International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Enhancing Testing Efficiency through the Implementation of an Optimal Test Automation Framework Selection Model

Geetika Singh[1], Jitendra Choudhary[2], Lokesh Kumar Laddhani[3]

Submitted: 26/04/2023 **Revised**: 25/06/2023 **Accepted**: 05/07/2023

Abstract: Selecting the best automation testing framework from a diverse range of approaches poses a significant challenge, as the available selection schemes fail to produce substantial results when multiple testing scenarios with varying functional requirements are present. In this paper, the authors present the Quality Assurance (QA) aware algorithm and an Optimal Test Automation Framework Selection (OTAFS) model, which consider QA parameters for selecting the optimal automation testing framework. The reliability, throughput and execution time of the selected framework are identified as the most efficient parameters. The study discusses the results of QA parameter values for the selected automation frameworks, which are implemented using Python and Selenium platform is used to create test cases. The proposed model and algorithm is experimentally evaluated on an e-commerce website, and a comparative analysis of the results is provided with QA parameter values of different automation testing frameworks.

Keywords: *Quality Assurance, Test Automation Framework, Automation Tools, Test Cases, Test Suite.*

1. Introduction

The rapid adoption of digital technologies is causing significant disruptions and digitization, fundamentally changing the world. In this new landscape, speed plays a crucial role in all IT operations, necessitating a paradigm shift in quality assurance (QA) practices [1]. The ability to deliver high-quality products at a faster pace has become the primary focus of digital assurance and organization want to deliver quality products much faster than ever before. This is making QA teams to bank on test automation. Various advancements have evolved in the area of automation testing. However, with advancements in automation testing, it is essential for organizations to choose the right automation framework [2], as it is a critical factor for success. Since, each project is unique, presenting its own set of challenges, duration, and tool requirements. To achieve agility in their business processes, organizations need a robust test automation solution that ensures superior software quality. Therefore, successful test automation frameworks for digital assurance not only prioritize the functional aspects of the project but also consider QA attributes.

1Research Scholar, Department of Computer Science, Medi-Caps University, Indore Madhya Pradesh,

geetika.singh3@gmail.com
2Associate Professor, Department of Computer Science, Medi-Caps University, Indore Madhya Pradesh,
jitendra.choudhary@medicaps.ac.in
3Assistant Lecturer, Institute of Computer Science, Vikram University, Ujjain, Madhya Pradesh,
lokesh.laddhani@gmail.com

QA is an additional aspect that encompasses non-functional properties, ensuring that automated tests accurately and reliably verify the expected functionality of the software application [3],[4]. However, if QA parameters are not carefully defined, automated tests may produce false positives or false negatives, resulting in inaccurate test results and undermining the credibility of the testing process. The effectiveness of test automation frameworks heavily relies on evaluating QA parameters. Several important QA parameters, such as reliability, throughput, test report results, execution time, and portability, play a significant role in ensuring the quality and performance of automated testing procedures. This paper aims to explore how different QA parameters impact the selection of the right framework, ultimately helping organizations achieve their digital assurance goals. The proposed model will recommend an automation framework that leads to smarter automation, better overall results, productivity benefits, and cost-effectiveness in the dynamic digital landscape. The paper is organized as follows: Section 2 describes some available testing framework selection approaches. section 3 gives an overview design of proposed system architecture and defines mechanism. Section 4 describes the proposed algorithm for QA evaluation. Section 5 provides the details of implementation, experimentation and discussion. Section 6 describes the conclusion and future scope.

2. Related Work

Some approaches and algorithms have been proposed for the dynamic selection of test automation framework. There are selection scheme of testing framework that considers different parameters such as functionality, technology, data environment, cost and scalability.

An existing literature evaluated testing tools focusing on identifying the quality parameters that aid to achieve usability, correctness and robustness in the application under test and provides support for quality assurance [5]. Xu.D et al. [6] have presented an automated test generation technique Model-based Integration and System Test Automation (MISTA) for integrated functional and security testing. It generate executable test with respect to a variety of coverage criteria of test models to enhance quality.

Brochi et al. [7] compared one open source testing tool with a commercial tool on the basis of quality assurance metrics like testability, learnability and supportability and deduced that Selenium is better. Huang et al. [8] proposed an efficient service selection scheme to help service requester choose Web services by considering non functional characteristics. They presented QOS model of Web service, it consider different data types and uses Multiple Criteria Decision Making (MCDM) technique to help service requester evaluate service numerically. It facilitates dynamic adaptable composition.

Dobslaw F. et al. [9] presented a manual replay method for ROI estimation for the Automated GUI Testing (AGT) frameworks and compared two fundamentally different AGT frameworks, namely Selenium and EyeAutomate and investigate difference between the two testing framework. ROI is also compared to manual testing, and here defect finding capabilities and usability are also reported. Miranda B. et al. [10] proposed a FAST technique that provides scalable similarity based test case prioritization in both white box and black box testing. The simulation study of scalability showed that FAST technique can prioritize a million test cases in less than 20 minutes. It helped to scale up the industry demand.

Winkler et al. [11] presents a flexible Test Automation Framework (TAF) based on Behavior-Driven Testing, which enables continuous integration and testing for control code variants in Production Systems Engineering (PSE). The framework uses Abstract Syntax Tree (AST) for human-based verification and validation and is evaluated using an Industry 4.0 Testbed. The results show that the TAF concept supports flexible configurations of testing tool chains and can support human-based verification and validation of control code variants. Elberzhager et al. [12] shows the systematic mapping study aims to identify existing approaches that can reduce testing effort and provide an overview for researchers and practitioners. The study found an increased interest in this topic in recent years, with automation and prediction approaches receiving the most attention. Some input reduction approaches were also identified, but only a small number of approaches combine early quality assurance activities with testing to reduce test effort. The study highlights the need for future research in this area to address the ongoing challenge of reducing test effort

Borg et al. [13] This paper discusses the use of AI-enabled conversational agents for language practice, specifically in the context of virtual job interviews. The authors present ongoing action research aimed at quality assurance of generative dialog models and describe a set of requirements and corresponding automated test cases. Results show that some test case designs can detect meaningful differences between candidate models. The paper offers initial steps towards an automated framework for machine learning model selection in the context of conversational agents, with future work focusing on model selection in an MLOps setting. Yu et al. [14] presents SimRT, an automated regression testing framework for detecting data races introduced by code modifications in concurrent programs. SimRT uses a regression test selection technique to reduce the number of test cases that must be run and a test case prioritization technique to improve the detection rate of such races. The empirical study of SimRT shows that it is more efficient and effective for detecting races than other approaches, and both its test selection and prioritization components contribute to its performance

Shahamiriet al.[15] proposes an automated test oracle framework to address output-domain generation, input domain to output domain mapping, and comparator challenges. The proposed approach uses I/O Relationship Analysis to generate the output domain automatically and Multi-Networks Oracles based on artificial neural networks to handle the second challenge. The last challenge is addressed using an automated comparator. The proposed approach was evaluated using an industry strength case study, which was injected with faults, and the results showed that the proposed approach automated the oracle generation process 97% in this experiment. The accuracy of the proposed oracle was up to 98.26%, and the oracle detected up to 97.7% of the injected faults.

Salariet al.[16] addresses the problem of choosing a suitable test automation framework for testing software on Programmable Logic Controllers (PLCs) used in industrial control systems. The authors focus on the popular COntrollerDEvelopmentSYStem (CODESYS) development environment and explore its supported test automation frameworks. They identify 29 different criteria for evaluating these frameworks, validate them with an industry practitioner, and compare the resulting frameworks in an industrial case study. The study shows that CODESYS Test Manager and CoUnit are the most frequently mentioned frameworks in the grey literature review results. The paper aims to increase knowledge in automated testing of PLCs and assist researchers and practitioners in selecting the right framework for test automation in an industrial context.

Lukasczyk et al. [17] Pynguin is a test-generation framework designed for dynamically-typed programming languages like Python. The tool aims to reduce the effort of writing tests manually by generating regression tests with high code coverage. Pynguin is designed to be easily usable by practitioners and can be extended to adapt to researchers' needs for future research. The paper highlights the need for tools and research on test generation for dynamically-typed languages, as they have gained significant popularity over the last decade. The authors provide a demo of Pynguin and make the tool, documentation, and source code available on their website. Overall, the paper presents a valuable contribution to the field of test-generation, addressing the gap in research and practice for dynamically-typed languages like Python.

Lukasczyket al.[18] This paper extends previous work on PYNGUIN to support more aspects of the Python language and evaluates various state-of-the-art test-generation algorithms. The results demonstrate that evolutionary algorithms can outperform random test generation in the context of Python and that DynaMOSA yields the highest coverage results, but fundamental issues remain, such as inferring type information for code without this information. Regression assertions were also generated and evaluated in the improved PYNGUIN tool.

Thörnet al.[19] This case study explores how strategies from safety-critical development can enhance quality assurance for a test framework in an agile, non-safety development context. The study identifies candidate solutions to quality assurance and divides them into four aspects. The importance of perceiving a test framework as a tool-chain, with sub-tools analyzed for applicable measures, is emphasized. Additionally, sub-tools can be classified on an individual basis and confidence argued as the sum of applied measures throughout the framework. The study offers insights for improving the quality of embedded systems through test framework risk mitigation.

Prasetya et al. [20] proposes an agent-based approach for robust automated testing in modern computer games. The proposed approach is based on the reasoning type of AI to address challenges such as huge interaction spaces, non-deterministic environments, and changing layouts and game logic. The approach is designed to maintain test robustness, which is lacking in existing game testing approaches. The paper presents a case study to validate the proposed approach and demonstrates its effectiveness in improving the reliability and efficiency of game development through automated testing.

The literature has highlighted several concerns that need to be addressed, including the absence of experimentation on an actual application, deficient quality assurance attributes, an unreliable environment, and the imperative to enhance efficiency while reducing complexity. To overcome these issues, a solution has been proposed that encompasses the identified challenges. The approach involves incorporating various quality assurance (QA) attributes, such as reliability, throughput, portability, execution time, and cost, to evaluate and compare frameworks effectively. The proposed model has been successfully tested on an e-commerce website and can be customized to meet the unique requirements of different projects and testing environments. Moreover, the solution is adaptable, scalable, and versatile.

3. Proposed Optimal Test Automation Framework Selection Model

We have considered an e-commerce website to select automation testing framework. This model selects a

framework among existing frameworks for test case execution. An optimal framework is selected for different number of test cases with respect to the computed values of QA parameters. Following subsections describes the overview of adopted methodology and gives the system architecture of the proposed model.

3.1 An Overview of Proposed Model

Choosing an appropriate automation framework is a critical aspect of ensuring that it can seamlessly incorporate various automation testing technologies and keep up with changes in the application being tested. Such a framework can be leveraged across projects within an organization and yield significant return on investment [21]. However, selecting the right framework for the entire testing process is a complex issue [22], and requires effective integration of different Quality Assurance (QA) parameters. The Optimal Test Automation Framework Selection (OTAFS) model provides an improved selection approach and enhances the performance of the testing process. This model utilizes a filtration strategy that drives the search based on the computed QA parameters of the automation framework, which enables better decision making during the selection process.

3.2 System Architecture

Fig. 1 presents the system architecture of the Optimal

Test Automation Framework Selection (OTAFS) model. The model is the combination of different modules such as Input module, Framework Repository, Evaluation Engine, Ranking Module, Output Module and Integration Module.

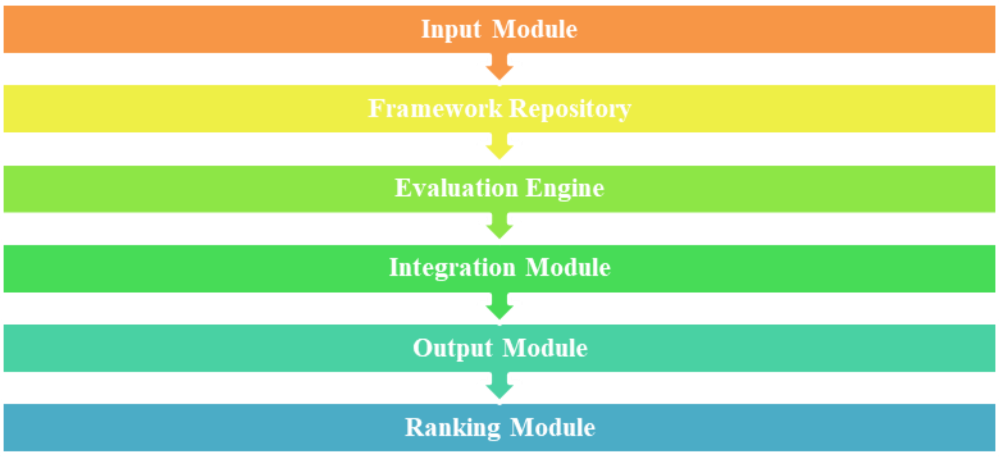

Fig. 1 Optimal Test Automation Framework Selection (OTAFS) Model

A detailed description of each module is as follows:

Input Module: The Input Module helps in enabling users to furnish input data for the tests that are being executed. This input may take the form of user inputs, data procured from databases or files. Typically, this module offers users a structured approach to specifying input data, such as through an XML file or via a programming interface like an API or a command line interface. Furthermore, it offers a mechanism to seamlessly read input data and transmit it to the testing framework for execution. In the context of an e-commerce website, this module helps in enabling users to provide input data that reflects the real-world scenarios and user behaviors that the website is designed to handle. This input data may include product information, order details, customer information, and other relevant data that is essential to the functioning of the website. The Input Module enables the testing framework to simulate these real-world scenarios, helping to identify any potential issues or errors that may arise in the system. This module facilitates the collection of user input to drive testing processes, thereby enabling the provision of optimal user experiences.

Framework Repository: The Framework Repository represents a centralized storehouse of crucial information pertaining to the array of available testing frameworks. This repository encapsulates critical details concerning framework features and their compatibility with various platforms and programming languages. Developers can leverage this repository to readily access and evaluate diverse testing frameworks, and compare their features and capabilities through specific testing requirements. This helps to ensure that the testing process is optimized for maximum efficiency and effectiveness, leading to better overall project outcomes.

In our repository, we have identified four popular testing frameworks, including Keyword-Driven, Data-Driven, Test Data Driven and Hybrid frameworks. Each of these frameworks has its own unique features and capabilities, making them suitable for different types of testing scenarios. The Framework Repository provides a detailed overview of these frameworks, enabling developers to evaluate their suitability for their specific testing needs.

Evaluation Engine: The Evaluation Engine compares and evaluates diverse testing frameworks based on Quality Assurance (QA) parameters specified with the help of the proposed algorithm. This engine scrutinizes each framework based on the QA criteria, such as reliability, throughput, execution time, cost, and portability. It assigns scores to each testing framework against the predefined QA criteria to determine how well each framework aligns with the needs of the project. The Evaluation Engine serves as a powerful tool for facilitating informed decision-making by project stakeholders regarding the most suitable testing framework for the project at hand. It enables project teams to choose the optimal testing framework, which can have a significant impact on the effectiveness and efficiency of the testing process.

*Ranking Module:*This module takes into consideration the diverse QA attributes evaluated by the Evaluation Engine, such as reliability, throughput, execution time, cost, and portability. Based on this assessment, the Ranking Module generates a comprehensive ranking that outlines the strengths and weaknesses of each testing framework. The ranking provides project stakeholders with an insightful and data-driven perspective on the suitability of each framework. It serves as a valuable phase for making informed decisions about the most appropriate testing framework to use in a given project, based on the values of QA parameters.

Output Module: The Output Module is a key component that provides users with critical information about the results of the evaluation process, including the ranking of the testing frameworks and individual scores for each criterion. This module generates a comprehensive report that details the tests that passed, failed, or were skipped, along with additional information such as the duration of each test, any error messages that were generated, and a summary of the overall test results. The Output Module plays an essential role in facilitating effective decision-making by project stakeholders, who can use the report to gain valuable insights into the strengths and weaknesses of the testing frameworks under consideration. Additionally, the Output Module allows for the export of results, enabling further analysis and sharing with other relevant stakeholders. This feature empowers project teams to identify areas for improvement, make necessary adjustments, and optimize the testing process for better outcomes.

Integration Module: The Integration Module enables seamless integration with diverse testing tools such as Test Automation and Management tools. This integration ensures that the selected testing framework can be readily integrated with the existing testing infrastructure, thereby enhancing the overall efficiency and effectiveness of the testing process. The Integration Module also tests the integration between different components or modules of the system, enabling project teams to assess how different parts of the system work together to achieve the intended functionality. Through this testing, the Integration Module helps to identify any potential issues or inconsistencies that may arise in the system, allowing for timely corrective action to be taken.

4. Proposed QA Evaluation Algorithm for OTAFS Model

This section describes the algorithm. The algorithm computes [23] the values of QA parameters which aids in finding the optimal test automation framework.

Let us assume T is the list of test suites for which QA parameters needs to be evaluated for different test automation frameworks. The testing tool is selected according to the application and testers preferences. In our study we have selected Selenium [24] as the testing tool for the execution of test cases. This algorithm will help in finding the optimal testing framework according to the values of QA parameters evaluated. This algorithm will aid in searching and optimizing the appropriate testing framework by the OTAFS model.

4.1 Optimal Automation testing framework selection algorithm:

```
Input: List of Test cases t = {t1, t2,t3,……,tn} for each test suite T, testing tool according to application and practitioner’s preference
Output: Optimum Test Automation Framework selected for each test suit T.
// Find the testing tool.
/* We have selected Selenium for execution of test cases in our previous study*/
For each test suite T do
{
 Fort each testsuiteurl do
{
Result[] = invokerbean( select testing tool, testsuiteurl)
// QA calculation phase
Calculation of Reliability
Calculation of Probability
Calculation of Execution Time
Calculation of Throughput
Calculation of Cost
// Normalization phase
// Normalize the resultant QA
/* Select the Test Automation Framework which is optimum for each Test suite T*/
}
}
```

5. Implementation and Experimental Evaluation

This section constitutes the description of tools and technologies used to implement the proposed approach. Further, experiment has been performed to evaluate the proposed OTAFS approach based on different QA parameters. Next, the evaluation of computational values of QA parameters is analyzed and ranked.

5.1 Implementation Details

Optimal Test Automation Framework Selection (OTAFS) is implemented on Python Platform. A demo website (http://www.saucedemo.com) and four frameworks are used to test the proposed approach over the dataset of varying test cases i.e. 48 and 98. These test cases are executed to evaluate the value of QA parameters. The approach is programmed on following specification -Packages {"pluggy": "1.0.0", "pytest": "7.0.1"}, Platform macOS-11.6-x86_64-i386-64bit, Plugins {"html": "3.2.0", "metadata": "2.0.4"}, Python 3.9.15. Table -1 and 2 shows the performance measures based different QA criteria.

5.2 Performance Metrics

The proposed model has been evaluated in terms of Quality Assurance (QA) parameters for selecting a Test Automation Framework. These attributes [23] are significant for successful implementation of Test Automation Framework. The mathematical representation and description of the metrics chosen are as follows:

Reliability: It is measured using a metric called Rel (F), which represents the probability that a test is correctly executed by the testing framework and that the framework handles all the exceptions within an expected time period.

$$Rel(F) = P(F) / T(F)$$

where, P(F) represents the probability that the testing framework will execute a test correctly without encountering any errors or exceptions. T (F) represents the expected time period within which the framework should handle any exceptions or errors that occur during the test execution process.

Execution Time: The test execution time reporting metric represents the number of successful test cases executed by the framework for a particular amount of invocation. This metric is denoted by Ext.

$$Ext = Nsuc(F)$$

where, Nsuc (F) represents the total number of successful test cases executed by the testing framework for a particular amount of invocation.

Throughput: It is calculated as the ratio of the total number of test data generated by the framework to the time it takes to execute them.

$$Ttp(F) = Ntd(F) / Tinv(F)$$

where,Ntd(F) represents the total number of test data generated by the framework and Tinv(F) is the total time taken to execute these test data.

Probability: The portability metric for a test framework is determined by the number of handled requests from various team members who use different platforms, such as Windows, Mac, or Linux, in a specific time period. A framework that can be easily ported across multiple platforms is highly desirable as it allows for a wider user base and can save time and resources.

The portability P(F) of a test framework is a qualitative parameter. A higher portability score indicates that the framework is easily portable and can function efficiently on different platforms. A low portability score may indicate that the framework may require additional resources and time to be deployed on different platforms, which can impact the testing process and the overall project timeline.

Execution Cost: The cost of executing a test automation framework refers to the financial expenses incurred in implementing the framework. The cost may vary depending on the type of framework utilized or developed. It may involve the cost of tools, licenses, hardware, and personnel. The cost can be measured by assessing the expenses required to maintain and operate the framework over a specified time period. It is crucial to consider the cost of implementing and maintaining a framework to ensure that the return on investment is worthwhile.

The cost function can be represented as Tcost(F).

5.3 Performance Measure

The given situation entails a comprehensive end-to-end testing of an e-commerce website, which involves thorough testing of every component, including input boxes, buttons, and other elements, to ensure their desired functionality. To achieve this, we have devised four distinct testing frameworks, namely Keyword Driven, Data Driven, Test Driven, and Hybrid, each utilizing the Python language and tailored to evaluate different aspects of the website's performance.

To ensure seamless execution of the test cases across these testing frameworks, we initially crafted a set of different number of test cases for each suite, aiming to encompass the entire website's functionality and logically organize them for easy management and maintenance. The Selenium automation testing tool [24] was utilized to evaluate these test suites on various testing frameworks, each having a different set of tools and utilities for organizing and running the tests.

Upon completion of the testing process, the evaluation engine generated a comprehensive test report based on the QA attribute values evaluated by the proposed QA evaluation algorithm, providing a detailed assessment of the effectiveness of each testing framework. The ranking module was then employed to rank these frameworks based on their suitability to the project's overall requirements. The most effective testing framework was selected by comparing the scores of each framework, thereby providing a robust and scalable testing solution for the Application under Test (AUT). Following the selection of the most effective testing framework, the application underwent real testing to ensure its seamless functionality. The utilization of this comprehensive testing process provides an efficient and reliable solution for evaluating the website's performance and ensuring its smooth operation.

Table 1. Performance measure based on QA attributes where Test cases t1= 48

Quality Assurance Parameters	**Number of Test Cases = 48**			
	Keyword Driven	**Data Driven**	**Test Driven**	**Hybrid**
Reliability	40	30	40	49
Execution Time (in min)	7.266	8	8.66	4.33
Throughput	5.505	3.75	4.619	11.316
Portability	High	High	High	High

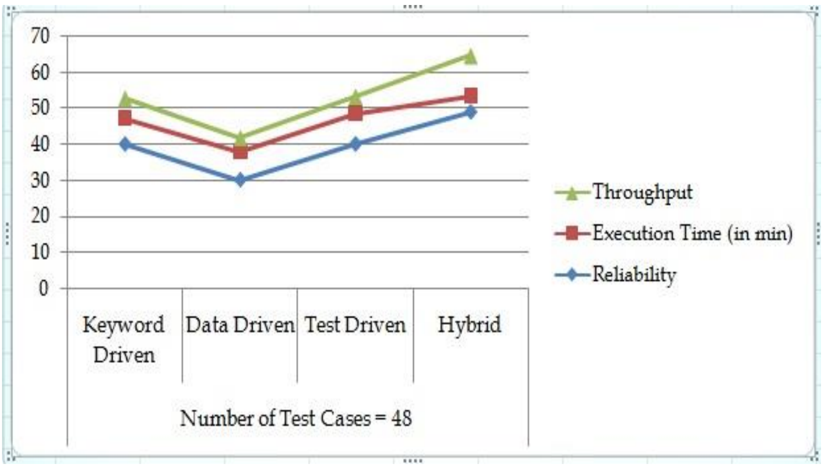

Fig 2. Graphical representation of QA attributes vs Testing Frameworks

Table 2. Performance measure based on QA attributes where Test cases t2 = 98

Quality Assurance Parameters	**Number of Test Cases = 98**			
	Keyword Driven	**Data Driven**	**Test Driven**	**Hybrid**
Reliability	80	62	85	97
Execution Time (in min)	14.533	16	17.333	8.667
Throughput	5.505	3.875	4.904	11.192
Portability	High	High	High	High

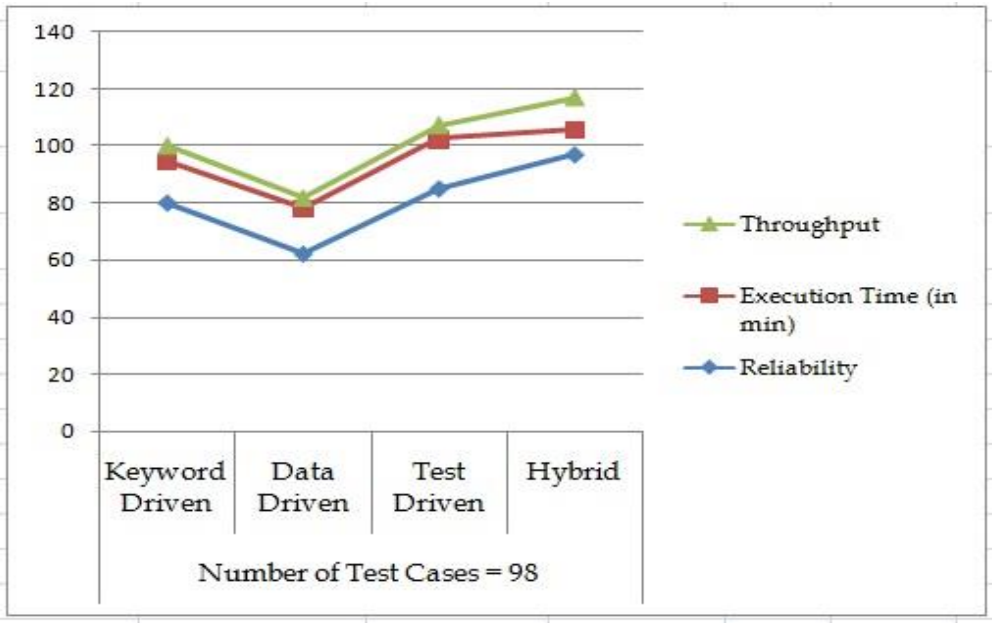

Fig 3. Graphical representation of QA attributes vs Testing Frameworks

In the evaluation process, we have identified reliability, throughput, and execution time as the critical parameters that significantly impact the automation framework's performance, especially with varying numbers of test cases. Hence, it is imperative to consider these parameters to ensure the quality and efficacy of the test automation framework.

To gain a better understanding of the performance of the different testing frameworks under varying numbers of test cases, we analyzed and compared their performance using a graphs represented in Fig. 2 & 3. The results from this analysis clearly indicate that the Hybrid framework outperforms the other frameworks based on the evaluated values of various QA parameters obtained from implementation the proposed algorithm.

These observations validate the importance of choosing a suitable testing framework that provides optimum reliability, high throughput, and efficient execution time, especially when the number of test cases varies. Therefore, it is crucial to consider these parameters while selecting a testing framework to ensure optimal performance and quality of the automation process.

6. Conclusion

The proposed algorithm and model offers a better solution to the challenge of selecting an appropriate automation testing framework. The Optimal Test Automation Framework Selection (OTAFS) model was implemented on an e-commerce website and its performance was evaluated using various Quality Assurance (QA) parameters, such as reliability, execution time, probability, throughput, and execution cost, across a dataset of test cases with the help of proposed algorithm. The results indicate that the proposed model is effective and efficient in determining the performance of existing testing frameworks. Moreover, the evaluation revealed that the Hybrid framework performed the best among the evaluated options. To further enhance this research, the OTAFS model could be expanded to include additional QA attributes and be evaluated on a broader range of automation testing frameworks. Overall, this study provides valuable insights into the effective selection of automation testing frameworks, and could greatly benefit organizations seeking to improve their testing processes.

References

[1] Tripathy, P., & Naik, K. (2011). Software testing and quality assurance: theory and practice. John Wiley & Sons.

[2] Amaricai, S., & Constantinescu, R. (2014). Designing a software test automation framework, Informatica Economica, 18(1), 152.

[3] Lin, Y. D., Rojas, J. F., Chu, E. T. H., & Lai, Y. C. (2014). On the accuracy, efficiency, and reusability of automated test oracles for android devices. IEEE Transactions on Software Engineering, 40(10), 957-970.

[4] Umar, M. A., &Zhanfang, C. (2019). A study of automated software testing: Automation tools and frameworks. International Journal of Computer Science Engineering (IJCSE), 6, 217-225.

[5] Anjum, H., Babar, M. I., Jehanzeb, M., Khan, M., Chaudhry, S., Sultana, S., ... & Bhatti, S. N. (2017). A comparative analysis of quality assurance of mobile applications using automated testing tools. International Journal of Advanced Computer Science and Applications, 8(7).

[6] Xu, D., Xu, W., Kent, M., Thomas, L., & Wang, L. (2014). An automated test generation technique for software quality assurance. IEEE transactions on reliability, 64(1), 247-268.

[7] Brohi, A. B., Butt, P. K., & Zhang, S. (2019). Software Quality Assurance: Tools and Techniques. In Security, Privacy, and Anonymity in Computation, Communication, and Storage: SpaCCS 2019 International Workshops, Atlanta, GA, USA, July 14–17, 2019, Proceedings 12 (pp. 283-291). Springer International Publishing.

[8] Huang, A. F., Lan, C. W., & Yang, S. J. (2009). An optimal QoS-based Web service selection scheme, Information Sciences, 179(19), 3309-3322.

[9] Dobslaw, F., Feldt, R., Michaëlsson, D., Haar, P., de Oliveira Neto, F. G., &Torkar, R. (2019, October). Estimating return on investment for gui test automation frameworks. In 2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE) (pp. 271-282). IEEE.

[10] Miranda, B., Cruciani, E., Verdecchia, R., &Bertolino, A. (2018, May). FAST approaches to scalable similarity-based test case prioritization. In Proceedings of the 40th International Conference on Software Engineering (pp. 222-232).

[11] Winkler, D., Meixner, K., & Novak, P. (2019). Efficient and flexible test automation in production systems engineering. Security and Quality in Cyber-Physical Systems Engineering: With Forewords by Robert M. Lee and Tom Gilb, 227-265.

[12] Elberzhager, F., Rosbach, A., Münch, J., & Eschbach, R. (2012). Reducing test effort: A systematic mapping study on existing approaches. Information and Software Technology, 54(10), 1092-1106.

[13] Borg, M., Bengtsson, J., Österling, H., Hagelborn, A., Gagner, I., & Tomaszewski, P. (2022, May). Quality assurance of generative dialog models in an evolving conversational agent used for Swedish language practice. In Proceedings of the 1st International Conference on AI Engineering: Software Engineering for AI (pp. 22-32).

[14] Yu, T., Srisa-an, W., &Rothermel, G. (2014, May). SimRT: An automated framework to support regression testing for data races. In Proceedings of the 36th International Conference on Software Engineering (pp. 48-59).

[15] Shahamiri, S. R., Kadir, W. M. N. W., Ibrahim, S., & Hashim, S. Z. M. (2011). An automated framework for software test oracle. Information and Software Technology, 53(7), 774-788.

[16] Salari, M. E., Enoiu, E. P., Afzal, W., &Seceleanu, C. (2022, April). Choosing a Test Automation Framework for Programmable Logic Controllers in CODESYS Development Environment. In 2022 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW) (pp. 277-284). IEEE.

[17] Lukasczyk, S., & Fraser, G. (2022, May). Pynguin: Automated unit test generation for python. In Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings (pp. 168-172).

[18] Lukasczyk, S., Kroiß, F., & Fraser, G. (2023). An empirical study of automated unit test generation for Python. Empirical Software Engineering, 28(2), 36.

[19] Thörn, J., Strandberg, P. E., Sundmark, D., & Afzal, W. (2022). Quality assuring the quality assurance tool: applying safety-critical concepts to test framework development. PeerJ Computer Science, 8, e1131.

[20] Shirzadehhajimahmood, S., Prasetya, I. S. W. B., Dignum, F., Dastani, M., & Keller, G. (2021, August). Using an agent-based approach for robust automated testing of computer games. In Proceedings of the 12th International Workshop on Automating TEST Case Design, Selection, and Evaluation (pp. 1-8).

[21] Graham, D. (2010). ROI of test automation: benefit and cost. Professionaltester. com, November, 2010.

[22] Shahin, M., Babar, M. A., & Zhu, L. (2017). Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices. IEEE access, 5, 3909-3943.

[23] Singh, G., Choudhary, J., Laddhani, L. (2023) : An Optimal Selection Scheme for Automation Testing Framework with Quality Assurance. Grenze International Journal of Engineering and Technology, Volume 9, No. 1,p. 2935-2940 https://thegrenze.com ISSN(Online): 2395-5295, ISSN(Print): 2395-5287.

[24] Singh, G., Choudhary, J., Laddhani, L. (2022): Taxonomic Analysis of DevOps Tools. JOURNAL OF ALGEBRAIC STATISTICS Volume 13, No. 3, p. 2725-2731 https://publishoa.com ISSN: 1309-3452.

[25] Paigude, S. ., Pangarkar, S. C. ., Hundekari, S. ., Mali, M. ., Wanjale, K. ., & Dongre, Y. . (2023). Potential of Artificial Intelligence in Boosting Employee Retention in the Human Resource Industry. International Journal on Recent and Innovation Trends in Computing and Communication, 11(3s), 01–10. https://doi.org/10.17762/ijritcc.v11i3s.6149

[26] Mr. Rahul Sharma. (2013). Modified Golomb-Rice Algorithm for Color Image Compression. International Journal of New Practices in Management and Engineering, 2(01), 17 - 21. Retrieved from http://ijnpme.org/index.php/IJNPME/article/view/13