accelerated software development. Therefore, it is very important for software test professionals to look for a more effective and efficient way to execute their testing activities. One solution for improving the effectiveness of software testing has been applying automation to parts of the testing work.

But most of the occasions test automation process is much time consuming than the manual testing process and it may generate additional overhead to the project timeline and the budget. The investment cost for test automation can be significant. It is difficult to propose a generalizable cost model, due to a lack of published information available on the cost of ownership for test automation. However, the size and complexity of a test system can be in the order of, or often larger than, the complexity and size of the tested product (Wiklund et al., 2017). Even if a lot of resources and budget may spend on test automation procedures, it is still there a possibility to automation process to be not perform as expected. If the test automation does not perform well as expected, this investment is unlikely to be

recovered as planned, which may have significant consequences for the business.

Several optimization approaches have been introduced to optimize the software testing automation process, in order to minimize the cost and to provide an efficient and reliable output to the customer, while meeting the special requirements and constraints prevailing in the industry.

OBJECTIVES AND SCOPE OF THE STUDY

This research study focuses on the Automation testing process optimization. It addresses the Automated functional and regression testing process optimization for web-based applications. Furthermore, this research study focuses on optimization of software testing process activities i.e. selecting and prioritizing of test cases for automation, selecting best tool for test automation, Selecting best framework for test automation.

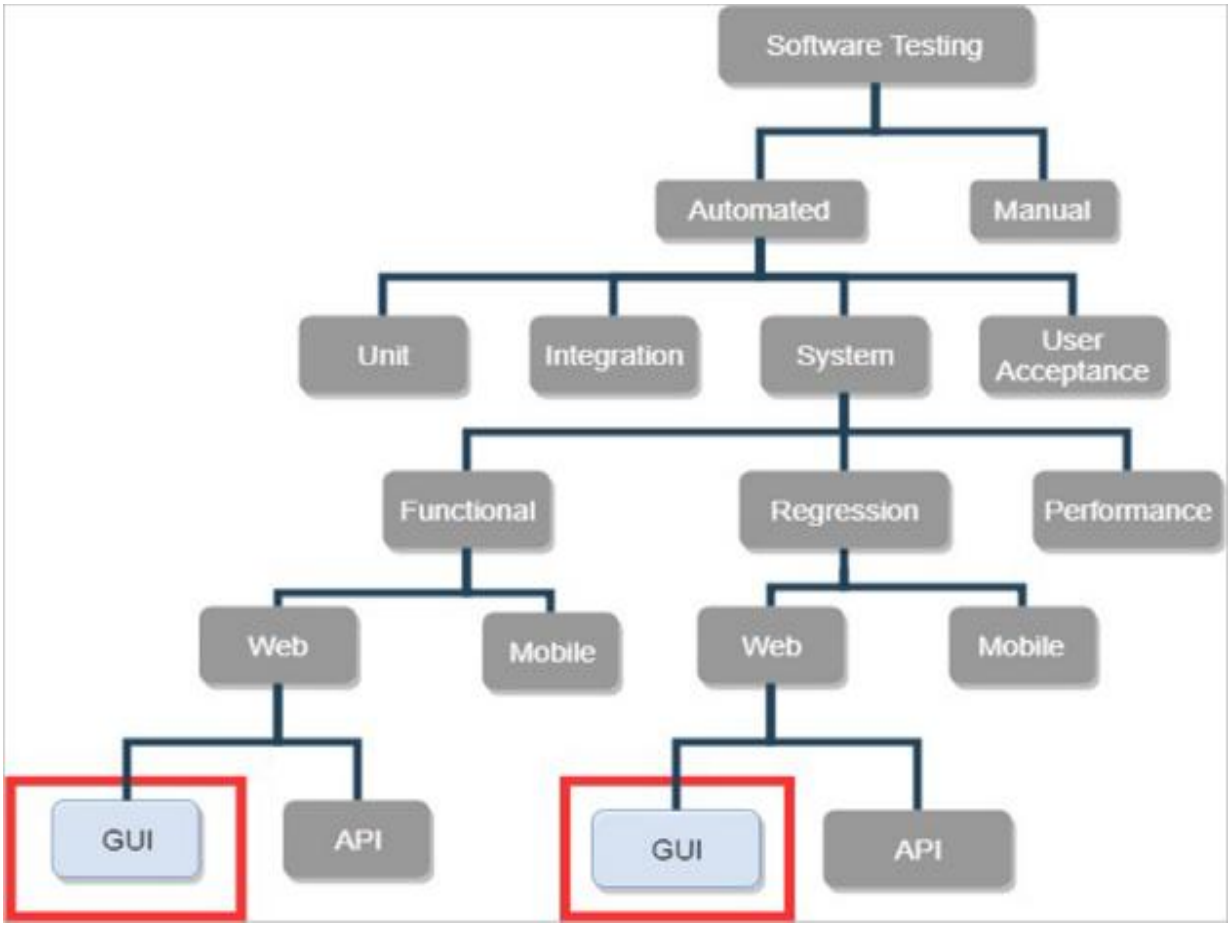

Figure 1- Scope of the Research (Software Testing Types)

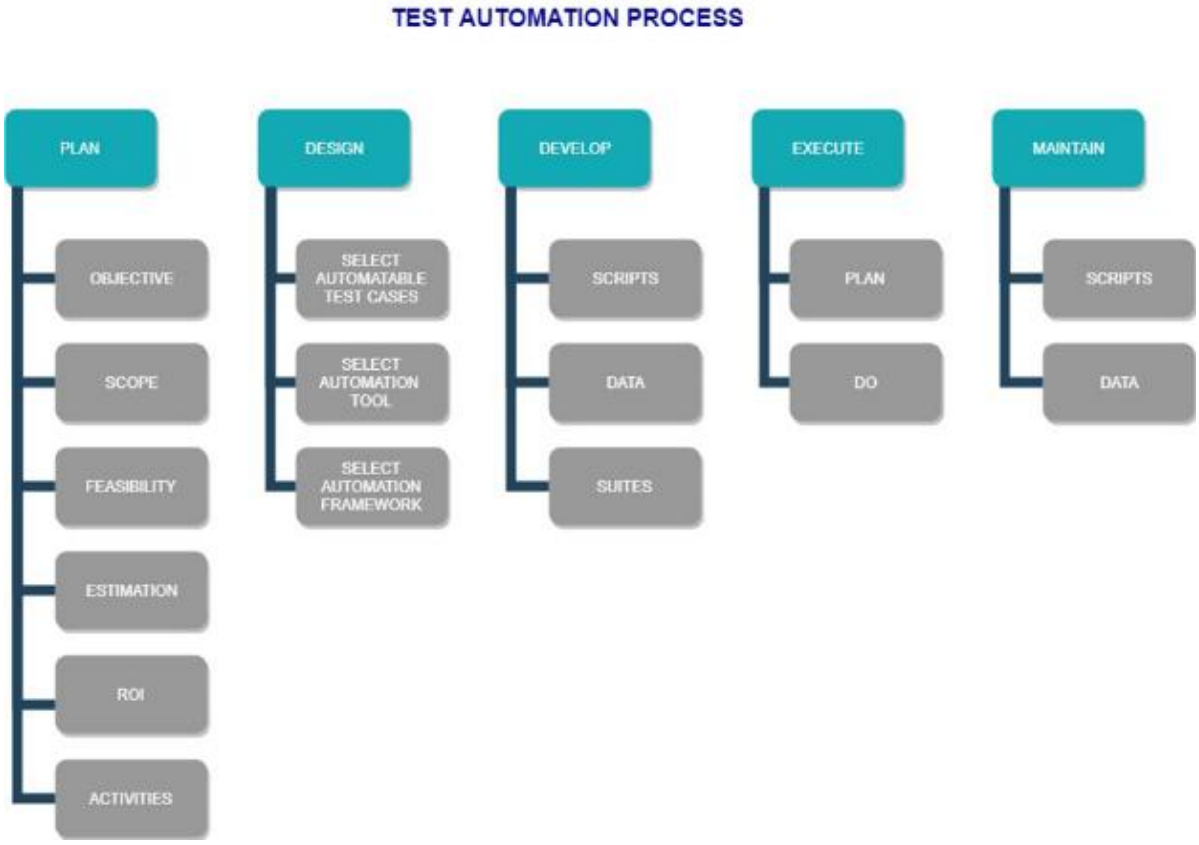

Figure 2 - Scope of the Research (Automation process)

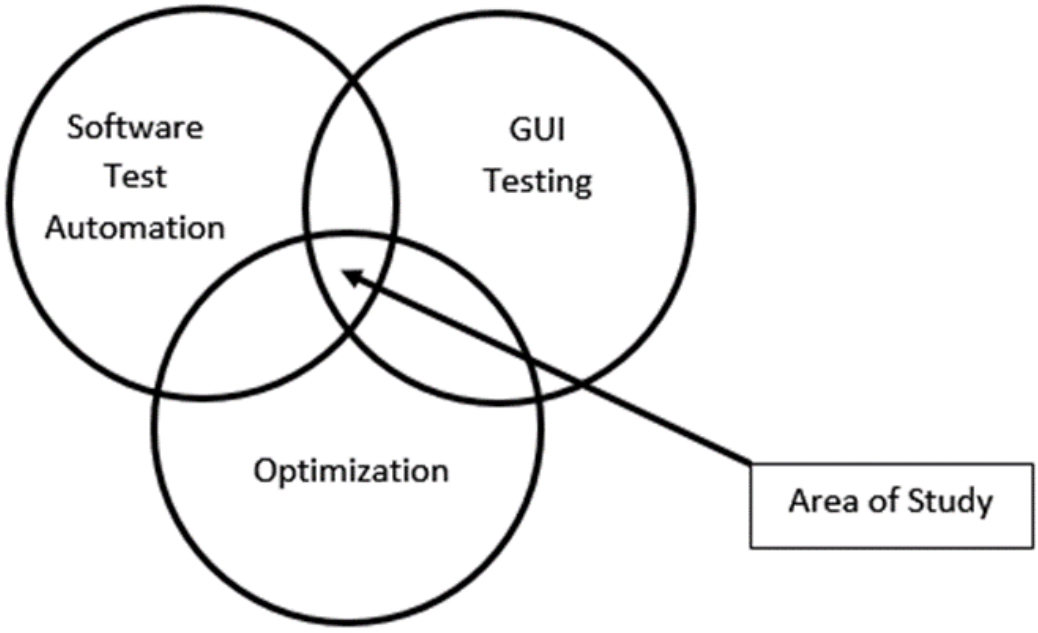

Figure 3 - : Scope of the Research (Areas)

RO1: Study current web based software test automation process

RO2: Identify problems related to web based software test automation process

RO3: Study literature and conduct an industrial survey to find optimal solutions for the identified problems

RO4: Develop a Meta model to define an optimized tool for web based test automation

RO5: Develop a checklist for finding a optimized framework for web based test automation

RO6: Develop an algorithm for optimizing the test case selection and prioritizing process

Background of the Research

In this rapid changing era of technology and highly competitive business environment, manual testing has made it difficult for organizations to analysis their web sites and applications. And time constraints, limited budget and limited human resources made the software testing a very big challenge for the software testers and also for the management.

As a result of previously mentioned limitations modern software testing industry is moving for the automated approach of testing. But when analysing the literature and industrial reports it can be identified that the software test automation is not giving anticipated output. Many software projects have been become failures because of the software test automation approach.

Researchers have identified that main reason for the failures in automation testing approach is the large amount of time consumption at the beginning of the test automation. Initial set up of the automation testing environment is much time consuming. And the next main cause for the failures is the organization is unable to return the money which was spent for the test automation process and the organizations may become unable to exist in the industry because of huge losses.

It is recommended to analyse all those risk factors and also it is compulsory to make sure that the test cases that are automating has a ROI in return.

LITERATURE REVIEW

Testing has been identified as the most expensive task of a software project. (Kasurinen, Taipale and Smolander, 2010) phase took over 50% of the project resources especially human resources and the project time (E. Kit, 1995). Besides causing immediate costs, testing is also highly connected with the costs in the

maintenance phase such as costs related to poor quality, and malfunctioning programs and errors cause large additional expenses to software producers (E. Kit, 1995) (G. Tassey, 2002).

Researchers tend to find a most effective and efficient methods for web automation testing as web-based applications have become the new trend in modern software industry. The rapidly increasing nature of web applications and their massive economic significance in the society made the field of web application testing a field with a very high importance. (Sandler, Badgett and Myers, 2013).

A Web application is a system which typically consists of a database (the back-end) and Web pages (the front-end), with which users interact over a network using a browser (Li, Das and Dowe, 2014). A web page can be either static, in which case the content is fixed, or dynamic, where the contents may depend on user input. User input to a web application consists of both navigational requests and data provided often through forms, which eventually affect the state of the underlying code on the server. (Sam path et al., 2018).

Web applications are complex and speedily updating software systems. (Akhtar Khan, 2012), (Mansour and Houri, 2006). Their testing is both challenging as well as critical. It is challenging because only the traditional testing methods and tools are not capable to fulfill the needs in modern web-based applications, since they do not address their exclusive features. Examples of the new features of web applications are: extensive use of events, rich graphical user interface, and incorporation of server-side scripting. Testing web-based applications is critical because failure may be very costly. (Mansour and Houri, 2006).

Web applications generally tend to take faster and quicker release cycles which makes web testing very challenging. Other than that, there are many challenges that can be identified when considering web application testing, as web applications has many features when compared to standalone applications. (Löfberg and Molin, 2005). Some of those features can be identified as heterogeneous technology, heterogeneous execution environment, high user population, concurrent transactions, state changes, heterogeneity in operating systems, multitier architecture, faster maintenance rate (Lakshmi and Mallika, 2017). Because of all these

features web application testing becomes less efficient and more time consuming. All these features need to be considered when a quality engineer executes their test cases on web applications. Quality of the web application may be affected and application will be error prone if one of those features of web application is neglected when executing tests.

But executing tests by considering all those features of web applications will not be an easy task if the manual testing (Maurya and Rajender, 2012). Approach is taken as it is very time consuming to test all those features manually and also it will be error prone because of human errors and also it will cost more because the particular company has to invest huge amount on human resources as it will need more human resources to execute those tests manually. Furthermore, humans will feel tired and bored as they had to consider more aspects when doing web application testing (Sharma, 2014).

Web applications testing through the Graphical User Interface (GUI) is extremely important therefore it should be given the required amount of time and effort during software development. Brooks et al identified that large number of real faults have been detected by GUI testing and proved that a large portion of those faults are belong to the underlying business logic of the application rather than the GUI itself (Brooks, Robinson and Memon, 2009) (Yuan, Cohen and Memon, 2011). Additionally, Sommerville argues that Software testing should constitute 30 – 40% of total project time and effort in order to deliver software with acceptable quality (Sommerville, 2007).

Agile software development methodologies such as SCRUM, XP (Extreme programming) Crystal, FDD or DSDM to name a few have become highly using software development methodologies in the industry. (Al-Zain, Eleyan and Hassouneh, 2013) Applications and as a result of that software developers had to develop features of web applications within a very short period of time like within 2-4 weeks. Software quality engineers also need to plan the tests, create test cases and execute them within that limited period of time as the web application need to be released within that particular iteration. As a results of that software quality engineers may become stressed and they will try to finish their work on time and some areas of the application may remain untested because of the limited time constraints, lack of human resources and also the existing

human resources may not be efficient enough to complete huge targets as they are demotivated as a result of the massive work load. Ultimately the released software application may consist of bugs and product becomes a failure.

Automation testing [Vishwa Nath and Kumarr, 2012] can be introduced as the solution for those problems that were identified above. It is the best way to increase the effectiveness, efficiency and coverage of software testing. Automation testing is the process where manual intervention is very minimal to execute the test cases. According to R. M. Sharma there are many benefits of selecting the automated testing approach for the test execution [Sharma, 2014]. The main benefit that was mentioned about the automated testing is it is much faster than the manual time. As different automation tools are using instead of the manual effort the efficiency of the test execution has been increased drastically. This feature in the automated testing will be helpful to recommend the automation testing approach for web application testing as web application testing also needs very efficient testing within a very short period of time such as within an iteration.

The second benefit that R. M. Sharma has identified was that automation testing is cost effective in comparison to the manual testing approach. Manual testing is consuming large number of human resources and also big amount of time. This will be much costly when compared to automation testing as when considering automation testing approach there is very less human intervention is needed and the automation tool is doing the job on behalf of the software quality engineer. Automation tool can be used for replacing considerable number of human resources. And cost for automation tool will not be much when compared to the cost for maintaining and managing the number of human resources which was required for doing that particular testing using manual approach. Furthermore, Automation test approach is executing the same number of tests within a less amount of time when compared to the manual testing approach and if it is required quality engineer can set the automation tool to execute tests in overnight also. As a result, time consumption for testing will be reduced and costs also be reduced. (Sharma, 2014) , (Dudekula Mohammad Rafi et al., 2012)

According to R. M. Sharma, the other benefit that has identified was automation testing is more reliable than the manual

testing. When compared to the manual testing human intervention is very less in automation testing the possibility for human error is very less in automated testing. Human are tended to make errors because of certain circumstances like stress, lack of interest in the work engaged and etc. But automation testing is much tool centered approach and psychological issues are not emerged in that approach. Reliability is a very important aspect to ensure the quality of a web application because the user base is very high for web applications and it affects very much harmfully to the company which owns that web application both financially and also to the reputation of the company if the application failed to perform its required functionalities. (Sharma, 2014).

Another major benefit that has identified in automated testing approach is repeatability in test execution. In agile methodology of software development, the standard practice in the industry for web application development is to develop one main feature or implement a bug fix at each consecutive iteration. When a new feature is added it is needed to test that newly added feature for functional and non-functional aspects and also it is highly required to perform a regression testing (Zarrad, 2015) to test the existing features of the web application to make sure that the functional and non-functional aspects of the existing modules are not affected by the new implementation. For every release it is necessary to repeat same testing again as it a must to test the existing modules in addition to the newly developed modules.

Few years back mostly manual testing has been used for testing those functionalities and it has been identified that repeating the execution of same testing activities again and again reduce the quality engineers' interest in testing that application and it is resulting to slow down the effectiveness and efficiency of the quality engineer and the testing process. Untested areas have a huge chance to be containing bugs and ultimately the quality of the web application may be reduced. Automated testing can be presented as a better approach for scenarios where repetitive testing is needed. Automated testing can be introduced as a good approach for regression testing. (Sharma, 2014).

In 2012, Rafi et al. has presented a paper which analyzes benefits and limitations of automated software testing through a systematic literature review and a practitioner survey. According to the survey benefits of test automation were

related to test reusability, repeatability, test coverage and effort saved in test executions. High initial invests in automation setup, tool selection and training were identified as the limitations. (Dudekula Mohammad Rafi et al., 2012).

Web testing involves the various type of testing such as functional testing, regression testing, usability testing, interface testing, compatibility testing, performance testing, security testing, cookies testing., as well as testing of web services, etc. (Angmo and Sharma, 2014) (Chhabra and Singh, 2017). Functional testing process of web applications can also can be automated but the ROI [Return On Investment] of automation will be low as in many occasions the test suite can be used only for one time.

Regression testing is a type of software testing that is performed to make sure that the unchanged existing features of the application is working fine after adding the new functionalities or after doing a bug fix. Regression testing is costly, and represents a very important problem in the software development. Commonly used methods of regression testing are re-running previously completed test cases and then checking whether the program behavior has been changed or the previously fixed faults have re-emerged. This requires a lot of cost, time and human resources as the size and complexity of the software increase. (Kandil, Moussa and Badr, 2015). ROI of selecting the automation testing approach for regression testing is comparatively very high as it is required to perform the same test execution repeatedly. Although the initial costs for implementation of the automation test suite and the cost for preparing automation infrastructure is very high when considering short term benefits. But automation of regression tests has many long-term advantages both in financially and quality wise aspects of the software released.

Wide range of automated testing tools are available in the market with variety of features and capabilities to the extent that the user is getting confused when selecting the tool that is best suited for their testing purpose. (Binder, 2009) (Monier and El-mahdy, 2015).

There are two types of test automation tools available in the market as commercial and open source tools. Open source tools are free for users to use with open source code to be modified (Toth, 2006). Commercial tools take advantage in organizations and mentoring capabilities

providing the user with facilities needed to accomplish tasks with extra controlled features and low efforts.

i. According to Jain et al., there are three types of automation testing tools are broadly available in the market
ii. Functional Tools (like HP quickest, IBM RFT)
iii. Management Tools (Like HP Quality Canter)
iv. Performance Tools (Like HP Load Runner)

There are various types of tools which can be used for web application testing available in the market. Test tool selection largely depends on technology the application under test is built on. This is one of the main challenges to be tackled before automation. A tool must not be selected based on its popularity but it's fit to the automation requirements. (Kakaraparthy, 2017). When selecting the appropriate tool for testing, quality engineer must consider and compare the features of each tool and need to come to the decision that which tool to be best suited to test the application. It needs lots of considerations like whether the tool is suited for integration and also need to check whether the tool is fit to the estimated cost as well as with the expected performance of your application. (Angmo and Sharma, 2014).

Furthermore, According to Satheesh and Singh, they have identified few more aspects to take in to consideration when selecting a test automation tool for web application testing. First characteristic they have identified is tool should support cross browser testing (Satheesh and Singh, 2017). As web applications need to be available to access using any browser available in the market. Therefore, when executing tests for web applications quality engineer should perform their test executions in every browser which is available in the market [But because of the resource limitations most of the time tests are executed by selecting at least three browsers and versions which has high user base by analyzing the statistical reports on browser usage].

Availability of online support materials and tutorials is also considered as a key feature that should be taken into consideration when selecting a test automation tool (Satheesh and Singh, 2017) Here the main aspect expecting is availability of the support from the developers of the tool and also the availability of forums and having user and community support for the tool. The

customer support for any tool is important especially when doing large scale projects. If an error with the tool cannot be solved quickly then it will affect the development time of the project. And also, will affect to the quality of the final outcome.

Coding experience is also a key aspect that needs to be taken into account when selecting an automation tool. (Satheesh and Singh, 2017). In here experience required by the user in terms of coding is specified. When selecting a tool, it is essential to decide that the coding experience of the users who are working in the project is satisfying the degree of coding experience that is required to operate that tool.

Another aspect that is needed to be considered is the easiness to set up and configure the tool. (Satheesh and Singh, 2017) Some automation tools are very difficult to set up and configure as it needs to implement lot of third-party frameworks and also need to have a sound knowledge and experience in programming. Therefore, when selecting an automation tool, it is better to choose a tool which is easy to set up and configure unless the users of the tool has sound knowledge in programming Reporting functionality is a very essential feature that should be taken in to consideration when selecting an automation tool. (Satheesh and Singh, 2017). The reporting functionality should be mandatorily available with the tools or tool should support to integrate third party tools for generating reports.

In 2013 Rattan and Shallu presented a research on Performance Evaluation & Comparison of Software Testing Tools. Authors have compared market leading vendor tool in functional test automation, HP (QTP) Quick test professional with popular & free Selenium. Test cases are executed using these tools and results are evaluated on the basis of discussion and conclusion. Authors have used evaluation time complexity and execution speed as the parameters for analyzing the results. (Rattan and Shallu, 2013).

In 2017, Chhbra and Singh has presented a comparative study and evaluation of the main features in web-based automation testing tools. According to authors, all most all the tools currently available in market for web-based automation testing has been used for this study. Selenium, Quick Test Proffesional(QTP), Load Runner, Test Complete, Watir, Tellurium, NeoLoad, WindMill, Ranorex, SilkTest, WinRunner are the tools which were used in this study. In addition to the tools used

for functional testing performance testing tools also taken in to evaluation for this case study. Authors have compared and evaluated these tools under several parameters i.e. ease of installation of tools, ease of learning, performance, cost of tool, test reporting etc. For the comparative study parameters such as Operating system supporting, License, Function of Type of Tool, Browser supporting, etc. Objective of this research is to help the quality engineer to choose a best tool for the task and to help the quality engineer to easily compare the various features of tools and also to find the best tool that fulfils his requirement. [Chhabra and Singh, 2017].

In 2012, Nagarani et al. has presented a research which was based on introducing a new approach to automate the software testing process to minimize the time wastage and to provide better utilization of resources for testing. Authors have suggested a method based on Coded UI (User Interface) tool for this purpose. By analyzing the results taken from this survey it has been proved mathematically that the testing time has been reduced drastically when the automation approach is used. For this research work researchers have used about 120 test cases and they have identified that it has taken 40 hours to complete the execution when the testing is done manually. But when automation testing is performed on the scripts in an un-attended mode of the personnel the time is reduced to 2 hours of execution time. With these results they have concluded that the proposed Coded UI tool and the generic framework created for the tool helps in automating the software applications under test with complete test coverage and also helps to save the huge amount of time and resources. Furthermore, they have concluded that test automation using the tool helps in maintenance of the scripts in an understandable manner which can be reused any number of times depending on the requirement of the user. (Nagarani and Venkata Ramana Chary, 2012).

In 2014, Jain et al. has presented a research which is based on the comparison of two automation testing tools namely QTP and Ranorex and the impact of those tools to the industry. In this paper, it provides a comparison of manual testing and automation testing for an internet banking website by considering several specific parameters i.e. cost, time and ROI. According to authors, there are two main objectives of this paper: To identify the advantages of automated testing tools and their impact on software testing, and to analyze automation tools available in the

market to identify, which tool create test ROI for the customer. Comparison of two automation tools, QTP and Ranorex has been done by considering several parameters such as, cost, environment, browsers supported, Online support, Coding languages supported, ease of learning, training cost, etc. and points have been given for each tool by comparing those tools under those parameters. After that authors have analyzed and graded those tools as edges and weaknesses by considering project timeline and Project cost and came to the conclusion that The time required for training resources on QTP is much lower than training them in Ranorex and Project cost using Ranorex is drastically lower than by using QTP. (Jain, Jain and Dhankar, 2014).

In 2014, Dubey and Shiwani has presented a comparative study on two test automations tools in the market namely Ranorex and Test Complete. Authors have conducted this comparison based on several selected features of these two tools i.e. the hard work involved with generating test scripts, capacity to playback the scripts, end result reports, and expenditure. According to authors, the main objective of this study was to investigate the features and concepts supported by these two functional testing tools in order to access unconventionally what pros and cons of the tools and to suggest guidelines for its additional expansion. Results have been compared under below mentioned capabilities such as Recording efficiency, Data driven testing, Test results reports, playback capability, easy to learn, cost and came in to conclusions under each capability. According to that when considering recording efficiency, TestComplete has been rated as extremely good and Ranorex is rated as average. For test results reports Ranorex is rated as extremely good and TestComplete is rated as average etc. As the overall conclusion authors have reported that TestComplete may be exact for assured definite situation but Ranorex can be the superior choice in many more situations. (Dubey and Shiwani, 2014).

In 2017, Kakaraparthy has presented a study based on the comparison between Ranorex, Test Complete and Selenium automation tools available in market and their use in the software project scenario. As the first section of the research author has analyze the automation tools and conclude that automated testing is suitable in most cases compared to manual testing. As the second part of the research, the above selected automation tools were assessed under certain parameters such as

execution speed, playback capability application support, cost of tool and additional training costs, etc. Results of the study have depicted that Ranorex is more suitable for web-based applications as it has many tools in built to the software package and also it does not require to learn the scripting language. Selenium can reduce the cost and limit the budget as it is open source and also Selenium tool supports cross platform and cross browser testing. When considering Test Complete, it has easy to learn UI and efficient playback. Test Complete is suitable for applications with lesser security needs. (Kakaraparthy, 2017).

In 2012, Rafi et al. has presented a paper which analyzes benefits and limitations of automated software testing through a systematic literature review and a practitioner survey. 45% of the respondents who participated for the survey agreed that available tools in the market offer a poor fit for their needs. Authors have suggested several characteristics that should be available in the automation tool by analyzing the feedback from the respondents i.e. should have an easy learning curve., should utilize test cases that are highly maintainable and robust, allow an efficient creation of test cases, should be easily configured and fitted to the various software development environments and ways of working, should support an incremental delivery of test automation. (Dudekula Mohammad Rafi et al., 2012).

In 2013, Leotta et al. has presented an industrial case study about test automation and test suite maintenance in the context of Web applications. The specific type of web applications he has used for this survey is Learning Content Management Systems (LCMS). In this research work authors have proposed a realignment approach and analyze the costs associated with equivalent selenium web driver test suites which were implemented using the page object pattern and also using other techniques that are used to locate elements in web pages. As the result of this study authors have revealed that ID-based test suites required significantly less maintenance effort than the XPath-based ones. The main objective of this author work is to identify which Selenium web driver method is best suited for locating elements in a web page that reduces the maintenance efforts when realigning the test cases to a new release. (Leotta et al., 2013).

In 2013, Leotta et al. has also presented an industrial case study on investigating the

potential benefits of adopting the page object pattern to improve the maintainability of Selenium Web Driver test cases. In this research work authors have compared two equivalent test suites, one of them was built using page object pattern and other one was built without using page object pattern. As the result of this study authors have identified that time required to repair the test suite and the number of lines of code need to modify when repairing test suites, has shown a strong decline when the page object pattern is used. The main objective of this author work is to identify whether there is a positive effect to the maintainability of Selenium Web Driver test cases when using page object model for building test suites. (Leotta et al., 2013).

In 2012, Wang et al. has presented a new automation framework integrated by selenium and Jmeter that is supported for cross platform and cross browser testing. Test steps and test data were shared by this automation framework, which is easy to switching in a range of testing for web application. One can proficiently recover the extensibility and reusability of automation test by using this software framework. (Fei Wang and Wencai Du, 2012).

A test case is a set of tests performed in a sequence and related to a test objective (Fewster and Graham, 1994) and a test suite is a set of test cases that will execute sequentially. There are two major accepted methods to execute regression tests. The first one is by re-executing all test cases in order to test the entire system once again. But there may not be sufficient resources to allow the re-execution of all test cases every time when a modification is done on the application. The second way to perform regression test is to prioritize the test cases considering their beneficial factors such as coverage, and re-execute the test cases according to that ordering. (Maia et al., 2010).

Yoo and Harman have identified four major approaches for regression testing (Yoo and Harman, 2010)

- **Retest All**
  All test cases in the test suite have been executed.
- **Regression Test Selection**
  Chosen test cases are executed to verify and test modifications occur in program.
- **Test Suite Reduction**
  Identify relevant or superfluous test cases and exclude those test cases.
- **Test Case Prioritization**
  Choose the ideal set of test cases

In 2009, Maia et al. has presented a paper on higher-level procedure to prioritize test cases using Reactive GRASP. Authors have compared Reactive GRASP algorithm with other well-known search-based algorithms such as Greedy algorithm, Additional Greedy algorithm, Genetic algorithm, and Simulated Annealing. Coverage and time performances are the two main parameters that have been taken in to consideration for the above-mentioned comparison. By analyzing the results of this study these are the conclusions that could be made, In terms of coverage performance, Reactive GRASP algorithm performed significantly better than the genetic algorithm and Reactive GRASP algorithm based approach has more stable behaviour and also consumes less time when compared to all other solutions. (Maia et al., 2009).

In 2015, Cui and Wang has presented a cost-benefit evaluation model for automated testing which is based on test case prioritization. This study has introduced a new cost-benefit model for automated testing. In this model, to determine the prioritization of automated test cases, K-means algorithm has been proposed and to select the proportion of automated test cases, the principle of production possibilities frontier has been introduced. [Wang and Cui, 2015].

In 2017, Sultan et al. has presented a study on test cases prioritization Techniques based on an analytical review. Authors have compared seven major test case prioritization techniques which are commonly in use i.e Fault Severity, Fault Localization, Mutation faults, Ordered Sequence of Program Elements, APFD, Model Checker, Search Algorithm and analyzed their pros and cons. Furthermore, they have analyzed the prioritization techniques available in the literature and factor based comprehensive table is developed (Sultan et al., 2017).

In 2017, Sagar and Prasad presented a paper based on a survey on test case prioritization techniques for regression testing. In this research study authors have investigated several test case prioritization techniques which are classified based on various parameters such as Average Percentage Faults Detected (APFD), effect size and statistical testing. Many existing methods are surveyed in this work and the findings suggest that model-based test case prioritization is best in all aspects. This survey investigates several test case prioritization techniques and provides the

idea for efficient methods for future work. (Sagar and Prasad, 2017).

METHODOLOGY

This study was conducted in order to address the knowledge gap in optimization of software test automation process. An Industrial survey was conducted as the first phase of this study for the purpose of finding the highly using test automation tools and frameworks in the Software test automation industry.

Three major test automation tools were selected based on the survey

i. Selenium web driver
ii. Test Complete
iii. HP QTP

Feature based analysis was undertaken for the identified tools by considering functional and non-functional testing for a set of pre-defined scenarios. Thereby a meta model was developed for the purpose of defining an optimized test automation tool and the framework for software test automation. A mathematical-model based approach was taken for optimizing the test case selection and prioritization which is identified as a NP-hard problem.

RESEARCH DESIGN

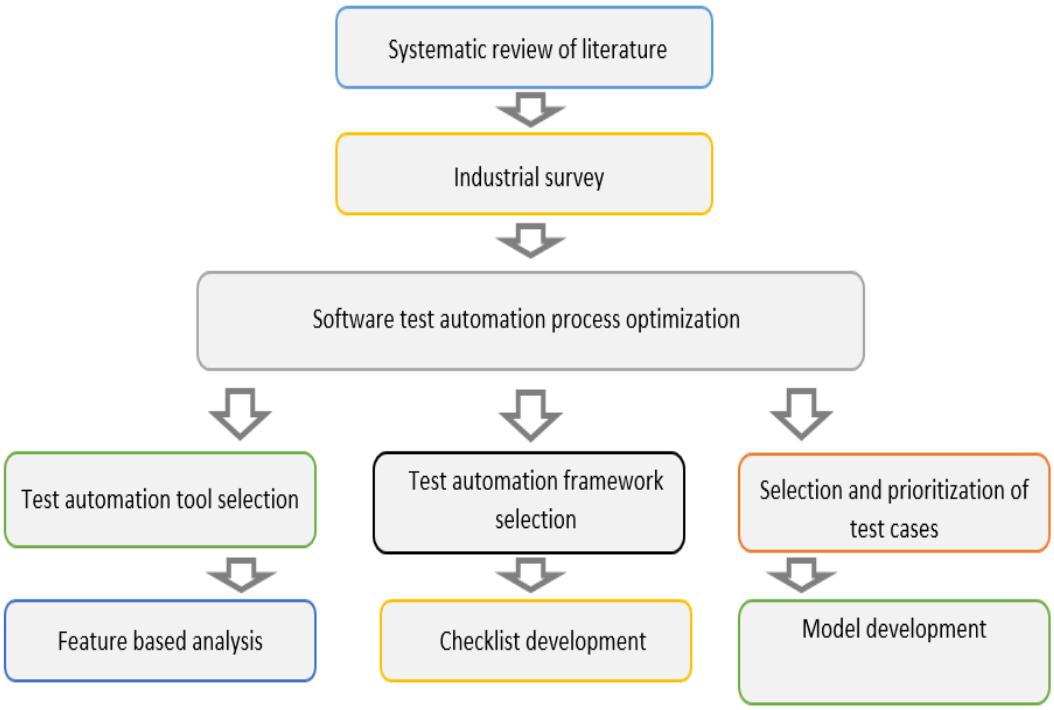

Figure 4 - Research design

The commencement is via literature review of the research area, that is conducted to identify the studies that have been already done related to Software test automation. In this phase, current studies are examined to identify the fundamental characteristics in web-based software test automation. The literature survey is conducted based on the four major areas of concern i.e. Web based software test automation, test automation tools, test automation frameworks, test case prioritization techniques Then the results of the literature survey are related to the business context and research questions will be formulated. Furthermore, the case studies are identified in order to gather data for the study. Through this Systematic Review of Literature (SRL), Software test automation process related issues solving approaches were analyzed. The gaps are identified systematically.

There is a wide range of software test tools and frameworks made available in the market by popular vendors. But some of them do not have same demand as other tools. An industry wide survey was conducted by considering 10 software companies in Sri Lanka to identify most demanded software testing frameworks and tools used in Sri Lankan Software testing industry. By considering the results taken from that survey this study is narrowed down to three major tools and frameworks. The selected tools are Selenium web driver, Test Complete and HP QTP.

Software test automation process has been optimized by minimizing the time consumption, resource consumption and by increasing the efficiency and effectiveness of the tools and the staff. The overall process optimization has been achieved through optimizing the identified sub processes in the process. This study has a three-fold approach.

In automation testing process, automation tool selection is identified as a major phase where a lot of time has to be allocated to get the correct decision. Through this study feature based analysis is conducted to compare and contrast the features in each of the three selected tools and to provide detailed guide for the tester to take in to consideration when choosing a test automation tool for their testing activities. Following list of features were examined under this sub section of this study.

- Recording efficiency
- Capability of generation of scripts
- Data driven testing
- Test result reports
- Playback of the scripts

- Easy to learn
- Cost

Automation framework selection is also identified as one of the major phases that is needed to be optimized in order to optimize the overall process of web-based automation testing. By analyzing the literature and the feedback from the testers in the industry, a checklist has been introduced which is recommended to refer before selecting the framework for the software automation testing.

Test case selection and prioritization is paramount important phase in the software test automation process as it is hard to allocate resources and time for automating each and every test case within the fixed cost and time constraints. Test selection and prioritization is needed to be conducted very carefully and need to make sure that any of the important test cases never be missed. Test case selection and prioritization is considered as a NP-hard problem.

Test case selection and prioritization optimization approach can be sub divided in to two major categories for the ease of the understanding.

i. Selection of test cases for automation

A key word (tag) is assigned to each test case in the initial stage of the test case preparation. The keyword should be a unique text which can be used to identify the release or the functionality where that particular test case is referring to. All the test cases are placed in a well-managed test case repository. Test management tools i.e. HP ALM, Test Rail can be used for this purpose. Test Rail is used for maintaining the test case repository for this study.

When retrieving the test cases for test automation user can search the test case repository by the previously assigned key word and all the test cases related to the particular release or functionality can be retrieved.

ii. Prioritization of selected test case

Test prioritization technique was selected by analyzing the literature related to test case prioritization problem. At the initial stage, four major types of test case prioritization techniques were identified.

i. Coverage-based techniques
ii. Cost Effective-based techniques
iii. Customer Requirement-based techniques

iv. Chronographic history-based techniques

For the purpose of selecting the most suitable technique for this study an analysis based on pros and cons of each technique was conducted and a protype was designed for the purpose of optimizing the test case selection and prioritization. **See Figure: 5 below**

Data Acquisition and Validation

Data required for test automation tools selection is gathered through an industry based survey and the feature-based analysis was done through the data gathered from the analysis done by studying the literature.

The checklist for selecting the framework is gathered through literature review and Meta model was created by using the analysed data.

Test case prioritization techniques were validated through the data taken from the literature survey and the protype was validated using test cases and through a real user feedback scenario.

IMPLEMENTATION

As the first phase of the study web-based test automation tools were identified through a literature survey and then the list of identified tools was narrowed down in to three specific tools based on the survey conducted considering the industrial utilization of the automation tools.

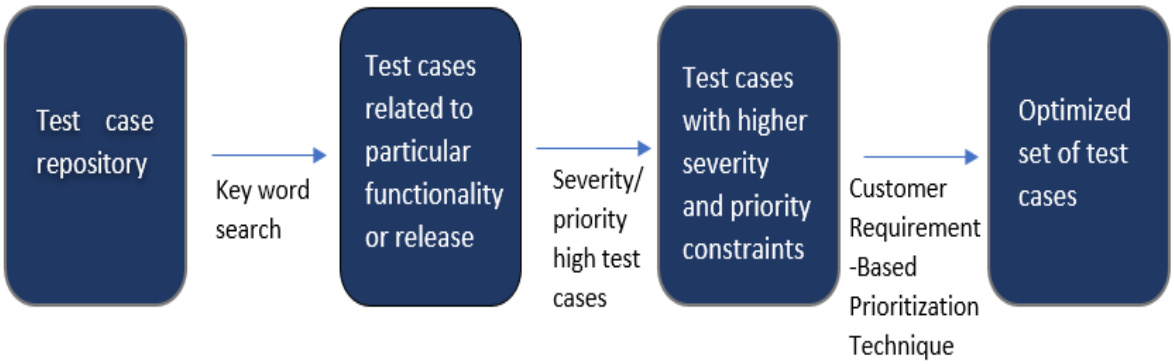

Figure 5 - High level diagram for test case selection and prioritization

List of identified tools through the literature survey are depicted in table 1

Table 1 - Tool Names with Supporting Testing Types

	Tool name	Type of testing supported
1.	WATIR	Functional Testing
2.	Selenium	Functional Testing
3.	HP-QTP	Functional Testing
4.	Fitnesse	Acceptance Testing
5.	TestComplete	Functional Testing, Unit Testing
6.	Load Runner	Load Testing
7.	TestNG	Integration Testing, Functional Testing, End-End Testing, Unit Testing
8.	TOSCA	Functional Testing
9.	SilkTest	Functional Testing
10.	WinRunner	Functional Testing
11.	ApacheJMeter	Performance Testing, Load Testing
12.	NeoLoad	Load Testing
13.	LoadUI	Load Testing
14.	WebLoad	Load Testing
15.	WAPT	Load Testing, Stress Testing
16.	Rational Performance Tester Testing Anywhere	Performance Testing Functional Testing
17.	Qengine	Functional Testing
18.	MUTANDIS	Functional Testing
19.	ATUSA	Functional Testing
20.	Crawljax	Navigation Testing
21.	JSART	Regression Testing
22.	webMate	Regression Layout Testing
23.	reAjax	Functional Testing
24.	WebVizor	Functional Testing
25.	Web Portal In Container Testing	Integration Testing
26.	Veriweb Tool	Navigation Testing
27.	WebScarab	Security Testing
28.	Acunetix	Security Testing, Penetration Testing
29.	Fortify	Security Testing

Tools that were identified as the highly using web-based testing tools through the industry-based survey are depicted in Table 2

Table 2 - Tools Selected Through Industry-Based Survey

	Tool name	Type of testing supported
1.	Selenium Web driver	Functional Testing

2.	TestComplete	Functional Testing, Unit Testing
3.	HP-QTP	Functional Testing

Selected tools were analysed against following performance parameters

Table 3 - Parameter List for Tool Analysis

	Parameter	Description
1	Programming Language Supported	Supported programming languages for writing test cases
2	Test Result Report	How the test result is represented
3	Record & Playback	Details about how easy it is to record & playback a test case or test suite
4	Platform Compatibility	Various platform support to perform testing
5	Browser Compatibility	Various browsers support to perform testing

For the purpose of rating performance parameter against each tool following 5 – point criteria is used

Table 4 - Rating criteria

Point value	Performance Rating
1	Extremely bad
2	Fairly bad
3	Average
4	Fairly good
5	Extremely good

As the first step set of test cases were written to automate the functional testing of the Google home page www.google.com. Eclipse, IntelliJ idea or any other IDE can be used for this purpose and Eclipse is used as the development environment in this study.

Then need to select a platform and browser for the testing, for this study cross platform testing analysis was done using

literature and for cross browser testing Chrome and Firefox browsers are used.

Second step was to execute the test cases that were written in first step using previously identified web-based test automation tools in order to perform the performance analysis.

Last step is to perform tool evaluation based on specified parameters for rating the tools based on their performance.

EVALUATION OF RESULTS

Set of guidelines to consider when selecting the best framework for web-based test automation is developed under this section. Selecting a best framework that will fulfil high number of tester's requirements is identified as a fact that improves the efficiency of software automation process. This guideline is developed by analysing the frameworks used in the industry and by studying the literature. It is not possible to recommend a framework or rate frameworks by analysing COTS frameworks available in the market as most of the companies use their own custom build frameworks for automation testing activities. By getting this concern in to consideration a set of guidelines is developed under this study to use as a checklist when designing their own frameworks. Or if someone needs to select any COTS framework this checklist is helpful to choose the best out of available frameworks.

Questions that should be answered when designing a custom build framework

Q1: What percent of application will be automated using the framework?
Q2: How stable is the application?
Q3: Should the Automation Framework be tightly or loosely coupled with the application under-test (AUT)?
Q4: Who is the target audience for using Test Automation Framework?
Q5: What are testing skills of the target audience?
Q6: How much is the enterprise willing to invest in automation?
Q7: What is the expected ROI from the automation efforts?

Features that should be available in a optimized framework (Framework selection checklist)

Feature 1: Availability of trainings, tutorials and guidelines
Feature 2: Capability to generate test cases easily

Feature 3: Ability to optimize test execution

Feature 4: Low Licensing and support costs

Feature 5: Low level of programming skills needed

Feature 6: Ability to generate good test reports

Feature 7: Availability of big user community

Feature 8: Availability of CI, DevOps support

Feature 9: Good UI/UX features

Feature 10: Availability of support services (troubleshooting, enhancements,..)

Feature 11: Ability to integrate with **ALM tools**

Feature 12: Low level of experience needed.

Test case selection and prioritization

Test case selection process is implemented based on a keyword search algorithm. As mentioned in Chapter 3 user can assign a tag for each test case at the test case writing stage. Several test cases may be assigned the same tag as the tag is assigned based on the release or functionality and it is possible to have more than one test case for each release/function. TestRail is used as the test management tool for maintaining the test case repository for this study.

User is allowed to enter the tag related to particular release/functionality which is planned to be automated through a Graphical User Interface (GUI) and test cases which are tagged with that entered name is processed with the use of key word search algorithm. And the output from this step is used as the input for the prioritization algorithm.

Four major test case prioritization techniques have been selected through literature. The most suitable prioritization technique for this study is identified by performing a comparative analysis considering the pros and cons of each selected technique. Thereby a protype for test case selection and prioritization is developed by using the selected technique. Initially selected test case prioritization techniques are depicted in below Table

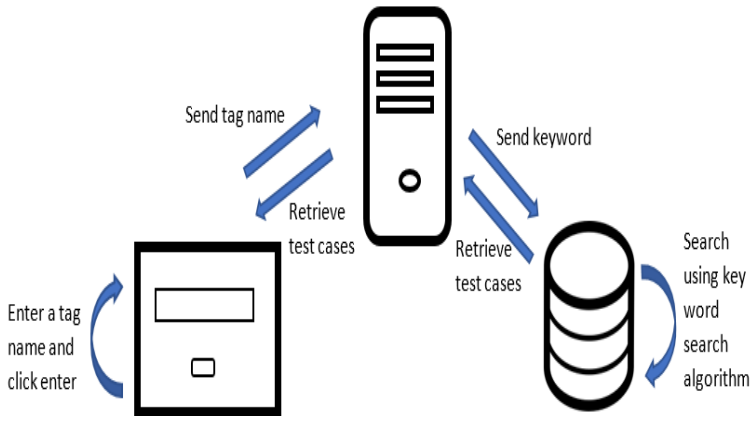

Figure 1 - High-Level Architectural Diagram for the Implementation of the Test Selection Process

Table 5 - Test case prioritization techniques

	Prioritization technique	Description
1	Coverage-based techniques	Coverage-based techniques are methods to prioritize test cases based on coverage criteria, such as code coverage, branch coverage and statement coverage
2	Customer Requirement-based techniques	Customer requirement-based techniques are methods to prioritize test cases based on requirement documents
3	Cost Effective-based techniques	Cost effective-based techniques are methods to prioritize test cases based on costs, such as cost of analysis and cost of prioritization
4	Chronographic history-based techniques	Chronographic history-based techniques are methods to prioritize test cases based on test execution history

Functional testing concerns which are identified through the literature are analysed against the test prioritization techniques using a matrix as depicted in Table

Table 6 - Functional testing constraints against test case prioritization techniques

Functional testing concerns	Contribution from test case prioritizing techniques for functional test case prioritization			
	CBT	CRBT	CEBT	CHBT
Requirement specification is the		X		

baseline				
Black box testing approach is used		X	X	X
Purpose is to find severe faults		X		X
Verify that all major functionalities are working fine	X	X		X

According to the derived matrix, customer requirements-based prioritization technique satisfies all the functional testing concerns thereby customer requirements-based prioritization technique is identified as the most suitable prioritizing technique for functional test case prioritization.

Brottier et al. presented the requirements-based test case prioritization approach to prioritize a set of test cases. They have identified several factors to weight (or rank) the test cases. Those factors are the customer-assigned priority (CP), requirements complexity (RC) and requirements volatility (RV).

Additionally, Value (1 to 10) has been assigned for each factor for the measurement. Higher factor values indicate a need for prioritization of test case related to that requirement. Weight prioritization (WP) measures the importance of testing a requirement earlier. (Brottier et al., 2006).

$WP = \Sigma (PF_{value} * PF_{weight})$; PF=1 to n

Where:

WP denotes weight prioritization that measures the importance of testing a requirement.

PF_{value} is the value of each factor, like CP, RC and RV.

PF_{weight} is the weight of each factor, like CP, RC and RV.

Test cases are then ordered such that the test cases for requirements with high WP are executed before others. Prototype has been developed by using this algorithm and the output which has been taken from the test case selection phase is considered as the input for test case prioritization phase when developing the prototype

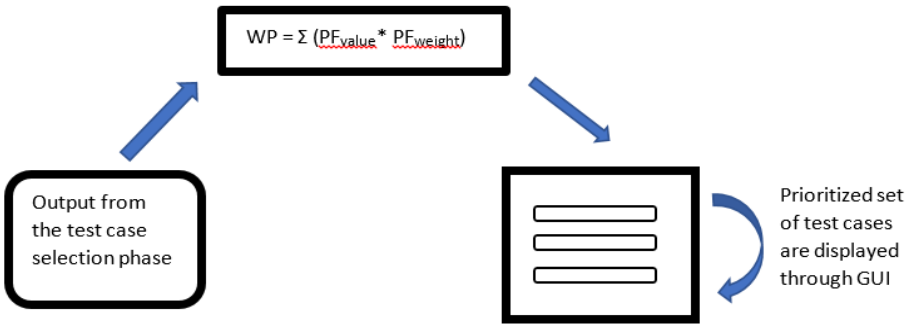

Figure 7 - High level architecture diagram of the test case prioritization process

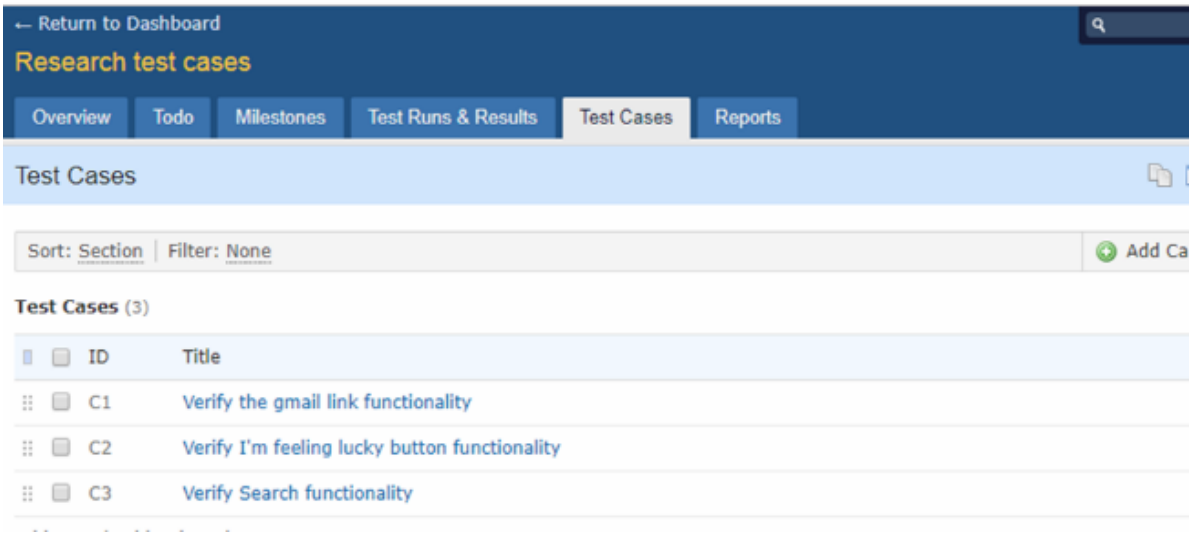

Figure 8 - TestRail test case repository

CONCLUSIONS

This study indeed contributes to literature in numerous ways, where a systematic review was done based on the optimization methods which are used to optimize the Web-based software automation testing. In this study, a three-fold approach is used for optimizing the entire process of web-based automation testing i.e. Optimization of the test automation toll selection process, Optimization of the test automation framework selection process, Optimization of the test case selection and prioritization process.

Since software testing is a major step and an unavoidable phase in software development life cycle, optimization approach for the testing phase is a high necessity. But the current software test automation approaches tend to become failures because of some inefficiencies in the automation process.

According to the literature survey and by analysing industrial reports it has been identified that most of the software automation approaches has been failed as a result of the inefficient decisions and actions taken phase at the initial phase of the test automation process. Initial set up phase of the test automation process has been identified as the most time consuming and resource consuming phase of the software test automation life cycle. This study is suggesting a method to optimize the entire test automation process by optimizing the initial set up phase of the software test automation life cycle

Feature based analysis of the web-based software test automation tools, a checklist to use as guideline for selecting the most optimized test automation framework for web-based test automation, prototype design of system for test case selection and optimization can be identified as the major deliverables of this study.

REFERENCES

1. Akhtar Khan, I. (2012). Quality Assurance and Integration Testing Aspects in Web Based Applications. International Journal of Computer Science, Engineering and Applications, 2(3), pp.109-116.
2. Al-Zain, S., Eleyan, D. and Hassouneh, Y. (2013). Comparing GUI Automation Testing Tools for Dynamic Web Applications. Asian Journal of Computer and Information Systems, 1(2).
3. Angmo, R. and Sharma, M. (2014). Performance evaluation of web based automation testing tools. 2014 5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence).
4. Binder, R. (2009). Testing object-oriented systems. Boston: Addison-Wesley.
5. Brooks, P., Robinson, B. and Memon, A. (2009). An Initial Characterization of Industrial Graphical User Interface Systems. 2009 International Conference on Software Testing Verification and Validation.
6. Chethan, M. and Padma, M. (2015). Test Automation Framework on Library Architecture. International Journal on Recent and Innovation Trends in Computing and Communication, 3(5).
7. Chhabra, S. and Singh, S. (2017). COMPARATIVE STUDY AND EVALUATION OF WEB BASED AUTOMATION TESTING TOOLS. International Journal of Computer Application, 7(2).

8. Dubey, N. and Shiwani, S. (2014). Studying and Comparing Automated Testing Tools; Ranorex and TestComplete. International Journal Of Engineering And Computer Science, 3(5).
9. Dudekula Mohammad Rafi, Katam Reddy Kiran Moses, Petersen, K. and Mantyla, M. (2012). Benefits and limitations of automated software testing: Systematic literature review and practitioner survey. 2012 7th International Workshop on Automation of Software Test (AST).
10. E. Kit, Software Testing in the RealWorld: Improving the Process, Addison-Wesley, Reading, Mass, USA, 1995
11. Fei Wang and Wencai Du (2012). A Test Automation Framework Based on WEB. 2012 IEEE/ACIS 11th International Conference on Computer and Information Science.
12. G. Tassey, "The economic impacts of inadequate infrastructure for software testing," RTI Project 7007.011, U.S. National Institute of Standards and Technology, Gaithersburg, Md, USA, 2002.
13. Jain, A., Jain, M. and Dhankar, S. (2014). A Comparison of RANOREX and QTP Automated Testing Tools and their impact on Software Testing. International Journal of Engineering, Management & Sciences, 1(1).
14. KAKARAPARTHY, D. (2017). OVERVIEW AND ANALYSIS OF AUTOMATED TESTING TOOLS: RANOREX, TEST COMPLETE, SELENIUM. International Research Journal of Engineering and Technology, 4(10).
15. Kandil, P., Moussa, S. and Badr, N. (2015). A Study for Regression Testing Techniques and Tools. International Journal of Soft Computing and Software Engineering, 5(4).
16. Kasurinen, J., Taipale, O. and Smolander, K. (2010). Software Test Automation in Practice: Empirical Observations. Advances in Software Engineering, 2010, pp.1-18.
17. Kit, E. and Finzi, S. (1999). Software testing in the real world. Harlow, England.: Addison-Wesley.
18. Lakshmi, D. and Mallika, S. (2017). A Review on Web Application Testing and its Current Research Directions. International Journal of Electrical and Computer Engineering (IJECE), 7(4), p.2132.

19. Leotta, M., Clerissi, D., Ricca, F. and Spadaro, C. (2013). Improving Test Suites Maintainability with the Page Object Pattern: An Industrial Case Study. 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops.
20. Leotta, M., Clerissi, D., Ricca, F. and Spadaro, C. (2013). Repairing Selenium Test Cases: An Industrial Case Study about Web Page Element Localization. 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation.
21. Li, Y., Das, P. and Dowe, D. (2014). Two decades of Web application testing—A survey of recent advances. Information Systems, 43, pp.20-54.
22. Löfberg, M. and Molin, P. (2005). Web vs. Standalone Application - A maintenance application for Business Intelligence. Master of Science in Software Engineering. School of Engineering at Blekinge Institute of Technology.
23. M. Fewster and D. Graham, Software Test Automation, Addison-Wesley, Reading, Mass, USA, 1st edition, 1994.,
24. Maia, C., do Carmo, R., de Freitas, F., de Campos, G. and de Souza, J. (2010). Automated Test Case Prioritization with Reactive GRASP. Advances in Software Engineering, 2010, pp.1-18.
25. Mansour, N. and Houri, M. (2006). Testing web applications. Information and Software Technology, 48(1), pp.31-42.
26. Maurya, V.N., & KUMAR, E.R. (2012). Analytical Study on Manual vs . Aumated Testing Using with Simplistic Cost Model.
27. Maurya, Vishwa Nath and Er. RAJENDER KUMAR. "Analytical Study on Manual vs . Automated Testing Using with Simplistic Cost Model." (2012).
28. Monier, M. and El-mahdy, M. (2015). Evaluation of automated web testing tools. International Journal of Computer Applications Technology and Research, 4(5), pp.405-408.
29. Nagarani, P. and VenkataRamanaChary, R. (2012). A tool based approach for automation of GUI applications. 2012 Third International Conference on Computing, Communication and Networking Technologies (ICCCNT'12).

30. Rattan, R. and Shallu (2013). Performance Evaluation & Comparison of Software Testing Tool. International Journal of Information and Computation Technology, 3(7).
31. Sampath, Sreedevi & Hill, Emily & Sprenkle, Sara & Pollock, Lori. (2018). Coverage criteria for testing web applications.
32. Sandler, C., Badgett, T. and Myers, G. (2013). The art of software testing. Hoboken, N.J.: Wiley.
33. Satheesh, A. and Singh, M. (2017). Comparative Study of Open Source Automated Web Testing Tools: Selenium and Sahi. Indian Journal of Science and Technology, 10(13), pp.1-9.
34. Sharma, R. (2014). Quantitative Analysis of Automation and Manual Testing. International Journal of Engineering and Innovative Technology (IJEIT), 4(1).
35. Sommerville, I. (2007). Software engineering. Harlow, England: Addison-Wesley.
36. Sultan, Z., Abbas, R., Nazir, S. and Asim, S. (2017). Analytical Review on Test Cases Prioritization Techniques: An Empirical Study. International Journal of Advanced Computer Science and Applications, 8(2).
37. Toth, K. (2006). Experiences with Open Source Software Engineering Tools. IEEE Software, 23(6), pp.44-52.
38. Wang, C. and Cui, M. (2015). Cost-Benefit Evaluation Model for Automated Testing Based on Test Case Prioritization. Journal of Software Engineering, 9(4), pp.808-817.
39. Yoo, S. and Harman, M. (2010). Regression testing minimization, selection and prioritization: a survey. Software Testing, Verification and Reliability, p.n/a-n/a.
40. Yuan, X., Cohen, M. and Memon, A. (2011). GUI Interaction Testing: Incorporating Event Context. IEEE Transactions on Software Engineering, 37(4), pp.559-574.
41. Zarrad, A. (2015). A Systematic Review on Regression Testing for Web-Based Applications. Journal of Software, 10(8), pp.971-990.
42. Nisha Gogna. (2014) ― Study of Browser Based Automated Test Tools WATIR and Selenium ‖ . International Journal of Information and Education Technology, Vol. 4, No. 4.

43. J. Križani, A. Grguri, M. Mošmondor, P. Lazarevski. (2010). Load testing and performance monitoring tools in use with AJAX based web applications ‖ , MIPRO 2010, May 24-28, 2010, Opatija, Croatia.
44. Valentin Dallmeier, Bernd Pohl, Martin Burger, Michael Mirold, Andreas Zeller. (2014) WebMate: Web Application Test Generation in the Real World, ICSTW '14 Proceedings of the 2014 IEEE International Conference on Software Testing, Verification, and Validation Workshops, Pages 413-418.
45. Wiklund, K., Eldh, S., Sundmark, D. and Lundqvist, K. (2017). Impediments for software test automation: A systematic literature review. Software Testing, Verification and Reliability, 27(8), p.e1639.
46. Pardha Sagar, G. and Prasad, P. (2017). A Survey on Test Case Prioritization Techniques for Regression Testing. Indian Journal of Science and Technology, 10(10), pp.1-6.
47. Brottier, E., Fleurey, F., Steel, J., Baudry, B. and Traon, Y. (2006). Metamodel-based Test Generation for Model Transformations: an Algorithm and a Tool. 2006 17th International Symposium on Software Reliability Engineering.
48. Dukes, L., Yuan, X. and Akowuah, F. (2013). A case study on web application security testing with tools and manual testing. 2013 Proceedings of IEEE Southeastcon.
49. Kaur, H. and Gupta, G. (2019). Comparative Study of Automated Testing Tools: Selenium, Quick Test Professional and Testcomplete. Journal of Engineering Research and Applications, 3(5), pp.pp.1739-1743.
50. Ngo, D. (2019). [Infographic] Challenges in Test Automation: Survey Key Results. [online] Medium. Available at: https://medium.com/katalon-studio/infographic-challenges-in-test-automation-survey-key-results-959e5c0e0fc1 [Accessed 21 Jan. 2019].