ISSN No:-2456-2165

Comprehension of Software Testing and Impacts of Automation Testing

[1]Priyanka Prakash Sawant
University of Mumbai
Institute of Distance & Open Learning
(IDOL), Mumbai, India

[2]Ritu Bisht
University of Mumbai
Institute of Distance & Open Learning
(IDOL), Mumbai, India

[3]Neha Mahesh Raut
Assistant Professor (AI&DS)
Vidyavardhini's
College of Engineering and Technology

Abstract:- In order to find and fix errors and assure the quality and dependability of software applications, software testing is a critical step in the software development lifecycle. It entails comparing the functionality, efficiency, security, and usability of a software product to predetermined standards and guidelines. Software testing's main objective is to find and fix defects, guaranteeing that the finished product is free of serious flaws and satisfies user expectations. The essential principles, approaches, and goals of software testing are covered in this abstract as a whole. It emphasizes how important testing is to producing a reliable and bug-free software solution and lowering the possibility of expensive post-release faults. Unit testing, integration testing, system testing, and acceptance testing are just a few of the testing layers that are discussed

The abstract also discusses test automation, a crucial component of contemporary software testing that facilitates continuous integration and delivery practices while speeding up test execution and increasing test coverage. Testers may manage test cases and data effectively with the help of test automation tools and frameworks, which streamlines the testing process overall. In conclusion, software testing is essential for assuring software quality, fulfilling user expectations, and creating applications that are dependable, secure, and high-performing. Software development companies may increase the stability of their software, lower maintenance costs, and ultimately produce products that offer a great user experience by utilizing the right testing approaches, embracing test automation, and adhering to industry best practices.

Keywords:- *Software Testing Life Cycle, Software Testing Techniques Automation in Quality Assurance, Analysis of Automation Testing Tools.*

I. INTRODUCTION

Software testing is a critical phase in the software development process that ensures the quality and reliability of software applications. It involves the evaluation of a software product to identify defects, errors, or discrepancies between expected and actual outcomes. By validating and verifying that the software meets its specified requirements and adheres to industry standards, testing helps build confidence in the product's functionality, performance, security, and usability

A. Manual Testing:

A traditional method of software testing is manual testing, in which test cases are carried out by hand without the aid of automated technologies. With this approach, testers mimic end-user behaviour as they use the application to test various features, user interfaces, and capabilities. Although manual testing requires human labour, it has many benefits, including the capacity to react to last-minute modifications, a more thorough method of evaluating the user experience, and cost-effectiveness for small-scale projects.

B. Automation Testing:

Automation testing entails running pre-defined test cases through automated tools and scripts and comparing the results to what was anticipated. This method greatly quickens the testing procedure, requires less human work, and enhances test precision and reproducibility. Regression testing and situations requiring the execution of several test cases profit most from it.

II. LITERATURE REVIEW

Software testing is an essential stage in the development process that looks for defects and ensures the dependability and quality of software programs. Manual testing is a time-honored technique for testing software, in which test cases are performed manually without the use of automated tools. This literature review offers a general overview of the core concepts, methods, and best practices in manual software testing while also examining its benefits, drawbacks, and role in modern software development.

A. Importance of Manual Testing

Numerous studies emphasize the importance of manual testing in the software development process. Manual testing allows testers to apply their domain knowledge, experience, and creativity to identify usability issues and potential edge cases that may be overlooked in automated testing. Bhagyashree A. Shegokar and Rakesh R. Gupta (2017) in their research highlighted that manual testing is valuable in uncovering user experience issues and providing comprehensive feedback on the application's functionality.

B. Testcase Design & Execution

The literature often discusses the test case design and execution techniques in manual testing. Nandini Nandhakumar et al. (2019) conducted a comparative study of manual and automated test case design techniques. Their findings revealed that manual test case design can be more effective in uncovering specific defects and dependencies, particularly in complex applications. Additionally, manual execution allows for dynamic testing, where testers adapt to the system's behavior in real-time and make critical decisions based on their observations

C. Test Oracles and Subjectivity

One of the key challenges in manual testing is the subjectivity of test results and test oracles. Test oracles are used to determine the expected results, and in manual testing, this often depends on the tester's judgment. Catherine A. Wah et al. (2018) highlighted that the subjective nature of manual testing can lead to inconsistencies in test results and might require additional effort to ensure consistency and accuracy.

D. Limitations and Drawbacks

Several studies address the limitations and drawbacks of manual testing. Jie Zhang and Saurabh Sinha (2018) argued that manual testing can be time-consuming, resource-intensive, and prone to human errors, particularly in repetitive test scenarios. The lack of repeatability in manual testing can hinder the efficient detection of regression issues.

E. Integration with Automation

The literature also explores the integration of manual testing with automation. Studies suggest that a hybrid approach, combining both manual and automated testing, can lead to improved testing efficiency and effectiveness. Chan-Wei Lin et al. (2019) demonstrated that a well-planned combination of manual and automated testing can optimize testing efforts and maximize test coverage.

III. METHODS

A. Here is the Graphical Representation of the STLC:

Fig 1 Software Testing Life Cycle (STLC) – Manual

➢ *Requirement Analysis:*

- Understanding the software requirements and its scope of testing that has to be carried out.

➢ *Test Planning:*

- In this phase the test efforts, resource allocation and test environment is decided for the testing by the Senior/Lead QA.

➢ *Test Design:*

- Designing of Test Criteria/Scenario and Test Cases as per the requirement and scope.
- Test Scenarios/Cases Review is also done by Lead/Senior QA in this stage to make sure everything is covered.

➢ *Test Environment Setup:*

- The QA needs to configure the test environment so that the production settings can be replicated
- This involves installing required software, hardware, and configure same as production.

➢ *Test Execution:*

- Execution of the test cases in the test environment.
- Updating test results which includes pass/fail status.

➢ *Defect Reporting and Tracking:*

- Logging the defects found during test execution.
- Assign the defects to the development team so that the issue can be fixed.
- Tracking the status of defects until they are fixed and verified.

➢ *Test Closure:*

- Evaluating the testing process against the test plan.
- Preparing a test summary report with test results and activities.
- Obtaining approval for the closure of the testing phase.

B. Here is the Graphical Representation of the ATLC

Fig 2 Software Testing Life Cycle (STLC) – Automation

➢ *Automaton Scope Analysis:*

- First step is to narrow down the cases that can be automated as automation cannot provide 100% test coverage.

➢ *Automation Tool Selection:*

- This step involves us to use type of automation tools and language which will help us for test coverage and the knowledge about the tool.
- Costing and expertise has to be considered while tool selection.

➢ *Plan & Strategies:*

- Framework designing and cases format preparation. Deciding the data flow within the code.
- Experiment the device whether it supports the tools or any extra supporting files required.

➢ *Setup Test Environment:*

- The test environment setup is basically making the project which needs to be automated.
- It is required to check the framework designed is working or not on that project.

➢ *Test Scripting and Debug:*

- Test Scripting means coding as per the strategy and making sure the required variables, class, functions are declared appropriately and deign the cod reusable and maintainable.
- Debugging ensures the code the working, if not hen the necessary changes should be done.

➢ *Automation Run:*

- End to end run of all the prepared cases is done by Automation Tester once Scripting and Debugging the cases are completed.
- The full automation run ensures the framework can be reused again.

➢ *Log Analysis and Reporting:*

- Once Full run is completed the logs are ready to be analyzed and if any difference found during run then the same has to be reported.

C. Types of Software Testing Techniques

➢ *Black Box Testing*

Black-box testing, also called functional testing, is a technique where the tester examines the software's external interface and behavior without access to its source code or any consideration of its internal logic.

➢ *White Box Testing:*

White box testing, also referred to as clear box testing, is a method of testing in which the tester possesses knowledge about the inner mechanisms of the product, can access its source code, and verifies that all internal operations adhere to the specified requirements.

➢ *Functional Testing*

Functional testing is an essential method in software testing, which centers on confirming that a software application operates correctly and aligns with the predetermined criteria, executing its intended functions accurately. The key aim of functional testing is to verify the application's performance and features, as perceived by end-users. The examination assesses various aspects of the Application under Test, such as User Interface, APIs, Database, Security, and Client/Server communication. This evaluation can be carried out through either manual means or automation.

D. Types of Software Functional Testing

➢ *Unit Testing:*

Unit testing is centered around examining separate units or elements of the software independently to confirm their precision. This type of testing is conducted during the application's development stage.

➢ *System Testing:*

Evaluates the entire integrated system to ensure that it functions correctly as a whole and meets the specified requirements.

➢ *Acceptance Testing:*

Conducted by end-users or stakeholders to determine whether the software meets the business requirements and is ready for deployment.

➢ *Regression Testing:*

This testing method primarily focuses on ensuring that the code changes do not negatively affect the current functionality of the system. When bugs are fixed, regression testing is employed to check if all components are functioning as expected and to assess whether any impact on the system has occurred. The main objective of regression testing is to verify the system's stability post updates.

➢ *Smoke Testing:*

Smoke testing, a form of functional testing, focuses on validating the fundamental features of a system to ensure its core functionality is working. It is also referred to as "Build Verification Testing." The primary goal of smoke testing is to ascertain that crucial functions are operational. For instance, it verifies that the application can launch successfully and checks the responsiveness of the GUI.

➢ *Sanity Testing:*

Sanity testing is a part of regression testing, and its purpose is to verify that recent code changes are functioning correctly. It acts as a checkpoint to determine if further testing of the build can proceed. During sanity testing, the primary focus of the team is to validate the overall functionality of the application, rather than conducting exhaustive testing. Typically, this type of testing is conducted on builds that demand immediate production deployment, such as critical bug fixes.

E. Types of Software Non Functional Testing

➢ *Non-functional testing is a software testing approach that evaluates non-functional aspects, such as reliability, load capacity, performance, and accountability of the software. Unlike functional testing, its main objective is to assess the software system's responsiveness based on non-functional parameters. These parameters are not examined during functional testing and focus on gauging the software's behavior under various conditions.*

➢ *Performance Testing:*

Performance testing aims to identify and address the root causes responsible for the software's sluggish and constrained performance. The objective is to optimize the software's reading speed to achieve maximum efficiency.

➢ *Load Testing:*

Load testing is a process that assesses a system's capacity to handle increasing user loads effectively. This capacity refers to the system's ability to accommodate a growing number of simultaneous users.

➢ *Security Testing:*

Security testing is used to detect the security flaws of the software application. The testing is done via investigating system architecture and the mindset of an attacker. Test cases are conducted by finding areas of code where an attack is most likely to happen.

➢ *Stress Testing:*

Stress testing is a Non-Functional testing technique conducted as part of performance testing. It involves subjecting the system to an overload to assess its ability to withstand such stress. During stress testing, the system's performance is closely monitored to ensure its resilience under intense conditions.

➢ *Portability Testing:*

The test for portability evaluates the software's capability to operate smoothly on multiple operating systems without encountering any bugs. Additionally, the test also assesses the software's functionality when running on the same operating system but with different hardware configurations.

➢ *Reliability Testing:*

The testing is based on the assumption that the software system will function without any errors within the predefined parameters. It requires a specific amount of time and processes to execute the system. Additionally, the reliability test will not pass if the system encounters failures in specific predetermined situations. For instance, it is essential for all web pages and links to be dependable.

➢ *UI Testing:*

UI testing, also referred to as User Interface testing, offers a solution to address such concerns. Its primary purpose is to ensure that the software user interface (UI) is displayed accurately to every user. By conducting UI tests, one can verify that each element on a web page is appropriately sized, shaped, and positioned as intended. The objective of UI testing is to evaluate the visible output of an application and compare it with the expected design outcomes. In essence, this type of testing helps to identify "UI bugs," which are distinct from functional bugs, and focus on issues related to the appearance and layout of a page or screen.

IV. TOOLS USED FOR AUTOMATION TESTING

➢ *Selenium:*

Selenium, an open-source automated testing framework, serves as a valuable tool for web application validation across diverse browsers and platforms. With support for various programming languages like Java, C#, Python, etc., it enables the creation of Selenium Test Scripts. The testing conducted through this tool is commonly known as Selenium Testing. The tagline of Selenium confirms its role as an automation testing tool for web applications, making it the most dependable option among web automation testing tools.

➢ *Appium*:

Appium is a well-known open-source automated testing tool, is designed primarily for mobile applications. It enables automation of native, hybrid, and mobile web applications developed for both iOS and Android platforms. Appium operates on a server architecture and leverages automation frameworks provided by vendors. Its user-friendly setup and usage have contributed to its

widespread adoption. Over the past years, Appium has gained tremendous popularity and stability, solidifying its position as one of the top choices for mobile automation testing.

- *Rational Functional Tester (RFT):*

RFT, an acronym for Rational Functional Tester, is an automated testing tool developed by the Rational Software division of IBM. It is primarily used for automated regression testing of software applications. RFT's script, which integrates with Eclipse and employs Java as its scripting language, is stored as a .java file. This script has complete access to the standard Java APIs as well as any other APIs exposed through different class libraries. Additionally, RFT offers a comprehensive API, allowing users to extensively modify the scripts generated through the recorder. To facilitate control identification based on specified properties, the Rational Test Script class serves as the foundational class for any Test Script and provides a useful find API.

- *SoapUI*:

SoapUI is an open-source functional testing tool designed by Smartbear–a leader in Gartner Magic Quadrant for Software Test Automation. It provides a comprehensive API Test Automation Framework for Representational State Transfers.

- (REST) and Service-Oriented Architectures (SOAP).It is not an automation testing tool for web or mobile app testing; however, it can be a tool of choice to test API and services. It is a headless functional testing application, especially for API testing.

- *TWIN*:

Twin, an automation tool for Windows graphical applications, empowers users to script actions like button clicks, text entry, and result visualization. Originally created at eBay, it serves the purpose of enabling automated functional testing for Windows applications. Drawing inspiration from Selenium/Webdriver, Twin's design allows for analogous application in various scenarios.

- *AutoIT:*

AutoIT, an open source tool, enables the automation of diverse processes involving Windows and desktop applications. Its automation approach encompasses a blend of keystrokes, mouse movements, and window/control manipulation to accomplish various tasks. This tool is compatible with all versions of Windows and utilizes C# and VB as its scripting languages. As a third-generation programming language, AutoIT employs a classical data model and supports multiple data types capable of storing various kinds of information. With just a few lines of script, or by recording the processes using the AutoIT Recorder, any specific task can be automated effectively.

V. RESULT/ BENEFITS OF AUTOMATION TESTING

- *Automating test one-liners is the practice of crafting succinct and targeted automated test cases that can be swiftly executed with minimal effort. These tests bring several advantages, particularly when a rapid validation of specific functionalities is required. The benefits of employing automated test one-liners include the following: Units*

- *Swift Feedback:*

One-liner tests provide immediate feedback on the functionality being tested, facilitating early detection and resolution of issues before they escalate.

- *Efficient Regression Testing:*

One-liners are particularly useful for regression testing, enabling the quick verification of specific functionalities after code changes or updates.

- *Seamless Integration:*

One-liners can be easily integrated into continuous integration and continuous delivery (CI/CD) pipelines, supporting automated testing throughout the software development lifecycle.

- *Simplified Debugging:*

The concise nature of one-liner tests makes the debugging process more efficient, as failures are isolated to specific functionalities.

- *Expanded Test Coverage:*

By automating multiple one-liners that cover various aspects of the application, overall test coverage can be increased in a scalable and manageable manner.

- *Reusability:*

One-liner tests, which focus on testing individual functions or features, can be reused across multiple test cases and projects.

- *Test Prioritization:*

One-liners allow for easy prioritization of critical functionalities, enabling essential features to be tested first and gradually expanding test coverage.

- *Reduced Maintenance Effort:*

Due to their targeted nature, one-liner tests are less likely to require frequent updates, resulting in reduced maintenance efforts.

- *Enhanced Collaboration:*

One-liner tests can be readily shared among team members, promoting collaboration and knowledge-sharing within the testing team.

- *Low Overhead:*

Automating test one-liners requires minimal setup and overhead, making them an efficient choice for quick tests.

- *Integration with Development:*
  One-liners can be seamlessly integrated with the development process, allowing developers to run tests locally during development.

- *Scalability:*
  Automating test one-liners empowers teams to scale their testing efforts as needed, ensuring efficient testing in both small and large projects.

- *Productivity and Time Saving:*
  Automation testing enables testers to dedicate their attention to more critical testing activities, as automation takes care of repetitive and time-consuming tests. As a result, productivity is enhanced within the testing process.

- It is crucial to acknowledge that while one-liner tests offer these advantages, they may not be suitable for testing all scenarios comprehensively. For robust software quality assurance, a combination of one-liner tests and more extensive, detailed test cases should be employed. Utilizing these benefits, automation testing enables teams to achieve higher software quality with increased efficiency, resulting in enhanced customer satisfaction and faster time-to-market.

VI. SURVEY/STATISTICS

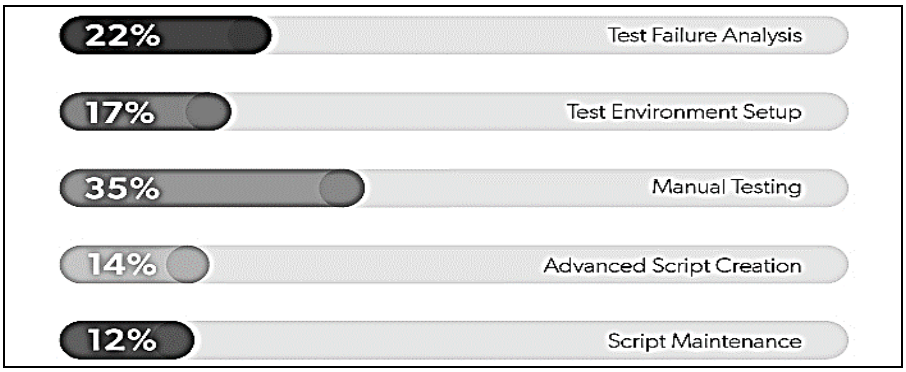

Fig 3 Time-Consumed within a Test Cycle for the Activities (2023)

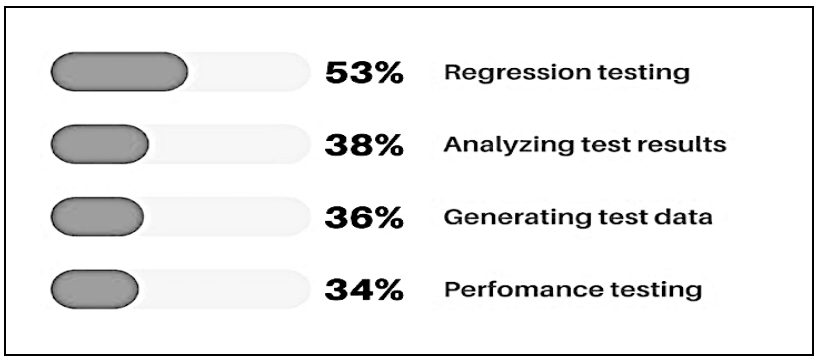

Fig 4 Purpose of Automation Testing (2023)

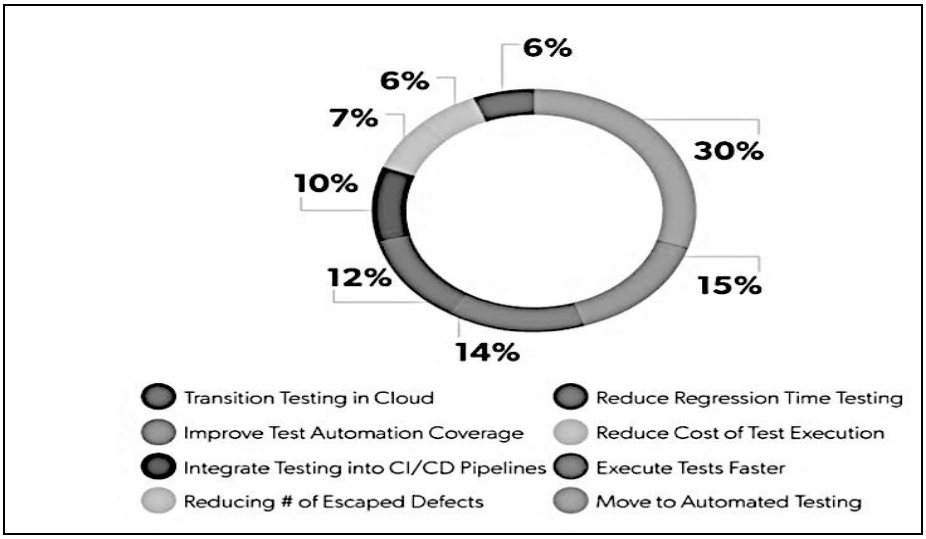

Fig 5 The Foremost Focuses in the Realm of Software Testing 2023

ISSN No:-2456-2165

VII. RESULT/ BENEFITS OF AUTOMATION TESTING

- *Complex Scripting:*

The creation of a scripting language similar to conventional programming was driven by the need to address certain challenges associated with automated tests. When not adequately planned, automated tests might take longer to develop compared to manual tests, and their integration into the development workflow can prove more challenging. Additionally, if the tests are complex and hard to manage, it can adversely affect the overall quality of the test suite, potentially impeding the goal of achieving continuous testing throughout the application's lifetime.

- *High Cost:*

Automated testing comes with a significant drawback, necessitating substantial time and financial resources for its implementation. Nevertheless, this initial investment frequently pays off swiftly through enhanced developer productivity and consistently dependable outcomes.

- *Requires Updating for Every New Environment:*

Whenever there is a modification in any environment, it is essential to update your automated tests to maintain their reliability. Regrettably, achieving functionality for your automated test scripts across various locations such as your local development environment, CI system, and production settings will require rewriting them multiple times.

- *Can Produce Failure Due to Automation Issue:*

Whenever there is a modification in any environment, it is crucial to update your automated tests to maintain their reliability. Regrettably, this means rewriting the automated test scripts multiple times to ensure they function correctly in your local development environment and production settings.

- *Needs Precise Designing:*

Creating a comprehensive set of automated tests is a challenging task, and they need to be reliable enough to be executed repeatedly and consistently, avoiding any inaccurate results (false positives or negatives). On the other hand, your test scripts should be flexible enough to accommodate changes made to your application. Achieving this demands precise planning, skillful implementation, and a deep understanding of the development process.

- *Cannot be used for GUI Validation Like Sound:*

Although automated tests are effective in verifying the majority of your application's functionality, they are inadequate for testing graphics or sound files. This is because such examinations typically rely on textual descriptions to validate the results

VIII. CONCLUSION

Automation testing provides several advantages over manual testing, making it the preferred approach for various scenarios, particularly for repetitive tasks and regression runs. While automation testing brings numerous benefits, it's essential to acknowledge that manual testing remains valuable in specific situations, such as exploratory testing, usability testing, and when dealing with frequently changing test cases. Employing a balanced approach that combines both automation and manual testing can lead to comprehensive and robust software testing strategies. It's worth noting that while automation testing cannot completely replace manual testing, it can significantly reduce the efforts of manual testers, ultimately increasing productivity.

REFERENCES

[1]. Heidilyn V. Gamido, Marlon V. Gamido "Comparative review of the features of automated software testing tools", 2019
[2]. Mubarak Albarka Umar 1*, Chen Zhanfang, A Study of Automated Software Testing:Automation Tools and Frameworks, 2019
[3]. Prachi Kunte 1, Prof. Dashrath Mane. "Automation Testing of Web based application with Selenium and HP UFT (QTP" June -2017
[4]. Monika Sharma, Rigzin Angmo "Web based Automation Testing and Tools" 2014
[5]. Sikender Mohsienuddin Mohammad "Automation Testing in Information Technology" 2015
[6]. Ritu Patidar, Anubha Sharma, Rupali Dave "Survey on Manual and Automation Testing strategies and Tools for a Software Application" 2017
[7]. https://dogq.io/blog/test-automation-statistics-for-making-the-right-decisions/