DOI : https://doi.org/10.56452/7-6-122

ISSN: 0974-5823 Vol. 7 No. 6 June, 2022

International Journal of Mechanical Engineering

SOFTWARE TESTING TOOLS AND FRAMEWORKS WITH VISUAL GUI TECHNIQUES IN INDUSTRIAL PRACTICE

Priyanka Yadu
School of Information Technology, Mats University, Raipur, C.G., India
(priya_tc@rediffmail.com) https://orcid.org/0009-0002-9783-0926

Dr. Bhawana Narain
School of Information Technology, Mats University, Raipur, C.G., India
(narainbhawna@gmail.com)

ABSTRACT

Software testing with visual Graphical Visual Interface (GUI) techniques is being done while maintaining a high level of accuracy with the ever-growing demand for speedier delivery of high-quality software, often known as "Quality at Speed". The use of relevant testing methodology(s) and the choice of suitable test automation tools and frameworks are two key components for a successful and efficient software testing project. A mix of numerous distinct testing processes is often necessary when testing software to make sure it is up to par; the use of a single testing method will not be sufficient. Similar to the previous point, finding the right tool combination for automated testing is difficult because no single tool can meet all of the needs. The first step in performing successful and efficient software testing is to familiarise industrial professionals with the various testing techniques, tools, and frameworks. An in-depth examination of the various test automation tools and frameworks is provided in this paper. An explanation of the various frameworks for test automation was delivered after an overview of automated testing and the categories it fits under. Finally, a brief summary of a few of the most popular automation solutions was provided, along with a comparison of those programmes.

Keywords: - Fasting, delivery, Automation tools, Testing methods and Practices

INTRODUCTION

Software testing is a crucial aspect of software system quality assurance. However, software testing is costly and accounts for roughly half of a typical software project's development costs [30]. Hence, software testing is referred to as a formal process in which a single piece of software, a group of connected software components, or a complete package is tested by executing the programmes on a computer. On approved test cases, all associated tests are run in accordance with approved test procedures [12]. GUI is one of the unique forms of software testing that is often used to examine the graphic user interface features of an application or piece of software. The majority of the efforts that have been put into researching software testing have been focused on developing new methods and determining how well they function in real-world development settings. Testing approaches have had a persistent hard time keeping up with the more rapid shifts in paradigms across the entirety of the software development process.

The GUI testing process is either manually or automatically implemented or repeatedly executed by a third-party organization, as opposed to the developers or end-users. The implementation of a graphic user interface is required for the execution of the other categories of software testing techniques. It is anticipated that in the not too distant future, the marketing of software will place a greater emphasis on the content of the software. As a consequence of the deterioration of this situation, testing procedures are becoming an increasingly significant component of the whole picture. Unfortunately, the real software testing and quality assurance (QA) practises that are utilised by software professionals are not extensively documented by recent statistics. As a result, in order to make an effort to find these practises, a comprehensive reviews of software testing strategies [10]. The primary objective of any form of testing is to deliver a high-quality product that satisfies the customer's requirements and is bug-free. Shift-left testing is a process that can be implemented earlier in the SDLC (Software Development Life Cycle) to enhance the product's quality. Instead of waiting until an application is complete to perform system testing, the development teams devote more time and resources to unit and interface testing. Consequently, early error detection in the development process will reduce the costs associated with their correction.

Recent research on software testing for visual GUI has focused primarily on the development of new methods and the evaluation of their efficacy in real-world development scenarios. This has been the principal focus of the majority of research endeavours [11]. Throughout the entire history of software development, testing approaches have had a difficult time keeping up with the ever-faster trends in software development paradigms. To bring about a positive shift in the current state of practise, it is necessary to make substantial efforts in predicting future trends, gaining a comprehension of the perspectives of stakeholders, and identifying problem areas in software testing.[3]

Recent efforts in software testing research have focused primarily on the development of novel methodologies and the assessment of their applicability to actual development scenarios with visual GUI features. Current software testing research has been on-going for quite some time. This is a relatively recent development in the academic discipline. Throughout the entire history of software development, testing methodologies have struggled to keep up with the ever-increasing rate of change brought about by alterations in software development paradigms [5]. The testing with GUI has excellent features for software testing mechanisms. The GUI testing methods (Figure 1) for software are utilized to execute or allocate tests on a Selenium Grid with a fixed Selenium Web Driver. GUI testing will enable us to assess an application's functionality from the user's perspective. Sometimes the system's internal performance is correct but the user interface is not; therefore, GUI testing is an excellent method for testing other types of applications. If there is to be a positive shift in the present state of practise, significant effort will be required in the areas of trend prediction, gaining an understanding of the mentalities of stakeholders, and identifying software testing problem areas in GUI. In order for there to be a positive shift in the current state of practise, there must be a positive shift in the current state of practise [6].

NEEDS OF SOFTWARE TESTING WITH VISUAL GUI TECHNIQUES FOR INDUSTRIAL ASPECTS

Software testing is the process of putting a product through its trials so that developers can identify bugs and other issues while it is operational. As a component of the quality assurance procedure, it contributes in some way to the final outcome. With the aid of this resource, software developers are able to construct error-free products. In addition, it assists in validating a product in accordance with consumer demands and expectations. SDLC stands for "**software development lifecycle**," which refers to the procedure used by the software industry to create new software and it is effective with GUI visual techniques. Typically, there are five processes involved in a scenario (visual GUI testing). This process encompasses the phases of analysis, design, implementation, testing, and maintenance. The software development process begins with the client submitting requirements, which serve as the process's starting point. The process continues through Analysis, Design, Implementation, Testing, and ongoing maintenance. After the implementation phase concludes, it is hypothesised that testing will commence. This is anticipated to take place at some stage in the future. In actuality, testing is a parallel procedure that begins with the accumulation of requirements for the evaluated product or service. After providing an explanation of the project's requirements, it is necessary to conduct a parallel check. Imagine for a moment that an error can be discovered before it is implemented, as opposed to after it has already been implemented [9].

The need of software testing is to detect defects and other potential issues within a system. Using the most effective and efficient test cases, it is possible, with a high degree of probability, to find defects that were not previously discovered and undetected. The testing phase of a project is both the most essential and the most expensive phase. It is essential that testing consume forty percent of the total effort expended. In spite of this, it is not always feasible to guarantee that software is bug-free. If there is a defect in the customer-delivered product, it is the responsibility of the testing team to identify it and make the necessary adjustments. Because of this, it is crucial that the software developed by software evaluators contains no flaws. The qualifications of an examiner consist of operability, observability, controllability, decomposability, simplicity, stability, and the capacity to comprehend.

In industrial process, the advancements in software processes, methods, and solutions, software complexity is increasing at an exponential rate and software developers face greater challenges than ever before. It is of the utmost importance that software engineers have access to the proper software tools in order to increase their capacity to produce high-quality software products effectively and efficiently. The industry with software testing methods examines the most recent practises that have emerged in the industry of software tools, as well as the most recent and anticipated future developments in the creation of software tools. The applications throughout the software lifecycle, but our primary focus is on aspects of software tools that have changed significantly in recent years and are expected to change significantly in the near future as tools continue to advance. In other words, we provided an overview of the tool applications that occur throughout the software development lifecycle [4]. The internal structure of tools, the availability of multiple view interfaces, integration options, collaborative work support, and an increasing reliance on automated assistance within tools are some of the characteristics.

GUI TESTING METHODS USED IN INDUSTRIAL PRACTICES

Experts in industry who are doing many software testing techniques with GUI and rectify the errors in software development and is effective with the techniques, they use to conduct using evolutionary testing. In recent years, many researchers have made some encouraging discoveries regarding automated functional testing, also known as black-box testing. However, despite the encouraging results, these techniques have had only limited success when applied to complex systems in the real world. As a direct result, information regarding the scalability, applicability, and acceptability of these approaches in the corporate sector is scarce. Graphical

DOI : https://doi.org/10.56452/7-6-122

User Interface testing must be performed and the development of the software product depends on how the GUI interacts with the end-user and facilitates the use of its various features. Graphical User Interface testing is essential for ensuring that an application looks and functions uniformly across multiple platforms and browsers. Therefore, GUI testing is very important because it ensures a substantial customer base and business value. In GUI testing, Manual GUI testing can sometimes be a repetitive and laborious procedure. However, Automation is strongly recommended for GUI testing. There are distinct types of testing techniques for GUI-based software, including analog recording and object-based recording. Analog Recording is the first form of graphical user interface testing. Using analog recording, individuals will always have access to the GUI testing tools. The GUI testing tools are used to capture the precise keystrokes, mouse movements, and other user actions and then save them to a file for playback. Let's examine an illustration to comprehend the fundamental functionality of analog recording. A second form of GUI testing is object-based recording. In this manner, the testing tool can programmatically connect to the application that requires testing and observe each of the specific user interface modules, such as a text box, icon, and hyperlink, as a separate object [24].

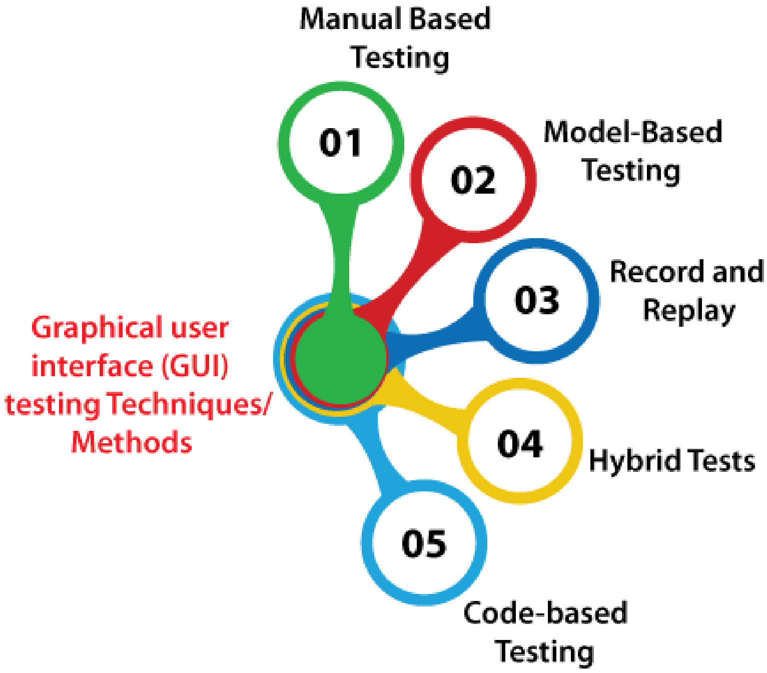

Figure 1. GUI Testing Techniques

The first testing procedure for GUIs is manual testing. Manually utilizing the application is the most straightforward method for GUI testing. Model-based Testing was necessary because we knew that a model is a visual narrative of System performance that enables us to comprehend and predict the performance or activity of the system. There are two categories of GUI testing that can be performed with the aid of Automation tools. Throughout the recording phase, the automation tool encapsulates the test processes. In addition, these recorded test steps are implemented on the application under test during playback. The record and replay method requires a test engineer to use a particular instrument to record a testing session. The substantial advantage of the Record and replay method is that it does not require coding expertise, which lowers the barrier for us to use it. The primary disadvantage of record-and-playback tests is their vulnerability. Currently, hybrid tests represent an alternative method for GUI testing. It is advantageous for non-technical users to record their sessions in

DOI : https://doi.org/10.56452/7-6-122

order to develop a test case. And then, a user who is conversant with coding can technically control these recorded tests [24].

Other Software Testing Methods used in Industry

White Box testing

During this testing, the internal workings of the system as well as its structure are made visible. As a result, it is a very cost-effective method for locating and fixing issues, as defects are often detected before they become a source of inconvenience as a result of using this method. Testing in the white box is also sometimes referred to as testing in the clear box, white box analysis, or just plain old box analysis. It is a technique for finding flaws in which the tester is provided with comprehensive information regarding the operation of the various program components. This methodology is not utilized frequently for the purpose of debugging large systems and networks; rather, it is utilized for the purpose of developing applications for the internet. Some examples of different kinds of white box testing include control structure testing, basic path testing, and loop testing [19].

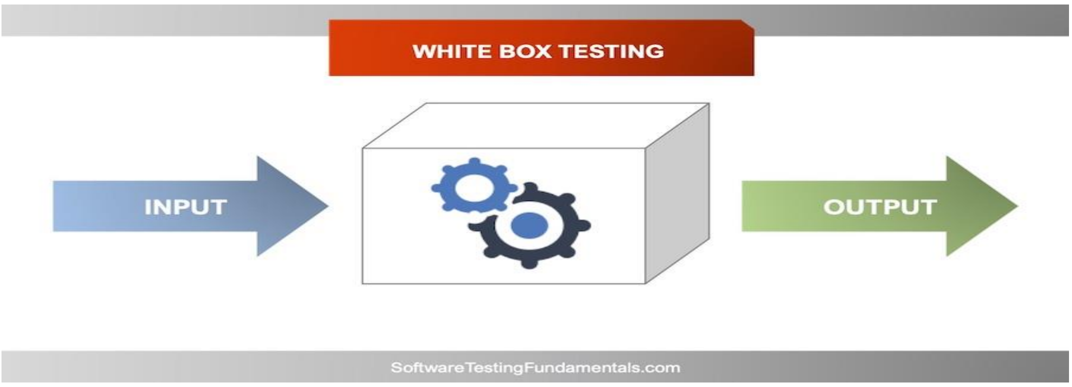

Figure 2. WHITE BOX TESTING

Black Box testing

It is claimed that a piece of technology is a "black box" when the one who makes use of it is unable to fully comprehend it or obtain access to its inner workings. This method of testing satisfies the requirements that the programme lays out for its output and supports the program's specifications, but it does not collect any information about the underlying structure of the programme. Finding out how well the system fulfils the requirements that have been established for the system is the primary goal of this project. When carrying out black box testing, very little or no information is gathered concerning the internal logical structure of the system being tested. As a result, it focuses solely on investigating the most fundamental component of the system. It ensures that each input is accepted properly and that outputs are created in the appropriate manner at all times. There are many different kinds of black box testing, some examples of which include the equivalence class partitioning test, the boundary value analysis test, and the cause effect graphing test [20].

DOI : https://doi.org/10.56452/7-6-122

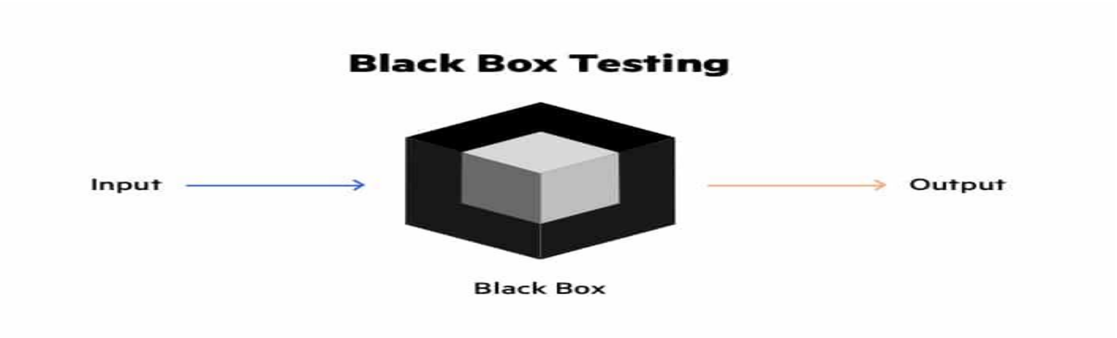

Figure 3. BLACK BOX TESTING

Grey Box Testing

In recent years, a third testing method known as "grey box testing" has been jointly considered alongside the other two methods. It is defined as testing a software package while simultaneously having some knowledge of the software's core logic and the code that underlies it. It favors white box testing over black box testing and makes use of the system's own information structures and algorithms for developing test cases. Black box testing is also performed. This approach involves performing reverse engineering in order to determine boundary values. Because it does not require the tester to have access to the application's internal ASCII text file, grey box testing is objective and does not intrude on the user experience [21].

Black Box Testing
+
White Box Testing
=
Gray Box Testing

Figure 4. GRAY BOX TESTING

Manual Vs Automated Testing

The testing of software can be carried out using one of two primary approaches:

Testing Software by Hand Is Called "**Manual Software Testing,**" and it's exactly what it sounds like: the process of testing software by hand, either by one person or by several people working together**. Automated Software Testing** is the process of producing test scripts that can then be run automatically, repetitively, and through a great deal of iteration automated software testing is the process of producing test scripts that can then be run automatically. Obtaining the appropriate ratio of software testing an efficient method for testing software often consists of a variety of different sorts of tests that are carried out in a manner that combines both manual and automated testing. The quality criteria of the application will dictate the mix of tests to be performed as well as the total number of tests. Each approach, whether it be automatic or manual, is applied according to the circumstances [22].

Manual testing is most effectively utilized for tests that call for spontaneity and inventiveness, in addition to a significant amount of subjectivity, user interface or usability testing, and exploratory/ad hoc testing. Automated testing is most effective when applied to tests that are specific and repeatable.

DOI : https://doi.org/10.56452/7-6-122

Table 1: Comparison between Manual and Automation Testing [23]

Manual Testing	**Automation Testing**
The simple low-level type for each QA runs all tests without using any software	A QA uses special tool for running the tests.
A time-consuming process if it is done	Time-saving, less manual effort, QA can re-run same tests again and again
Can be repetitive and boring	Helps to avoid repetitive tasks as QA delegates them to computer.
It is suitable for any software products	It is suitable for stable system and used mainly for regression
It helps to define whether automation setting is possible or necessary	100% automation is not possible in this scenario

Due to this, the industry is mainly tried to opt automation testing as means of assisting software developers with the testing of their products for generating high-quality test cases. Throughout the past several years, the use of manual testing has been supplemented by the investigation of numerous methodologies for the development of automated tests. The objective of this investigation was to complement manual testing. Even if there is some evidence to suggest that autonomously generated test suites may cover even more code than those manually written by engineers, this does not imply that these tests are effective at locating software flaws. This does not necessarily imply that these evaluations are beneficial. Through comparative research, the advantages and disadvantages of manual testing versus automated code coverage-directed test development should be determined. This is due to the fact that automated code coverage-directed test creation and manual testing are two inherently distinct techniques, and each has its own unique set of inherent limitations.[11]

In this study, we doctrinally reviewed the automated test creation and compare it to test suites prepared manually by industrial engineers for 61 programmes from an actual industrial train control system. Using a real-world industrial train control system, a comparison is conducted. To achieve this, the information collected by the genuine industrial train control system (or their websites). This system contains software that was developed using IEC 61131-3 [10], a language frequently employed in the safety-critical industry for the creation of control software. This language was used to develop the included software for this system. According to the results of our case study, automated test generation can accomplish code coverage comparable to that of manual testing conducted by industrial engineers in a fraction of the time required by manual testing [21-22].

Automated test generation has the potential to reduce testing time by approximately 90% when designing software in conformance with IEC 61131-3. Even if complete code coverage is achieved, there is no guarantee that automatically generated test suites are superior to manually created test suites in terms of their ability to detect errors. 56% of test suites developed with COMPLETEEST detected fewer defects than test suites developed manually by industrial engineers. This distinction was discovered across all software categories. As a consequence, our case study led us to this conclusion. It appears that manually written tests can detect specific categories of bugs (such as logical replacement, negation insertion, and timer replacement) more effectively than computer-generated tests can [22]. When comparing manual tests to computer-generated tests, this is the case. If, in addition to structural characteristics, the identification of these specific modifications was used as the coverage criterion by the automated test generation tool, we could generate more effective test suites.[2]

CONTINUOUS INTEGRATION AND TESTING WITH GUI VISUAL INTERFACE

The software development discipline known as "continuous delivery" refers to the practise of producing high-quality software that can be deployed into production at any time. This is a possibility at any time. However, despite the fact that written instructions on how to put it into practise can be found in relevant research, there have been a great deal of difficulties in doing so. The examination procedure is one of these most difficult obstacles. On the one hand, the relevant corpus of academic research has uncovered a number of Continuous Delivery testing challenges. According to a number of sources, Continuous Testing is the aspect of Continuous Delivery that is deemed to be absent.

Visual GUI testing (VGT) is the third iteration of GUI-based testing methods [25]. It is a tool-driven technique for interacting with and asserting the behaviour of a given System Under Test (SUT) using image recognition. The advantage of VGT is its adaptability to any GUI-based system. Due to the relative immaturity of VGT technologies and instruments, however, studies have reported robustness issues with VGT. In particular, Alegroth et al. [26] discovered that faulty image recognition could lead to false test results. Evidently, efficient and effective implementation of test automation in the software industry, particularly for VGT, may be difficult. Particularly, based on the authors' experience, when VGT and GUI test automation are not properly planned, designed, or implemented by test engineers, the efforts have resulted in disappointments and various negative outcomes (such as test artifacts becoming less useful or even unusable for regression testing).

We investigated a broad range of testing issues and evaluate a vast array of proposals, methodologies, approaches, methods, frameworks, tools, and solutions. We attempted to determine whether Continuous Testing is the missing component of Continuous Delivery by examining the various definitions of Continuous Testing and the testing phases and levels that are part of Continuous Delivery. In addition, we reviewed the various implementations of Continuous Testing. We have reviewed a number of problems have not yet been resolved. [6] We've observed that the first-generation VGTs are almost extinct (i.e., they're rarely used) and that, when test teams want to conduct automated GUI testing, they use the second-generation tools. The adoption of VGT (3rd generation) instruments is beginning to increase in the industry. To determine the type of GUI or VGT testing instruments to be selected and used [27], the industry considers a number of factors, such as the type of GUI being tested ("native", web, or mobile application). There are many commercial and open-source utilities available for each of the aforementioned GUI types. Such a discussion is beyond the scope of this paper, but interested readers can consult online resources such as [28].

VISUAL GUI SECURITY TESTING IN INDUSTRIAL PRACTICES

An external security testing team almost always conducts a security assessment of the application at the project's conclusion, either following or in conjunction with user acceptance testing. Visual GUI security testing in industry ensures that only authorized personnel can access the program and features made available to them based on their security level by utilizing a variety of testing tools for security checks. Abbot, Jemmy, JFC Unit, Jacareto, and Marathon are industry-standard Java GUI testing utilities [29]. Moreover, encryption and decryption are the security techniques used for assessing the security of visual GUI techniques.

By encrypting the application, utilising a variety of software, hardware, and firewalls, among other measures, the security testing is done to determine whether there has been any information leakage. Due to the proliferation of cyber-physical systems (CPS), testing automation applications have become an integral part of every production systems engineering (PSE) endeavour. In light of new attack vectors against CPSs, which have arisen in part as a result of growing connectivity, security considerations must be incorporated into each phase of the PSE process. Increased connectivity has contributed to the emergence of novel attack vectors against CPSs. Numerous valuable assets, such as system configurations and production information, are susceptible to information theft and subversion due to the absence of adequate security measures [12].

Therefore, software testing in industry has become an activity of paramount importance as they usually adopt other software testing techniques like manual or automated. Automated testing techniques provide efficiency but the security features are not much effective in automated (Table 2). In addition, because there are insufficient safety systems in place, the significance of software testing has increased there. The only method for businesses to protect themselves from the dangers posed by these threats is to conduct routine security checks on the software testing techniques they use. On the other hand, these efforts may be doomed to failure if there is insufficient expertise in the field of security or if there is insufficient funding to cover security-related expenditures [14].

This includes data flows, assets, entities, hazards, and mitigation strategies. The German-developed VDI/VDE 2182 recommendation functions as its foundation. The structure of the framework includes a default testing method model. Users are able to modify this model in order to better align the inspection objective with the environment in which they conduct software testing. In particular, the testing technique being considered for the production of the default model conforms to the ISO/IEC/IEEE 29119 set of software testing standards and is based on the best practises observed by a prominent system integrator. It ensures the default model is accurate and trustworthy. In addition, we developed a programme capable of automating the construction of attack-defence trees by using formal models of a company's software testing procedure as a starting point [16]. A structure in the shape of a tree was utilised to achieve this objective. The findings of the illustrated security analysis provide guidance, are intended to raise awareness in the industrial sector, and are intended to facilitate the efficient, effective, and timely execution of security studies [6].

TESTING TOOLS AND ITS CRITERIA FOR INDUSTRIAL PRACTICES

A series of procedures known as testing are carried out to evaluate the calibre of software [12]. GUI software testing always takes place at one of the following four test levels: unit testing (also known as component testing) when the target is a standalone software component, integration testing when the target is a standalone subset of a software system, system testing when the target is a standalone software system, and acceptance testing when the target is a standalone software system and the objective is to determine whether the software system as a whole is acceptable to the end-user. [13]

Taking into account the numerous considerations that must be made prior to selecting a tool, it is possible that selecting the best tool for software testing with VGT could be a difficult task at times. The success of test automation depends on a variety of fundamental factors, one of which is the choice of a testing instrument. To achieve this, one must first become familiar with the breadth of testing and the testing methodology, and then select the appropriate testing tool to meet the requirements of automating the test suite for a specific product and release. A testing instrument can be used to test desktop applications, mobile applications, or any combination of the three [15]. A testing instrument may also include any testing capabilities, such as unit testing, regression testing, and integration testing, and so on. The following testing tools were selected for evaluation based on a set of inclusion and exclusion criteria that were applied to

DOI : https://doi.org/10.56452/7-6-122

the testing tools that have received the most attention in the relevant scholarly literature and are also among the most popular choices among industry practitioners. These instruments are briefly described, followed by a table-based comparison of their benefits and drawbacks. The comparison is founded on a variety of factors, including reusability, dependability, and price [18]. Table 2 explores the automation tools being used by the industry as given below:

Table 2: Automation Tools being used by Industry (with their ratings)

NAME	RATINGS
SELENIUM	9.5
APPIUM	9.5
ROBUST FRAMEWORK	9.3
CUCUMBER	9.1
CYPRESS	9
GUAGE	8.5
DOJO TOOLKIT	7.8
WATIR	7.6
TOX	7
SERENITY	7
TESTSIGMA	6.9
CARINA	6.9
NOX	6.8
GALEN	6.6

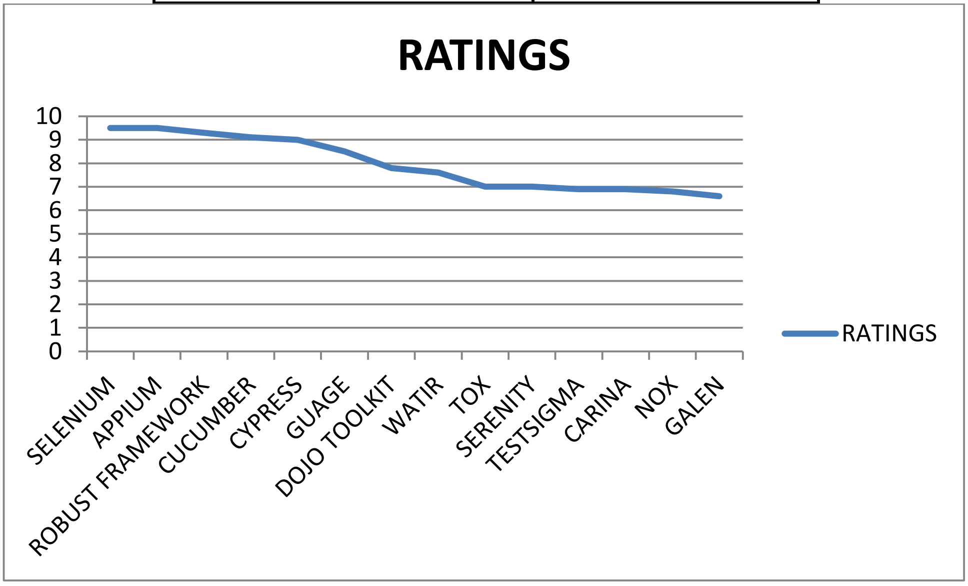

Figure 5. Automation Tools Ratings

From the table 2, it has been seen most of the automation tools being used in the industry is “Selenium” and it has high rating for them as it supports in enhancing the software testing experiences.

CONCLUSION AND RECOMMENDATIONS

From the paper, it has been observed the VGT software testing has become an essential phase for many industries. It ensures that the software that has been released to the public is free of bugs and side effects. Automated testing tools are preferred over manual testing methods by software evaluators because they reduce testing-phase costs and save time and being used by industry very well this time. Also the VGT for software testing seems easier for the industrialists and many industry it is being used. This is done to meet market requirements and time constraints. When the proper testing tool is selected, software testers will be able to select the best tools for testing apps with simplicity, saving them both time and money. There is no single ideal testing tool, but that compromises can be made to select the best GUI testing tools based on the scope of the project, the testing budget, the platform of the application, and the programming language used to develop the project. Based on the results of this study, we recommend utilising Test Complete and Ranorex as testing tools for all platforms. Since both of these programmes require licences, when evaluating a large project, the testing budget should be considered. While selenium is recommended for web testing and has the benefit of being open source, appium should only be used to test mobile applications. For the study, it is also suggested that future work encompass additional tools and criteria. Depending on the environment and budget, Selenium Webdriver, UFT, Ranorex, RFT, JMeter, and Appvance are the most frequently used tools, as determined by the review. However, there is no single method or framework that can enable fully automated web testing and meet all requirements. CCS concepts include software engineering, creation, and management, as well as software verification and validation.

This study will assist industrial professionals in selecting the optimal tool for a specific project, and it will also enable researchers to compare additional tools using more criteria where VGT testing techniques can be very much helpful. A high-quality, client-requirement-compliant software, frameworks and tools for automating software testing must be used appropriately in industries for enhancing their practical experience. Despite the fact that there have been numerous research studies on software testing and automated testing technologies, detailed standards are necessary. A variety of functional, load, and management testing tools have been analysed and compared based on similar properties, such as platform support, scripting language employed, browser compatibility, etc. Although manual testing tools are also available, this article focuses exclusively on web-based automated testing solutions. This study examines the pertinent literature to identify and summarise the available free source and paid automated web testing technologies, as well as the challenges they face. The quality for software testing is the most important aspect in every software engineering projects, hence, we recommend taking into account the project's scale, testing budget, and target platform when selecting a testing tool.

REFERENCES

1. Albarka, U. M., &Zhanfang, C. (2019). *A study of automated software testing: Automation tools and frameworks*. https://doi.org/10.5281/ZENODO.39247
2. Enoiu, E., Sundmark, D., Causevic, A., &Pettersson, P. (2017).A comparative study of manual and automated testing for industrial control software. *2017 IEEE International Conference on Software Testing, Verification and Validation (ICST)*.
3. Kassab, M., DeFranco, J., &Laplante, P. (2016). Software testing practices in industry: The state of the practice. *IEEE Software*, 1–1. https://doi.org/10.1109/ms.2016.87

4. Kumari, B., Chauhan, N., & Tech Scholar, M. (n.d.). *A comparison between manual testing and automated testing*. Jetir.org. Retrieved June 25, 2023, from https://www.jetir.org/papers/JETIR1812949.pdf
5. Sehgal, M., Sharma, S., & Mam, D. G. (n.d.). *Manual & Automated Testing*. Ijert.org. Retrieved June 25, 2023, from https://www.ijert.org/research/manual-automated-testing-IJERTV2IS4067.pdf
6. Mascheroni, M. A., &Irrazábal, E. (2018). Continuous testing and solutions for testing problems in continuous delivery: A systematic literature review. *Computación y Sistemas*, *22*(3). https://doi.org/10.13053/cys-22-3-2794
7. Okezie, F., Odun-Ayo, I., & Bogle, S. (2019). A critical analysis of software testing tools. *Journal of Physics. Conference Series*, *1378*(4), 042030. https://doi.org/10.1088/1742-6596/1378/4/042030
8. Vos, T. E. J., Marin, B., Escalona, M. J., & Marchetto, A. (2012). A methodological framework for evaluating software testing techniques and tools. *2012 12th International Conference on Quality Software*.
9. Gadwal, A. S., & Prasad, L. (2020). *Comparative review of the literature of automated testing tools*. Unpublished. https://doi.org/10.13140/RG.2.2.36836.19848
10. Garousi, V., Felderer, M., Kuhrmann, M., Herkiloğlu, K., & Eldh, S. (2020). Exploring the industry's challenges in software testing: An empirical study. *Journal of Software (Malden, MA)*, *32*(8). https://doi.org/10.1002/smr.2251
11. Causevic, A., Sundmark, D., & Punnekkat, S. (2010). An industrial survey on contemporary aspects of software testing. *2010 Third International Conference on Software Testing, Verification and Validation*.
12. Konka, B. B. (n.d.). *Master of science thesis in software engineering and management*. Core.ac.uk. Retrieved June 27, 2023, from https://core.ac.uk/download/pdf/16333079.pdf
13. Bäckström, K. (n.d.). *Industrial surveys on software testing practices: A literature review*. Helsinki.Fi. Retrieved June 27, 2023, from https://helda.helsinki.fi/bitstream/handle/10138/340855/Kim_Backstrom_thesis_2022.pdf?sequence=2&isAllowed=y
14. Garousi, V., Keleş, A. B., Balaman, Y., Güler, Z. Ö., & Arcuri, A. (2021). Model-based testing in practice: An experience report from the web applications domain. *The Journal of Systems and Software*, *180*(111032), 111032. https://doi.org/10.1016/j.jss.2021.111032
15. Reine De Reanzi, S., & Ranjit Jeba Thangaiah, P. (2021). A survey on software test automation return on investment, in organizations predominantly from Bengaluru, India. *International Journal of Engineering Business Management*, *13*, 184797902110620. https://doi.org/10.1177/18479790211062044
16. Wang, Y., Mäntylä, M. V., Liu, Z., & Markkula, J. (2022). Test automation maturity improves product quality—Quantitative study of open source projects using continuous integration. *The Journal of Systems and Software*, *188*(111259), 111259. https://doi.org/10.1016/j.jss.2022.111259
17. Hogan, M. D., Carnahan, L. J., Carpenter, R. J., Flater, D. W., Fowler, J. E., Frechette, S. P., Gray, M. M., Johnson, L. A., McCabe, R. M., Montgomery, D., Radack, S. M., Rosenthal, R., & Shakarji, C. M. (2001). Information technology measurement and testing activities at NIST. *Journal of Research of the National Institute of Standards and Technology*, *106*(1), 341–370. https://doi.org/10.6028/jres.106.013

DOI : https://doi.org/10.56452/7-6-122

18. Wohlin, C. (2013). Empirical software engineering research with industry: Top 10 challenges. *2013 1st International Workshop on Conducting Empirical Studies in Industry (CESI).*
19. Nidhra, S. (2012). Black box and white box testing techniques - A literature review. International Journal of Embedded Systems and Applications, 2(2), 29–50. https://doi.org/10.5121/ijesa.2012.2204
20. RedStone Software. (N.d.). Black-box vs. White-box Testing: Choosing the Right Approach to Deliver Quality Applications Retrieved June 25, 2023, from https://www.cs.unh.edu/~it666/reading_list/Defense/blackbox_vs_whitebox_testing.pdf
21. Testing, W. is G. (n.d.). Gray Box Testing. Idc-online.com. Retrieved June 22, 2023, from https://www.idc-online.com/technical_references/pdfs/information_technology/Gray_Box_Testing.pdf
22. Kumari, B., Chauhan, N., & Tech Scholar, M. (2018). A comparison between manual testing and automated testing. Jetir.org. Retrieved June 24, 2023, from https://www.jetir.org/papers/JETIR1812949.pdf
23. What are the Benefits of Automation Testing? (2019, November 6). UTOR. https://u-tor.com/topic/automation-testing-benefits
24. GUI testing. (n.d.). Www.javatpoint.com. Retrieved July 5, 2023, from https://www.javatpoint.com/gui-testing
25. Alegroth, E., Nass, M., & Olsson, H. H. (2013). JAutomate: A tool for system- and acceptance-test automation. 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation.
26. Alégroth, E., Feldt, R., & Ryrholm, L. (2015). Visual GUI testing in practice: challenges, problems and limitations. Empirical Software Engineer, 20(3), 694–744. https://doi.org/10.1007/s10664-013-9293-5
27. Raulamo, P., Mäntylä, M. V., & Garousi, V. (2017). Choosing the right test automation tool: a Grey literature review. In International Conference on Evaluation and Assessment in Software Engineering (pp. 21–30).
28. Banerjee, I., Nguyen, B., Garousi, V., & Memon, A. (2013). Graphical user interface (GUI) testing: Systematic mapping and repository. Information and Software Technology, 55(10), 1679–1694. https://doi.org/10.1016/j.infsof.2013.03.004
29. Atif, M., Memon, M. L., & Soffa, M. E. (n.d.). Plan generation for gui testing.
30. Britton, T., Jeng, L., Carver, G., Cheak, P., & Katzenellenbogen, T. (2013). Reversible Debugging Software.