Optimizing Automated Software Testing with Machine Learning Techniques

Satish Kathiriya[1], Rajath Karangara[2], Narayana Challla[3]

[1]Software Engineer, JP Morgan & Chase Co.

[2]American Express, U. S

[3]ETL Developer.

Abstract: ***This paper investigates the application of machine learning (ML) techniques to enhance the efficiency of software testing automation systems. Recognizing the escalating complexities and the critical need for quality assurance in software development, our study focuses on leveraging ML algorithms to refine the testing process. The methodology encompasses a comparative analysis of conventional testing methods against our ML - integrated approach, measuring performance through accuracy, execution speed, and resource utilization metrics. Our findings reveal a notable enhancement in testing efficiency, with the ML model proficiently identifying and rectifying software anomalies. This advancement signifies a pivotal shift towards more intelligent, adaptable, and efficient testing mechanisms in software development. The research underscores the transformative potential of ML in software testing, proposing a new paradigm for future explorations in this domain. The implications extend beyond immediate testing improvements, providing a foundational approach for continuous advancement in software quality assurance.***

Keywords: Software testing, Manual Testing, Automation Testing, Machine learning

1. Introduction

The landscape of software development has been profoundly transformed by the advent of Artificial Intelligence (AI). This transformation is not just in the creation of software but extends to various aspects of its life cycle, including testing and quality assurance. AI's role in enhancing efficiency, accuracy, and reliability in software development processes has been increasingly recognized, laying the groundwork for more advanced and automated methods.

Among the various applications of AI in software development, machine learning (ML) has shown exceptional promise in automating and optimizing software testing. Traditional software testing methods, while effective, often grapple with limitations such as high time consumption and manual effort. ML offers a pathway to overcome these challenges, introducing capabilities that can learn from data, adapt to new scenarios, and improve over time. This has opened new frontiers in the automation of software testing, making it more efficient and less prone to human error.

This paper aims to explore and demonstrate the efficacy of ML techniques in software testing automation. By integrating ML into the testing process, we propose a novel approach that not only streamlines the testing phase but also enhances its accuracy and speed. The core argument of this research is that the incorporation of ML into software testing represents a significant leap forward in software development practices, offering a more efficient, accurate, and cost - effective solution to the challenges of traditional testing methodologies.

2. Software Testing: A Comparative Overview

With the escalating demand in software development, ensuring quality through effective testing has become paramount. Initially dominated by manual methods, software testing has evolved significantly. Manual testing, though integral in the early development phases, presents challenges like higher costs and time consumption, necessitating the shift towards automated methods, particularly those employing metaheuristic techniques for enhanced efficiency and cost - effectiveness [1].

2.1 Manual Software Testing: Characteristics and Limitations

Quality Assurance (QA) analysts primarily carry out manual testing, which entails individually performing tests to detect and fix bugs in software applications. This method is particularly effective in the initial development phase, where writing automated test scripts may be more time - consuming or infeasible. It's also preferred for exploratory testing and in scenarios where UI testing focuses on the visual aspects of the application.

2.2 Advantages of Manual Testing

- Employs human intelligence for identifying technical errors.
- Enables focused testing on complex features and functions.
- Facilitates detection of non - code errors, like UI look and feel.
- Offers flexibility to emulate various user experience scenarios.
- Helps in identifying critical bugs that could render software untestable.

However, the scalability of manual testing is limited. It becomes less efficient with the growth of software projects, especially for repetitive tasks like regression testing. The manual approach, reliant on step - by - step human execution, is prone to errors and can lead to increased long - term costs.

2.3 Disadvantages of Manual Testing

- Consumes more time compared to automated methods.
- Higher susceptibility to human errors.
- Inadequate for extensive regression, load, and performance testing.
- Challenges in managing and executing numerous tests.
- Difficulty in accurately assessing UI elements like size and color combinations.

Given these limitations, there's a growing inclination towards automated testing techniques to enhance efficiency and accuracy in software testing.

2.4 Automated Software Testing: Characteristics and Limitations

Automated testing is a process in which expected software is compared with the developed software. In automated testing, tests are executed automatically via test automation frameworks, along with a few tools and pre - developed software such as Selenium, Katalan Studio, Unified Functional Testing (UFT), Appium, and Cucumber. Automation testing is a process in which testers utilize tools and scripts to perform the testing. It is mainly used where repetitive tests or time - consuming tests need to be run, to do parallel testing, and undertake non - functional testing like load, performance, and stress testing, to overcome human - made errors. There are both pros and cons of automated testing techniques.

2.5 Advantages of Automated Testing Techniques:

- Consistent reliability through uniform operations.
- Long - term cost - effectiveness.
- Reusability of automated test scripts.
- Versatility in application, especially for regression testing.
- Reduced need for human intervention, allowing automatic test execution.

Automated testing significantly lowers time and cost in software development, directing manual efforts more effectively. It's transformative across various business sizes, driving competitive advantages. However, challenges persist:

2.6 Disadvantages of Automated Testing Techniques:

- High initial setup costs due to expensive tools.
- Limited applicability to exploratory testing.
- Necessity for programming expertise.
- Limitations in assessing certain UI elements and dynamic content.
- Need for meticulous maintenance.

Businesses can fail to identify the right areas to automate to attain the best possible results. Moreover, automated testing can prove to be advantageous as long as it is developed and executed properly. In the current business environment, the testing automation benefits far outweigh the disadvantages, thus making the software development process more efficient.

3. Testing Frameworks and Tools: Necessity and Evolution

In today's world, software permeates many aspects of life, but human error can lead to software defects. These errors, if not addressed early in development, can escalate in cost and impact. To mitigate these risks, developers are encouraged to employ effective testing methods, such as Machine Learning (ML) and Data Mining Algorithms, alongside the right tools and frameworks. The realm of automated testing is rapidly advancing, offering a variety of frameworks and tools, each distinct in its architectural design and methodology. This paper will explore key test automation tools and frameworks vital for deploying Machine Learning and Data Mining techniques in software testing [7].

3.1 Test Automation Frameworks: Structure and Varieties

Test automation frameworks provide a structured environment for automating software testing processes. These frameworks consist of various elements, such as physical models for test creation, methodologies for handling test data, mechanisms for storing test results, object repositories, and integration with external resources. The flexibility of these frameworks allows for efficient modification, editing, and deletion of test scripts, offering scalable solutions with minimal effort and time.

Developers utilize these frameworks to write structured code, testing different components of the application. Key advantages include enhanced maintainability of scripts, higher test component reuse rates, and overall improvements in test speed, efficiency, and accuracy. This approach reduces risks and maintenance costs associated with testing.

There are several distinct types of test automation frameworks, each characterized by its unique architecture and suited for specific testing needs:

1) Modular Frameworks: These divide the software into isolated modules for individual testing, followed by a combined approach for comprehensive testing. This method ensures modularity and thorough coverage.

2) Data - Driven Frameworks: In this type, test data is separated from the scripts and stored externally. This separation allows for repetitive and versatile testing without needing to alter the original scripts, thus enhancing the flexibility of the testing process.

3) Keyword - Driven Frameworks: These frameworks separate test data and logic, using keywords and objects stored externally. This approach provides independence from the specific automation tools used, allowing for a more versatile testing process.

4) Hybrid Frameworks: As a combination of different framework types, hybrid frameworks aim to maximize the benefits of each individual type. They are particularly suitable for agile and adaptable testing environments, offering a comprehensive approach to automation. The research and development in these frameworks have made them robust and versatile, capable of effectively testing a wide range of applications. The choice of framework should align with the project's specific needs, ensuring seamless integration with other testing tools and adaptability to software changes [5].

5) Relevant Automated Testing Tools: Integration and Application

An automation tool is a software that is built on frameworks. Automated testing offers access to several features; automated testing enhances and expands the tester's capacity to evaluate the application. Different testing automation technologies are backed by their unique testing approach. The Test Automation Framework differentiates the testing code from the raw code, produces logs, and offers a list of frequently used libraries of functions. These utilities that testers employ when the real code is being used in the background are called automation tools. These tools provide enhancement in the testing process.

Test Suite Prioritization does upgrade the testing process by Combinational Criteria. Combinatorial testing is the testing approach where multiple combinations of the imputed parameters are used to ensure that the product is without any bug and is able to handle different sets of combinations. The significant strategy behind such an experiment is to change the weblogs into the test suites applicable to the client meeting and further record it into an Extensible Markup Language (XML) design. The algorithms utilized for this approach are precisely focused by the inclusion because of combinatorial test suites. The significant upgrade in the testing system drives the testing process towards test automation, which refers to the utilization of specific programming to execute testing and examine genuine outcomes with the normal outcomes. The light - footed lifecycle is one more advancement in programming testing. It incorporates short and expedient test cycles as often as possible, adjusting prerequisites. Furthermore, Test Driven Development is a procedure that utilizes mechanized unit tests for driving the plan of programming and compelling the decoupling interaction of the conditions.

Every testing automation tool has some strengths and some weaknesses based on which they are used for different purposes. Therefore, before selecting the tool a detailed comparative analysis should be made by the software developer. One should consider the budget, application type, and skill sets required to use the tools. Few automation tools that can be used in developing parallel automation testing architectures are discussed [8]:

a) **Selenium**: Selenium is an eminent testing framework comprising various tools and plugins used for testing web applications compatible with browsers and platforms like Windows, Linux, and Mac. The tool assists testers to write tests using various programming languages like Java, PHP, C#, Python, Ruby, Perl, etc. This tool supports recording and playback features without any need to learn test scripting languages. Selenium is primarily used in open - source test automation because of its powerful capability in performance testing. It is a user community to stay aligned with software technology advancements. Selenium can be integrated with other automation tools and frameworks that can enhance the software capabilities. Selenium is an open - source platform and there is no need for licensing or maintenance fees.
b) **Katalon Studio**: It is an automated testing framework to implement a complete automated testing solution for Desktop, Mobile Applications, API, and Web. It is an open - source framework and there is no need to have advanced programming skills to use it. Katalon Studio integrates required frameworks and features for effective test creation and execution. However, there is only one programming language option available that is Java/Groovy which is one of the drawbacks of Katalon Studio.
c) **Unified Functional Testing (UFT):** UTF formally known as QTP (QuickTest Professional) is an automation testing tool used for functional and regression testing. UFT covers most of the functional automated testing requirements of Web, Desktop, and Mobile Applications. UFT supports VBScript (Visual Basic Script) to register the test processes, operate, and control the test runs.

A lot of enhancement is undertaken in automated testing models to tackle complex software testing challenges. One of the challenges is that automated testing tools at times ignore the parallel execution. A complex test suite generates a huge data set from iterative code commits and multiple test runs. The sequential execution of tests abruptly stops the test cases due to queue timeout issues. Resultantly, the speed of detecting the regression bugs depletes during code integration and hence compromises the quality of the test queue in the designated test automation framework. To overcome this challenge Machine Learning Approach is implemented for better performance on test - case prioritization tasks. Machine learning will allow for the parallel execution of multiple tests in different environments at the same time.

6) Implementing Machine Learning in Automated Testing: Methods and Algorithms.

Machine learning is one of the techniques in automated testing that helps the programmer to get an early indication of the test runs having a higher probability of failing. This will help them to prioritize their work accordingly and thus increase the speed of the testing process. In addition, by prioritizing the execution of such test programs, the actual defects are found earlier in the Agile iteration [1]. To help run a test, Data Mining is used which is the collection and extraction of any unrecognized information or patterns that can potentially help run the test. This technique uses various statistical methods like developing statistical models and automation architectures. This helps capture the visible and usable characteristics evidenced in the given data [2]. Machine learning is more recommended as it strives to improve the overall performance and deduce effective and accurate predictions.

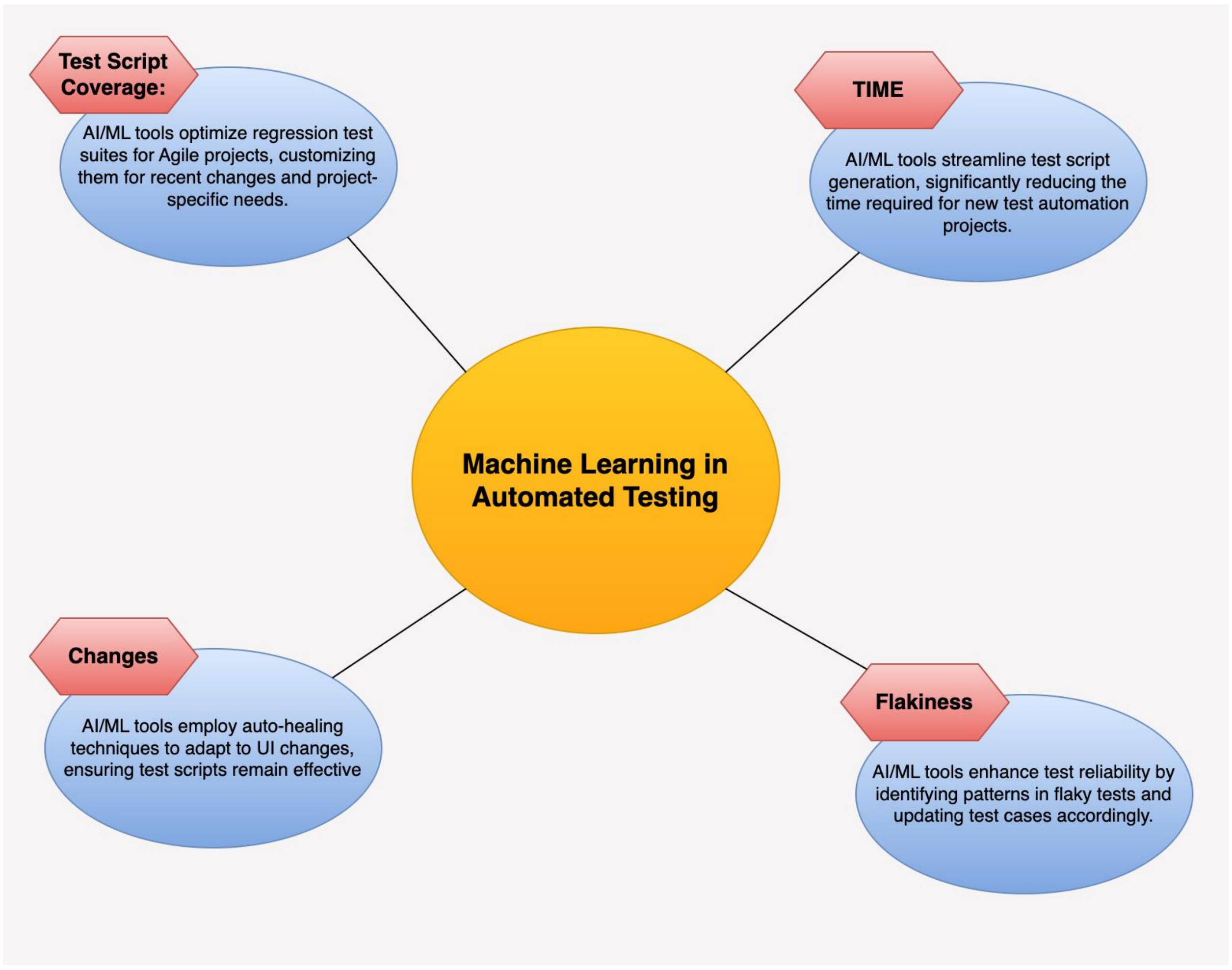
Test Script Coverage:
AI/ML tools optimize regression test suites for Agile projects, customizing them for recent changes and project-specific needs.
TIME
AI/ML tools streamline test script generation, significantly reducing the time required for new test automation projects.
Machine Learning in Automated Testing
Changes
AI/ML tools employ auto-healing techniques to adapt to UI changes, ensuring test scripts remain effective
Flakiness
AI/ML tools enhance test reliability by identifying patterns in flaky tests and updating test cases accordingly.

Figure 1: Machine Learning in Automated Testing

Few Machine learning techniques are being implemented which are not completely up to the mark. However, with more research, usability can be increased for automated test tools. According to the research published in IEEE by D. Talby and G. Fraser, they have already developed a machine learning algorithm where the method will generate the assertion statements and even summarize current behavior using a hybrid approach [3]. Using these assertions, developers can detect changes in the current situation of the project and help them identify future defects that may break other functionality. A lot of research is undertaken for Designing Regression Tests and evaluating the correctness of the software. Researchers have made several observations on testing methods to help programmers compare different automated testing methods and tools. Machine learning automated techniques are used to determine which Java source files are likely to have loopholes.

Based on the research conducted, the automated tests take four to five hours to run completed end - to - end integration tests that examine a complete flow of the data. Also, the tests run in alphanumeric order in which the newest test cases are executed at the last. Hence, it becomes essential to prioritize which test is more important or which test is more likely to have defects.

In the history of software development, various software metrics have been developed. Chidamber and Kenmere published a report to put forward new metrics to measure object - oriented design [9]. They defined six metrics based on measurement theory and reflected the viewpoints of experienced object - oriented software developers.

WMC	Weighted Methods per Class
DIT	Depth of Inheritance Tree
NOC	Number of Children
CBO	Coupling Between Objects
RFC	Response For a Class
LCOM	Lack of Cohesion in Methods

Chidamber - Kenmere metrics

This research was to prioritize testing of existing Java applications. The research was specifically about testing the existing codes. However, the majority of the Chidamber - Kemere metrics were ignored except WMC as at the time of testing, the design was finalized in the existing software. The WMC metric was used in calculating some class values and some average values per method. To calculate the collected metrics, a tool was developed. JavaParser was implemented to do the actual parsing of Java source code. An additional tool was developed to retrieve the Subversion (Source Control) data of each file. Using this research, defect fixes are flagged by entering specific code words in the subversion commit message. By retrieving the Subversion logs for each source file, it was possible to find which source files had defects at any point in time using the keywords Yes/No. The algorithm had a general format where 9 metrics value was separated by a comma:

Filename, value1, value2, …value9, Yes/No
In the given machine learning approach, the historical messages and values from the source code control system are accessed to uncover the defects in previous source code. From these source inputs, a data set is created containing the metrics and number of erroneous faults in the source file. This data is then sent to Weka which provides a wide range of tools that can be used to analyze and visualize the data. Weka is an interactive tool and a working bench of machine learning that creates a decision tree indicating the possible defects.

7) Goal of this Project
The paper intends to highlight how the automated test - generating methods would contribute to developing any software as compared to the single - handed dependency on manual testing. In the software industry, 50% of the project development cost is utilized for testing purposes. Nonetheless, the primitive way of testing software is manual. The Manual testing used by developers is not the most efficient testing procedure. It consumes time, and cost, and is vulnerable to erroneous outcomes. While automated testing tools can help subside these drawbacks. Evenly, while doing software testing, both manual testing and automated testing persuades us to consider our requirements of tools, costs, and expected benefits in the long run. Manual testing helps dig deep into the project enabling us to explore all the perspectives of the test. Whereas automated testing helps accomplish several tests in a short period. Thus, the true value of manual testing, automated testing, test automation tools, and frameworks are obtained when the right testing methodology and techniques are implemented in the right environment.

With the growth in the project, the number of test runs also increases. At times, even automated testing can take hours to run. As the test suits get longer, a technique is required where the tests which are more likely to fail are displayed in the front of the queue. The paper focused on machine learning techniques to discover the characteristics of a Java source file that indicates potential defects in the given source file. Using different sets of metrics, the subversion commit entries were accessed to find the historical defects. The extracted data was then analyzed using Weka's 148 decision tree [10]. Our main project justification states that by implementing machine learning techniques in automated testing, the number of test cases will be reduced based on the change in the software files and control flow, thus, improving the performance of software deployment.

4. Conclusion

Software automation testing processes are gaining more importance as it is more time efficient and reduces the frequency of human errors. According to the World Quality Report 2018 - 19, test automation is the most crucial part to deliver "Quality at Speed." The primary goal of this project is to understand the different types of testing and their advantages and disadvantages. We strived to create an understanding of when to use different testing techniques in different scenarios. Furthermore, we discussed various frameworks and tools available in automation testing methodologies and explained their importance in attaining successful test automation results. We stated the problems faced by the software team with the projected growth and provided them with effective solutions to overcome potential issues using machine learning algorithms and processes like data mining. There are several recommendations for future research studies such as employing these automation testing methods on a large scale and experimenting with their usage in various fields.

References

[1] K. Sneha and G. M. Malle, "Research on software testing techniques and software automation testing tools, " in Proc.2017 Int. Conf. Energy, Commun., Data Anal. Soft Comput. (ICECDS), 2017, pp.77 - 81. doi: 10.1109/ICECDS.2017.8389562.
[2] M. R. Blackburn, R. D. Busser, and J. S. Fontaine, "Automatic Generation of Test Vectors for SCR - Style Specifications, " in Proc.12th Annu. Conf. Comput. Assurance, Gaithersburg, MD, Jun.1997.
[3] L. Breiman, J. H. Friedman, R. A. Olshen, and P. J. Stone, Classification and Regression Trees. Wadsworth, 1984.
[4] G. Fraser and A. Arcuri, "Evosuite: automatic test suite generation for object - oriented software, " in Proc.19th ACM SIGSOFT Symp. and the 13th Eur. Conf. Found. Softw. Eng., 2011, pp.416–419.
[5] C. Pacheco and M. D. Ernst, "Randoop: feedback - directed random testing for java, " in Companion to the 22nd ACM SIGPLAN Conf. Object - oriented Programming Syst. Appl. Companion, 2007, pp.815–816.
[6] S. Berner, "About the Development of a Point of Sale System: an Experience Report, " in Proc. ICSE 2003, Portland, OR, May 2003.
[7] F. P. Brooks, "No silver bullet – essence and accidents of software engineering, " Comput., vol.20, no.4, Apr.1987.
[8] Q. Yang, J. J. Li, and D. M. Weiss, "A Survey of Coverage - Based Testing Tools, " Comput. J., vol.52, no.5, pp.589 - 597, Aug.2009. doi: 10.1093/comjnl/bxm021.
[9] M. A. Jamil, M. Arif, N. S. A. Abubakar, and A. Ahmad, "Software Testing Techniques: A Literature Review, " in Proc.2016 6th Int. Conf. Inf. Commun. Technol. Muslim World (ICT4M), 2016, pp.177 - 182. doi: 10.1109/ICT4M.2016.045.
[10] D. Talby et al., "Agile software testing in a large - scale project, " IEEE Software, vol.23, no.4, pp.30 - 37, 2006.
[11] G. Fraser and A. Arcuri, "Evosuite: automatic test suite generation for object - oriented software, " in Proc.19th ACM SIGSOFT Symp. and the 13th Eur. Conf. Found. Softw. Eng., 2011.
[12] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object - oriented design, " IEEE Trans. Software Eng., vol.20, no.6, pp.476 - 493, 1994. Available: https: //ieeexplore. ieee. org/abstract/document/295895/