An Analysis of The Impact of Test Case Prioritization on Software Quality

D. I. De Silva
Computer Science and Software Engineering
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
dilshan.i@sliit.lk

Anudi Wanigaratne
Computer Science and Software Engineering
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
anudi.w@sliit.lk

Athukorala.Y.J.
Faculty of Computing
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
it20029968@my.sliit.lk

Shaminda W.G.T.
Faculty of Computing
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
it20051020@my.sliit.lk

Jayawardhana E.H.K.
Faculty of Computing
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
it19192024@my.sliit.lk

Thennakoon S.U.
Faculty of Computing
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
it20147846@my.sliit.lk

***Abstract*— This article investigates how test case prioritization affects software quality and how it may be incorporated with various software development approaches such as agile or DevOps. In the framework of these procedures, we assessed the effectiveness of various prioritization methods and highlighted areas for improvement or optimization. In addition to functional requirements, we assessed the influence of test case prioritization on non-functional needs such as performance and security, offering a more thorough knowledge of the benefits and limitations of test case prioritization in various circumstances.**

Our observations suggest that prioritizing test cases can greatly enhance software quality, especially when combined with agile or DevOps techniques. Prioritizing non-functional needs has also been shown to increase overall program performance and security. Considering our findings, we encourage additional study to optimize and improve test case prioritization strategies, in addition to assessing their efficiency in various software development environments.

Keywords—prioritizing, software quality, non-functional, integration

I. INTRODUCTION

This emphasizes the importance of software testing in the development of software and introduces the idea of test case prioritization as a tactic to order the execution of test cases according to their importance. Software testing is an important aspect of the software development process since it ensures that the final product fulfills the required quality standards. Because of the expanding number of test cases that must be completed, testing durations and costs increase as software systems get broader and more complicated. To address this issue, test case prioritization is an approach that selects test cases based on their importance in order to improve the testing process's efficacy and efficiency. The purpose of this study is to investigate the effect of the test case prioritizing on software quality and to suggest a new prioritization technique to help software engineers make educated decisions about their testing process. The proposed method may also add to the existing literature on test case prioritization and improve software quality. According to the study, as software applications become more complicated, software testing becomes more critical, and test case prioritization can have a substantial impact on the quality and efficacy of the testing process. The study's main objectives are to investigate how test case priority affects software quality and to provide an alternative prioritization method. The research problem is to assess the efficacy of various test case prioritization approaches in various software development methodologies and to suggest a new prioritization methodology to increase the efficiency and efficacy of software testing procedures. The study's goal is to contribute to the current research on test case prioritization and software quality by offering insights that will help software engineers make informed decisions about which priority approach to utilize.

By assessing the influence of test case prioritization and suggesting a new prioritization technique, this research study intends to contribute to the existing literature on test case prioritization and software quality. The research findings can help software engineers and researchers in the field of software engineering by increasing the efficiency and effectiveness of the software testing process.

II. LITERATURE REVIEW

For several decades, software testing researchers have been working on test case prioritization. Many studies have been conducted to investigate the efficacy of various prioritization techniques and their impact on software quality. In this section, we will look at some of the relevant research studies in this field.

The coverage-based strategy is one of the most widely used prioritization techniques. This technique ranks test cases, according to their coverage of various program features such as statements, branching, and paths. One of the research projects discovered that coverage-based approaches can increase testing effectiveness by revealing defects that random testing misses [2]. And, another researcher discovered the effectiveness of several coverage-based strategies in the context of object-oriented software and discovered that branch coverage was the most successful technique [1].

The history-based technique, which ranks test cases based on their execution history, is another often used prioritization technique. In 1999, another researcher discovered that history-based strategies can increase testing effectiveness by discovering defects that coverage-based techniques overlook. And, the efficiency of several history-based strategies in the context of regression testing and discovered that techniques based on the frequency of past defects were the most effective [3].

Recent research has concentrated on the integration of test case prioritization with other software development approaches, such as agile and DevOps. In 2019, evaluated the efficiency of prioritization approaches in the context of agile development and discovered that user story-based techniques were the most effective [4].

Another study looked at the efficacy of prioritization approaches in the context of DevOps and discovered that techniques based on the frequency of code changes were the most effective.

Finally, studies have investigated the effect of test case priority on non-functional needs like performance and security. In 2017, study looked at the impact of prioritization approaches on performance needs and discovered that techniques based on the criticality of system operations were the most successful [5]. Another study looked at the influence of prioritization approaches on security requirements and discovered that techniques based on system component vulnerability were the most successful.

III. Methodology

The practice of introducing test case prioritization approaches into various software development methodologies such as Agile, Waterfall, and others is referred to as "integration with software development methodology." This entails examining the distinct features of each approach and determining how to adapt and integrate prioritizing techniques into their software testing process.

The assessment of software performance beyond functional requirements, such as usability, security, scalability, and stability, is referred to as non-functional requirements evaluation. The assessment in this context entails examining how test case prioritizing approaches affect the performance of software systems in terms of non-functional requirements and how they can be modified to improve overall software quality.

A. Integration with software development methodologies

Prioritization of test cases is an important part of software testing that may be combined with various software development processes to improve the quality of the software under development. In the context of these procedures, the effectiveness of various prioritization **techniques can be examined in order to discover areas for** improvement or optimization. A research study could take the following steps to evaluate the performance of different prioritization approaches in the context of these methodologies and find areas for improvement or optimization:

- **Identify the software development methodology**: The first step is to determine the software development methodology that will be used for the study. Different methodologies may necessitate different prioritization techniques, and understanding the specific characteristics of each methodology is critical in order to select the most effective prioritization technique.
- **Define the prioritization criteria**: The next step is to define the test case prioritization criteria based on the software development process. For example, priority criteria in Agile techniques may be based on user stories, whereas priority criteria in Waterfall methodologies may be based on project milestones. These criteria must be quantifiable and well-defined.
- **Assign weights to the criteria:** As explained in previous response, the weights for the prioritizing criteria can be assigned using a formula such as AHP. The weights should reflect the relevance of each criterion in relation to the others and take into consideration the software development methodology's specific qualities.
- **Calculate the priority score:** Following the assignment of weights, the priority score for each test case can be calculated using the formula described in the previous answer. The priority score can then be used to rank test cases before they are tested.

The priority score can be calculated as follow:

$$Priority\ score = (W \times R \times F) / (T \times C)$$

where:
W is the weight of the test case (e.g., its importance to the overall functionality of the program)
R is the risk associated with the test case (e.g., the severity of the bug that it could potentially uncover)
F is the frequency of the test case (e.g., how often it needs to be run)
T is the time required to execute the test case
C is the cost of executing the test case (e.g., the resources required to run it)

Assume we have a program with 100 test cases that we wish to prioritize using this equation. For one of the test cases, here is an example calculation:

Test case #25 has a weight of 8, a risk factor of 9, a frequency of 3, a time requirement of 2 hours, and a cost of $100.

Priority score would be: (8 x 9 x 3) / (2 x 100) = 10.8

We can repeat this calculation for all 100 test cases to generate a prioritized list. The higher the priority score, the higher the priority of the test case.

- **Evaluate the effectiveness of the prioritization technique:** The efficiency of the prioritization

technique can be assessed by comparing the results to a baseline, such as a random order of test cases or a specified order of test cases. The comparison can be based on the number of faults discovered, testing time, or other relevant metrics.

- **Identify areas for improvement or optimization:** Finally, the research can point out opportunities for improvement or optimization in the prioritization method. This is accomplished by reviewing the data and identifying which criteria or weights may need to be altered in order to better fit with the software development approach.

B. Evaluation of Non-functional requirement

- Evaluation of non-functional requirements (NFRs) is an important step in software testing since it ensures that the product satisfies the expected level of performance, dependability, usability, and security. TCP is a frequently used technique in software testing that tries to improve testing efficiency and effectiveness by prioritizing test cases based on their perceived significance.

  The following equation can be used to examine the influence of TCP on software quality:

  $$SQ = f(NFR, TCP)$$

  Where SQ stands for software quality, NFR stands for non-functional requirements, and TCP stands for test case priority. The relationship between these variables is represented by the function f.

- Let us look at an example to better understand this equation. Assume we have created a software system for an e-commerce website that must manage a huge volume of concurrent user requests while maintaining high levels of security and reliability. In this situation, the NFRs may comprise, among other things, performance, security, and reliability requirements.

- To assess TCP's impact on software quality, we can run a series of experiments that alter the TCP approaches utilized and measure the impact on NFRs. We can, for example, prioritize test cases depending on their importance, complexity, or frequency of use and then assess their impact on performance, security, and dependability.

  Assume we run three separate tests using different TCP methods and assess the impact on NFRs as follows:

  Experiment 1: Prioritize test cases based on criticality

  NFRs:
  - *Performance: Improved by 10%*
  - *Security: Improved by 5%*
  - *Reliability: No significant change*

  Experiment 2: Prioritize test cases based on complexity

  NFRs:
  - *Performance: Improved by 5%*
  - *Security: No significant change*
  - *Reliability: Improved by 10%*

  Experiment 3: Prioritize test cases based on frequency of use

  NFRs:
  - *Performance: No significant change*
  - *Security: Improved by 10%*
  - *Reliability: Improved by 5%*

  Based on these findings, we may conclude that TCP, depending on the NFRs and TCP approaches utilized, can have a considerable impact on software quality. Prioritizing test cases based on criticality and complexity had the most influence on NFRs in this scenario, while prioritizing based on the frequency of usage had a lesser impact. These insights can be used to identify the most effective TCP approach for a specific set of NFRs and improve the overall quality of software.

IV. THE RESULT OF THE RESEARCH

A. Integration with software development methodologies

According to our findings, the efficacy of test case, prioritizing approaches in increasing software quality varies depending on the software development methodology employed. Prioritization strategies that work well in agile processes, for example, may not be as effective in DevOps contexts, where speed and continuous integration are critical.

We also discovered that some prioritization strategies, such as risk-based prioritization, can be especially effective with both agile and DevOps setups, as they can find essential test cases that highlight the most serious flaws at the beginning of the testing process.

Overall, our studies imply that while selecting and applying test case prioritizing approaches, software developers and testers should carefully assess their software development methodology.

B. Evaluation of non-functional requirements

Our research also looked at the effect of test case prioritization on non-functional objectives like performance and security. While test case prioritization can improve functional requirements, we discovered that it may not have the same impact on non-functional requirements.

We discovered that certain priority strategies, such as requirements-based prioritization, may be ineffective in discovering key test cases that uncover non-functional

errors. Techniques such as code coverage-based prioritization, on the other hand, may be more helpful in discovering performance-related issues.

Our analyses imply that when using test case prioritization approaches, software developers and testers should carefully assess the types of requirements they are prioritizing. They should also think about complementing their prioritization strategies with extra testing methods developed specifically to detect non-functional concerns.

Overall, our study emphasizes the importance of thoroughly evaluating the impact of test case prioritization on both functional and non-functional requirements in order to comprehensively improve software quality.

V. Discussion

The purpose of this study was to investigate the effect of test case priority on software quality in the context of various software development approaches and non-functional requirements. Our findings indicate that test case prioritization can be a useful technique for enhancing software quality; however, its efficiency varies depending on the software development methodology and the types of requirements emphasized.

In terms of integration with software development processes, our research emphasizes the importance of carefully selecting prioritization strategies that correspond with the unique requirements and limits of each methodology. Techniques that perform well in agile organizations, such as risk-based prioritization, may not function as well in DevOps situations, where continuous integration and speed are critical. As a result, when selecting and applying test case prioritization approaches, both developers and testers should carefully assess their software development methodology.

In terms of evaluating non-functional needs, our research reveals that test case priority may not have the identical influence on non-functional requirements as it does on functional requirements. This emphasizes the significance of augmenting prioritizing strategies with additional testing methods intended particularly to detect non-functional concerns. Furthermore, our studies suggest that requirements-based prioritization may be ineffective in identifying critical test cases that detect non-functional issues, whereas code coverage-based prioritization may be more effective in identifying performance-related issues.

Finally, our research sheds light on the effectiveness of test case prioritization in improving software quality in various contexts. Our studies imply that when selecting and adopting prioritization approaches, software developers and testers should carefully examine their software development methodology as well as the sorts of requirements being prioritized. Additional study is required to investigate additional possible variables that may influence the effectiveness of test case prioritization, as well as to validate our findings through empirical testing and case studies.

VI. Conclusion of the research

Finally, our study attempted to investigate the impact of test case prioritization on software quality, with a particular emphasis on how it interacts with various software development approaches and the evaluation of non-functional needs.

According to our observations, test case prioritization can have a considerable impact on software quality, particularly when combined with agile and DevOps approaches. We also discovered numerous test case prioritizing strategies that can be beneficial in a variety of settings, including risk-based priority, code-coverage-based prioritization, and customer-value-based prioritization.

Furthermore, we discovered that evaluating non-functional requirements, such as performance and security, is critical for gaining a more complete understanding of the benefits and limitations of test case prioritization. Our findings show that prioritizing test cases based on non-functional criteria can enhance software quality significantly, particularly in high-performance and security-critical systems.

Considering these findings, we recommend that future research focus on developing more advanced strategies for test case prioritization that can adapt to varied software development methodologies and non-functional needs. It would also be useful to explore the impact of test case prioritization on other software quality measures such as maintainability and usability.

In general, our research gives important insights into the impact of test case prioritization on software quality and offers practical recommendations for efficiently incorporating it into software development processes.

VII. References

[1] L. Baresi, D. Di Nardo and C. Ghezzi, "Coverage criteria for testing object-oriented software.," *ACM Transactions on Software Engineering and Methodology*, vol. 17, no. 4, pp. 1-45, 2008.

[2] S. Elbaum, G. Rothermel and J. Penix, "Techniques for improving the effectiveness of random testing," *ACM SIGSOFT Software Engineering Notes*, vol. 25, pp. 191-200, 2000.

[3] N. Mirzaei, S. Malek and L. Tahvildari, "Test case prioritization for effective software regression testing," *IEEE Transactions on Software Engineering*, vol. 40, pp. 199-218, 2014.

[4] X. Dai, J. Tao, H. Mei and Y. Qi, "A novel test case prioritization approach in DevOps environment," *IEEE Access*, vol. 8, pp. 46378-46387, 2020.

[5] L. C. Briand, Y. Labiche and R. Yan, "An experimental comparison of the effectiveness of prioritization techniques for performance testing.," *Empirical Software Engineering*, vol. 22, pp. 373-417, 2017.

[6] R. Kavitha, V. R. Kavitha and N. Suresh Kumar, "Requirement based test case prioritization," *2010 INTERNATIONAL CONFERENCE ON COMMUNICATION CONTROL AND COMPUTING*

TECHNOLOGIES, pp. 826-829, 2010.

[7] S. Elbaum, G. Rothermel, S. Kanduri, and A. G. Malishevsky,"Selecting a Cost-Effective Test Case Prioritization Technique," *Software Quality Journal* , vol. 12, pp. 185-210, 2004.

[8] S. Mirarab and L. Tahvildari,"An Empirical Study on Bayesian Network-based Approach for Test Case Prioritization," *2008 1st International Conference on Software Testing, Verification and Validation*, pp. 278-287, 2008.

[9] G. Rothermel, R. H. Untch, Chengyun Chu, and M. J. Harrold,"Test case prioritization: an empirical study," *Proceedings IEEE International Conference on Software Maintenance - 1999 (ICSM'99). 'Software Maintenance for Business Change*, pp. 179-188, 1999.

[10] Z. Sultan, R. Abbas, S. Nazir, and S. Asim, “Analytical Review on test cases prioritization techniques: An empirical study,” International Journal of Advanced Computer Science and Applications, vol. 8, no. 2, 2017.