Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 45 No. 1 (2024)

Parameterized and Prioritized (P2M-TCP) Cloud Testing Approach for Optimized Fault Detection Rate (FDR)

Pushpanjali Saini[1], Anubha Jain[2], Vaishali Singh[3]

1 Department of Computer Science and IT, IIS (deemed to be) University, Jaipur, Rajasthan, India;

2 Department of Computer Science and IT, IIS (deemed to be) University, Jaipur, Rajasthan, India;

3 Department of Computer Science, St. Xavier's College, Jaipur (XIMI), Rajasthan, India;

Abstract:- Cloud Testing is one of the prominent areas to estimate functionality of the Software. It is the process of evaluating the quality of resources in the cloud environment. According to the prior studies, 50% of the performance issues in Cloud Software testing are faced due to the minimum optimal utilization of time, cost and effort. Software testing required dedicated infrastructure which becomes unaffordable for Small-Medium Scale Industry (SMSI). As per the requirements of SMSI, moving testing to cloud would deduct the cost of acquiring required infrastructure. The optimized cloud testing aims to detect maximum faults which could result in evaluation of High Fault Detection Rate (FDR). The research proposed P2M-TCP (Parameterized and Prioritized: Mutation Testing-Test Case Prioritization) approach for testing a software in cloud environment. The objective of the research is to improvise testing by achieving higher fault detection rate through the amalgamation of mutation testing and test case prioritization. The proposed research performs cloud testing based on two approaches, Mutation Testing and Test Case Prioritization. In mutation, the program assertions are mutated to achieve higher code coverage for maximum fault detection and TCP is applied on test cases produced against mutated code based on Occurrence Probability, Severity, Functional Interactivity. The focus of the proposed approach is to achieve higher fault detection rate by prioritizing test cases on the basis of test case execution history and other identified parameters. The current study implements the Parameterized and Prioritization approach to enhance cloud testing performance. The uniqueness of the proposed framework is revealed by comparison of the proposed framework with existing techniques used in other studies.

Keywords*: Cloud Testing, Test Case Prioritization, Mutation Testing, Testing Framework, Software testing.*

1. Introduction

Cloud Testing is a type of software testing which refers to the process of evaluating the functionality of the software using cloud computing services to simulate real world traffic. The goal of software testing is to remove bugs, errors, gaps and missing requirements in the software. Testing also focus on customer satisfaction by analyzing the performance and quality of the software before delivering it to the customer as it also provides the benefits of decreased cost and time along with enhanced security [1]. For customer satisfaction, testing a software requires enough resources and budget to complete it successfully which is practically very expensive and infeasible for a small and medium scale organization to attain adequate resources. Therefore, testing an application on cloud can lower the cost of obtaining infrastructure, time consumed in evaluation and will lead to proper resource utilization [2].

As time, cost and effort are the major parameters that effect the software testing, therefore previous studies had analyzed various software testing approaches that makes testing more effective [3]. Among which, one of the unit testing approaches i.e. Mutation Testing was visited to achieve maximum code coverage to detect faults [4]. The

mutants generated to test the code requires equal num of test case generation which could lead to larger number of test cases. To reduce the cost and effort invested in executing all the test cases, one of the regression testing approaches i.e. Test Case Prioritization was studied to prioritize test cases which would execute most important test cases on priority. Our proposed research is the amalgamation of unit testing (mutation) and regression testing (prioritization) to produce effective testing output.

The testing is considered to be effective if it produces highest Fault Detection Rate (FDR) and highest FDR could be obtained by maximum code coverage of the application. Since mutation testing provides higher code coverage by mutating the code, therefore in our proposed work it is applied on the application to be tested on the cloud. The motto of the mutation testing is to detect maximum faults by introducing mutants in the program assertions. The mutated program assertions require equivalent number of test case generation, which will lead to maximum consumption of time and effort in test case execution. So, to minimize the time consumed in test case execution, Test Case Prioritization (TCP) is applied on test suite. TCP prioritizes test cases from highest to the lowest priority and the prioritization is done on the basis on test case execution history, occurrence, criticality and interactivity of the test cases. In order to implement the above-mentioned approach, the research has proposed Parameterizes and Prioritized Mutation-TCP framework. The mutation testing works in parameterized fashion by focusing on major parameters (time, cost, effort) and TCP works as prioritized approach for test cases. Initially the cost of incurring infrastructure is reduced by moving testing to cloud and then time and effort of the testing is minimized by the application of both mutation and prioritization on execution process of testing the application. Beyond this introduction, the remainder of this paper is organized as follows. The next Section highlights review of literature which identifies the parameters for achieving highest fault detection. Section III and IV throw light on the Parametrized approach and the Prioritized approach on test cases respectively. Section V proposes the Parameterized and Prioritized approach based on Mutation Testing and Test Case Prioritization (P2M-TCP). Section VI explains the framework and methodology used, preceding with the Results and Discussions in the next section. Then ‘Conclusion and Future Research Directions’ are reported at the end of the paper.

2. Review Of Literature

The earlier detection of faults is associated with decision making techniques, expert system and Fuzzy Logic for high performance of the system which minimizes risk of failure. Fault Detection and diagnosis depicts high efficacy and quality production system in real time environment [5]. The operational testing covers most valuable parts of the system that could be merged with code and statement coverage for more effective bug fixing. Fault Detection capability of a test case enhances the reliability of a software [6]. For higher fault detection APFD metric is used to calculate average faults per minute and to determine the efficiency of prioritized and non-prioritized test cases [5].

To increase the performance of regression testing Test Case prioritization is used to avoid re-execution of all the test cases after modification. It aims at faster coder coverage and earlier fault detection [7]. The frequency response analysis is used for detecting the mechanical and electrical faults. To obtain information such as occurrence probability, type and severity of faults, the transfer function is measured and compared to the reference transfer function [8]. According to Quality-One site, the failure mode effects and criticality analysis is used to examine the effect of each failure on system performance. The criticality could be analysed using quantitative and qualitative approach. The choice of the approach to be used is dependent on the failure data rate [9]. An approach for optimized test case prioritization is performed with the application of Ant Colony Optimization i.e ANT-TCP. The algorithm uses the number of uncovered faults by selected test cases and the sum of severity of faults to minimize regression testing time [10].

For effective software testing three input parameters i.e., Occurrence Probability, Severity and Functional Data Interactivity is used along with Fuzzy Logic. Such an approach is used to prioritize test cases to ensure software reliability under the same testing budget [11]. The fuzzy technique is also used to represent the allocation factor for assigning weight to the test cases. To minimize time and cost, allocation of test cases should be based on the significance of the function. The parameters such as complexity, cost, criticality and occurrence probability are considered for prioritizing the test cases [12]. The the Software Defect estimation model using Bayesian belief network for effective time and cost management at earlier stage of software testing [13]. In software testing

coverage criteria is a powerful tool. Instead of full coverage, the relative coverage could provide higher reliability. In relative coverage the entities which are not required by the user are excluded in test case selection to provide more optimized prioritization [14].

The objective of any software testing strategy is to provide highest fault detection. Dynamic Random Testing (DRT) uses the testing results for selecting test cases. The distance-based DRT technique utilizes distance information of inputs which acts as a parameter for test case classification for improved fault detection [15]. The strategy named Covrel+ based on operational profile information and white box coverage measures for generating, selecting and allocating test cases. The operational and coverage data provides greater reliability and fault detection [16]. The regression-based approach is used on machine learning to select critical test cases. Various machine learning models are evaluated for test case selection with the application of natural language processing [17]. The Learn to Rank algorithm with Back Propagation technique is implemented to enhance the performance and improve fault detection rate. The research also utilized MATLAB for various simulations which increased fault detection rate and reduced execution time [18]. To reduce the cost of re-executing test suits, TCP methods use Machine Learning to explore the information of System Under Test (SUT) and related test cases [19]. To evaluate the software reliability a dual model integrated with fault detection and correction process has been incorporated for fault detection rate and testing coverage. Least Square Estimation method is used to measure the model parameters [20]. The probability of fault based on coverage requirement and chip quality is analysed. In modelled faults it is the probability with which the fault will occur and the fault is indicated by the test capable of detecting the fault [21].

3. Parametrized Approach On Test Cases

The parameterized approach on test cases works on the logic of mutation testing which is a type of white box testing technique used for unit testing. The set of certain statements in the source code are mutated to check whether the test cases built against it are capable to find errors in the program. The bugs are intentionally introduced into the source code to test the ability of testing tool to detect errors. The changes in the mutant program are extremely small that it does not affect overall objective of the program. The target is to evaluate the quality of test cases which should be strong enough to fail the mutant. In mutation testing actual program assertions are mutated to generate the mutated version of assertions. Test cases are applied to both original program as well as mutant program and the results are compared. If the output is same, mutant is said to be ‘Alive’ else mutant is ‘Killed’.

On behalf of these results mutation score is calculated. The mutation score is defined as the percentage of killed mutants with the total number of mutants. Test cases are mutation adequate if the score is 100% [22]. In mutation testing, program assertions specify the behavior of the system, so program assertions are powerful and effective tool for detecting the software faults. It acts as a guard against faults and errors. Program assertions are observations to source code that are explicit constraints on the system behavior [23].

The elementary aim of writing program assertions is to specify what the system is meant to achieve rather than how it is expected to do. The program assertions support the mutation testing to detect the bugs and errors. $\mathbf{P}_{\mathbf{actual}}$ refers to the original code of the program which will be altered to create mutated program assertion i.e., $\mathbf{P}_{\mathbf{mutated}}$ [24][4]. $\mathbf{P}_{\mathbf{mutated}}$ refers to the mutated program assertion which is created by applying mutation testing and regression testing on the original code to detect the errors. TCP strengthens the program assertions by prioritizing the test cases based on the Test Case Execution History to make effective prioritization of test cases to detect the faults at earlier stage.

The mutated program assertion results in change of code which requires the need to perform regression testing to check whether the mutated programs assertions had changed the overall objective or not. So, in this parameterized approach, the mutated code required equivalent number of test case generation leading to higher code coverage and thus providing the higher fault detection. The larger test case generation demands more time and effort. Therefore, the traffic of parameterized approach could be controlled by application of prioritized approach on test cases.

4. Prioritized Approach On Test Cases

The Prioritized approach focus on rearranging test cases through Test Case Prioritization in the order from highest priority to lowest priority test cases [25]. TCP is categorized as regression testing in which minor change in code does not adversely affect the overall objective. The regression exists if existing software does not execute in the similar way. Regression Testing is of fo19llowing four types [8][17]:

- ***Retest All*** *requires the execution of all the test cases to ensure existence of any bug due to alteration in the code and such execution requires high cost and time.*
- ***Regression Test Selection*** *lowers the cost as compared to Retest All by executing part of the test suite which is most important and discard other test cases.*
- ***Test Suite Minimization*** *is the technique of removing redundant and obsolete test cases but results in diminution of fault detection.*
- ***Test Case Prioritization*** *approach arranges the test cases priority so as to upsurge degree of fault detection.*

Prioritization refers to the arrangement of items or activities in particular sequence based on its importance. Test Case Prioritization (TCP) is an extension to the software testing which is implemented to prioritize the test cases [18][26]. Test Cases are set of instructions which are used to check the functionality of the application. The quality of the test case is measured by the ability of the test to detect the error. Therefore, the quality metrics is used to evaluate the test and seek the scope of improvement required in the Test cases [21]. TCP in assistance to regression testing improves its performance. With the help of this method testers can detect faults at earlier stage by executing test cases having higher priorities. The test case generation is a challenging task as larger test suites leads to higher time and cost constraints. Thus, TCP is used to prioritize and schedule test cases to reduce time, cost and effort.
The major challenge is to prioritize test cases that have same weight values [1]. To enhance the testing TCP prioritizes the test cases according to the priorities assigned TCP prioritizes and scheduled test cases according to their highest and lowest requirements due to which testers ensures that important test cases are executed first. So, the prioritized approach helps to rearrange the test cases from highest to lowest priority test cases with the application of TCP on test cases execution required after mutation testing and provides earlier fault detection by executing crucial test case on highest priority.
The priority of the test cases is based on the parameters analysed for fault detection such as occurrence probability, criticality and functional interactivity including test case execution history. The priorities assigned to the test cases based on the test case execution history, keeps the historical track of the of test cases already executed. This arrangement of test cases in TCP is a step towards highest Fault Detection Rate (FDR) [27][21].

5. Analysis Of Parameterized And Prioritized (P^2m-Tcp) Approach

Through the in-depth study of the literature, Occurrence Probability, Criticality and Functional Interactivity parameters have been identified from the prior studies for detecting highest fault rate at the initial stage. According to the researcher's, these parameters act as most significant factor in software testing techniques [9][28][21][22] (Fig.1.).

Figure 1: Identified Parameters of FDR

The occurrence probability evaluates the frequent occurrence of test cases based on which the consideration of a particular test wase will be determined, if the probability is high then it could be set with highest priority. The criticality highlights the severity of the test case, as non-execution of such test cases would lead to halt an application. The function provides the interaction and communication between the interfaces. The test cases with higher functional interactivity leads to more flexible and responsive application. To make the prioritization easier and effective, above identified parameters have been taken into consideration based on which the test cases will

be prioritized. The study has also identified that the maximum coverage (code coverage or statement coverage) provides better defect detection and reliability. Fault detection is the technique of identifying bugs or errors in the software testing process. The Fault Detection Rate (FDR), also known as Defect Detection Rate is the accuracy rate at which the test cases are able to detect faults. The FDR percentage is calculated by dividing the errors detected at the testing phase by the total number of errors ***[9] [29].***

DDE (Defect Detection Efficiency) = (Number of Defects Detected in a Phase / Total Number of Defects) * 100%

The high code coverage is required to identify faults through program assertions. To achieve higher code coverage mutation testing has been applied for program assertions before prioritizing them, which converts $\mathbf{P_{actual}}$ to $\mathbf{P_{mutated}}$ [30][18]. The equivalent number of test cases need to be generated for executing program assertions. If $\mathbf{P_{mutated}}$ is True, it indicates that the code is error free and if $\mathbf{P_{mutated}}$ is False, it indicates that there is a bug and it needs to be rectified. At this point, time and effort of generating test case generation could be reduced by automation of test case generation in the cloud environment. It could be achieved through TestComplete is free of cost available tool capable of testing desktop, mobile and web application and could be used by technical as well as non-technical. The automation of test case generation in mutation testing goes through several stages such as mutation generation (by altering operators, constants etc. which is mostly achieved by mutation tools such as PIT, MutPy etc.), mutant analysis, augmentation of test cases and then prioritizing the test cases to uncover new mutants. For optimizing test case execution test cases would be prioritized based on occurrence probability, Criticality and Functional Interactivity results. The occurrence probability indicates the number of times the test case can occur in testing process, criticality analysis predicts the severity of the test case which depicts most valuable test cases to the user and the functional interactivity provides the in-depth connectivity of the functions among the test cases. The table:1 created below shows weightage of test cases on the basis of these parameters.

OCCURRENCE PROBABILITY	CRITICALITY	FUNCTIONAL INTERACTIVITY	TEST CASE PRIORITY WEIGHTAGE
✓	✓	✓	High
✓	✓	X	High
X	✓	✓	High
✓	X	✓	Average
✓	X	X	Average
X	✓	X	High
X	X	✓	Low
X	X	X	Low

Table 1: Test Case Weightage Matrix

The above table portrays the priority of test case based on the availability of parameters in following scenario:

- If all the three parameters are present then test case is executed on High priority.
- If test case is critical then also it is executed on High priority.
- If any two parameters except criticality exists the test case is executed on Average priority.
- If either one parameter exists except criticality or no parameter exists the test case is executed with Low priority.

The results of above program assertions will be used in our future work while implementing fuzzy logic.

6. P²M-TCP FRAMEWORK APPROACH

In the proposed framework, functional and non-functional techniques both are implemented indirectly. Our previous research [3][4][25], had focused on TCP and mutation testing and further research of this study grasped the shape of the proposed framework. The Test case Prioritization is a type of Regression Testing which is black

box testing, so it promotes Non-Functional testing. On the other hand, Mutation Testing is white box testing, so it promotes Functional Testing. In comparison to the studies mentioned in literature review of this paper, the research proposal is different from others in terms of the combination of functional and non-functional testing along with the checklist validation to be done on the factors such as time, cost and effort using Fuzzy Logic.

6.1 Methodology

The proposed framework i.e P^2M-TCP (Figure:2) framework focus on earlier fault detection with an aim to minimize time, cost and effort. The framework is divided into three steps:

Step 1: In this phase based on the previous literature review, software testing factors i.e time, cost and effort are selected which have highest impact on fault detection while testing a software. The validation of these factors will be done through checklist framework using fuzzy logic.

Step 2: In the second phase of the framework, the parameterized approach is based on one of the functional testing approaches which will conduct white box testing i.e mutation testing. The mutation testing is done to achieve higher code coverage to achieve maximum fault detection.

The application will first go through the mutation testing to detect maximum faults by generating equivalent number of mutants. As it will lead to more consumption of time and cost in executing all the test cases, therefore to achieve earlier fault detection, the test cases are prioritized based on test case execution history, occurrence probability, severity and functional interactivity. This will execute highest priority test cases first leading towards lowest priority test cases.

Step 3: The mutation testing providing maximum code coverage combines with test case prioritization proving earlier fault detection results in optimized fault detection rate which benefits in terms of time cost and effort.

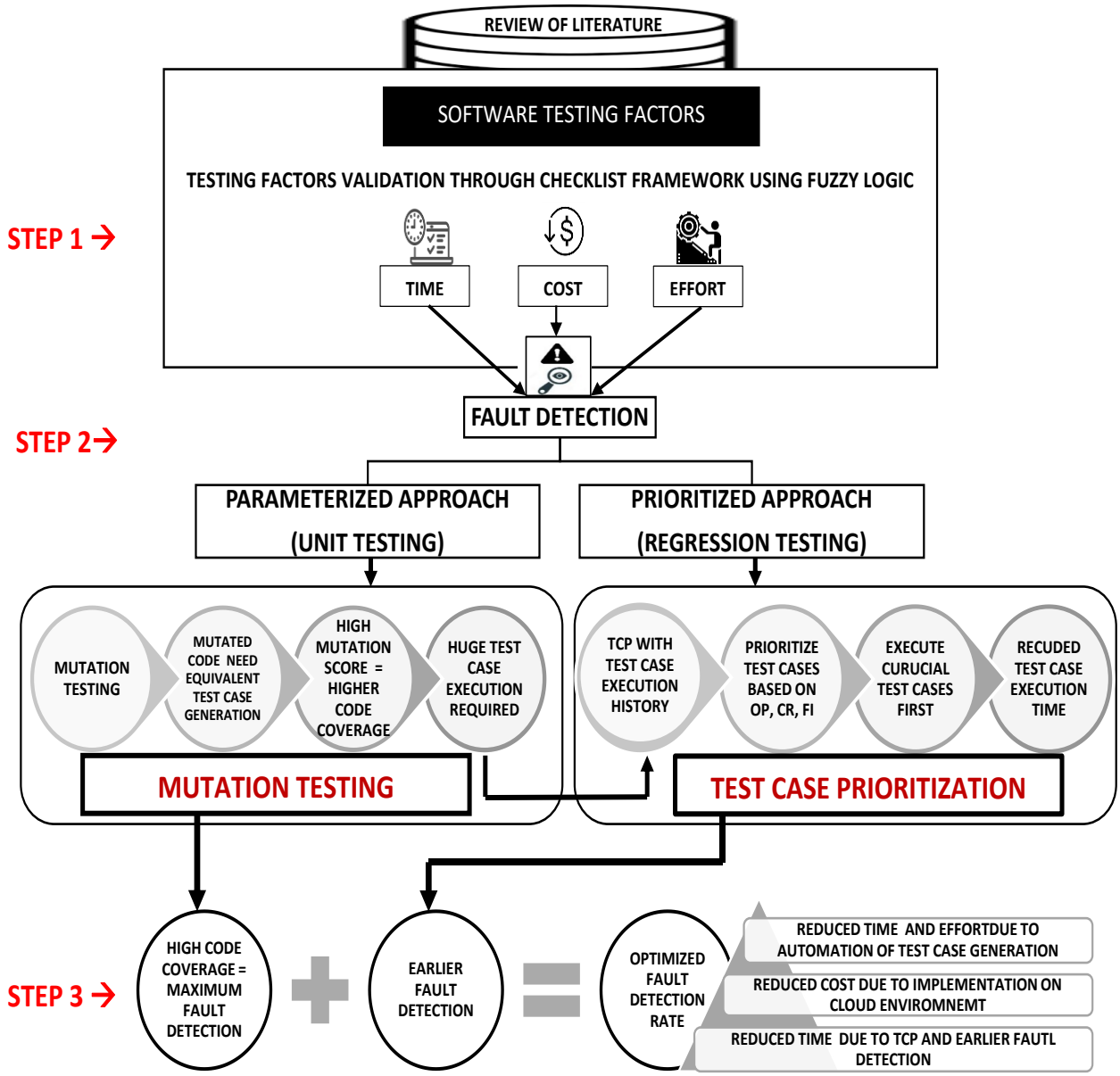

Figure 2: P²M-TCP Framework

7. Results And Discussion

The application initially will go through the mutation testing to detect maximum faults by generating equivalent number of mutants. As it might lead to high consumption of time and cost in executing all the test cases, therefore to achieve earlier fault detection, the test cases are prioritized to base on test case execution history, occurrence probability, severity and functional interactivity. This will execute highest priority test cases first leading towards lowest priority test cases. The uniqueness of the proposed approach is reflected through the below given table: depicting the combination of unit and regression testing along with its validation through checklist framework which has not been done in any of the previous studies.

FRAMEWORKS/ METHODS/ PRIOR STUDIES	CHECKLIST FRAMEWORK	REGRESSION TESTING (TCP)	UNIT TESTING (MUTATION)

P^2M-TCP Approach (Proposed Research)	**Y**	**Y**	**Y**
Dynamic Random Testing (DRT) [16]	**N**	**Y**	**N**
Frequency Response Analysis [9]	**N**	**N**	**Y**
ANT-TCP [11]	**N**	**Y**	**N**
Least Square Estimation method [21]	**N**	**Y**	**N**
APFD [7]	**N**	**N**	**Y**
Covrel + [17]	**N**	**Y**	**Y**

Table 2: Comparison with existing approaches

The outcome of the framework is optimized fault detection rate fulfilling the objective of effective testing with minimized time, cost and effort. The testing requires execution of larger number of test cases leading towards high time frames, which could be strained by spreading test cases across numerous virtual machines concurrently on cloud infrastructure. Parameterized and Prioritized Mutation-TCP shortens total testing time and maximizes resource use by running tests concurrently. Through this, Parameterized and Prioritized Mutation-TCP approach reduces the overall testing time and optimizes resource utilization. The cloud auto scaling feature could be used to adjust number of virtual machines according to the testing load to ensure availability of testing resources at the peak timings. Moreover, on the basis of the priority weightage more resources could be allocated to critical components.

8. Conclusion And Future Work

Software Testing requires huge budget and resources to complete it efficaciously [5], therefore it is not always possible for small and medium scale industries to meet such requirements. Most of the researchers have also analyzed that maximum time, cost and effort is spent on software Testing [21].
To address such issues, testing a software on cloud provides good alternative. The Fault detection rate is the major factor for achieving optimized results in testing a software. Thus, the testing techniques which provides highest fault detection rate should be considered.

The conclusion of the above study depicts that mutation testing with Test case Prioritization results in optimized fault detection. On the basis of literature review the identified parameters: occurrence probability, Criticality and Functional Interactivity are used for selecting and prioritizing the test cases to detect the faults at earlier stage. The proposed P^2M matrix of program assertion and the framework was compared with existing research studies. To sum up, the Parameterized and Prioritized Cloud Testing Approach has advantages beyond only finding faults. Its methods for prioritizing test cases, allocating resources, and performing mutation testing in a cloud environment are in line with contemporary software development techniques, promoting productivity, code quality, security, and innovation. These strategies are positioned to be extremely important in assuring the excellence and reliability of software products as software systems become more complex, dynamic, and distributed. Although automated test case generation in mutation testing provides useful insights into the effectiveness of test suites, still combining automated methods with domain knowledge and human judgment constitutes a balanced strategy. The overall quality of the testing process can be improved by using manual code inspections, extra testing methods, and carefully choosing mutation operators to help overcome some of the restrictions.

In our future work, a checklist will be designed to identify checkpoints for P^{2}M-TCP framework which will be validated based on time, cost and effort matrix using fuzzy logic. The validation and verification of P^{2}M-TCP approach will proceed with fuzzy checkpoints implementation on various projects.

Refrences

[1] Ahmad, J., & Baharom, S. (2018). Factor determination in prioritizing test cases for event sequences: A systematic literature review. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 10(1-4), 119-124.

[2] Almorsy, M., Grundy, J., and Müller, I (2016). An analysis of the cloud computing security problem. arXiv preprint. arXiv:1609.0110.

[3] Pushpanjali, S., & Anubha, J. (2021). A Model for Effective Software Testing in Cloud Environment. *In Rising Threats in Expert Applications and Solutions* (pp. 145-153). Springer, Singapore.

[4] Saini, P. (2019). Revisiting Mutation Testing in Cloud Environment (Prospects and Problems). *A Journal of Composition Theory*, 12(9), 2007-2011.

[5] Alvarez, G. P. (2020). Real-time fault detection and diagnosis using intelligent monitoring and supervision systems. In Fault Detection, Diagnosis and Prognosis. IntechOpen.

[6] Andrade, L., Machado, P., & Andrade, W. (2020). Can operational profile coverage explain post-release bug detection? *Software Testing, Verification and Reliability*, 30(4-5), e1735.

[7] Badhera, U., Purohit, G. N., & Biswas, D. (2012). Test case prioritization algorithm based upon modified code coverage in regression testing. *International Journal of Software Engineering & Applications*, 3(6), 29.

[8] Bigdeli, M., Azizian, D., & Gharehpetian, G. B. (2021). Detection of probability of occurrence, type and severity of faults in transformer using frequency response analysis based numerical indices. Measurement, 168, 108322.

[9] Failure Mode, Effects & Criticality Analysis (FMECA), Quality-One, https://quality-one.com/fmeca/.

[10] Vescan, A., Pintea, C. M., & Pop, P. C. (2022). Test case prioritization—ANT algorithm with faults severity. *Logic Journal of the IGPL*, 30(2), 277-288.

[11] Gupta, P., & Singh, P. (2021). Priority-wise Test Case Allocation using Fuzzy Logic. *International Journal of System Assurance Engineering and Management*, 1-14.

[12] Yadav, D. K. (2019). Software Test Case Allocation. In Proceedings of the 2nd *International Conference on Data Engineering and Communication Technology* (pp. 137-144). Springer, Singapore.

[13] Kumar, C., & Yadav, D. K. (2017). Software defects estimation using metrics of early phases of software development life cycle. *International Journal of System Assurance Engineering and Management*, 8(4), 2109-2117.

[14] Miranda, B., & Bertolino, A. (2020). Testing relative to usage scope: Revisiting software coverage criteria. *ACM Transactions on Software Engineering and Methodology* (TOSEM), 29(3), 1-24.

[15] Pei, H., Yin, B., Xie, M., & Cai, K. Y. (2021). Dynamic random testing with test case clustering and distance-based parameter adjustment. *Information and Software Technology*, 131, 106470.

[16] Bertolino, A., Miranda, B., Pietrantuono, R., & Russo, S. (2019). Adaptive test case allocation, selection and generation using coverage spectrum and operational profile. *IEEE Transactions on Software Engineering*, 47(5), 881-898.

[17] Cheruiyot, V. (2021). Machine learning driven software test case selection.

[18] Singh, J., & Sharma, S. (2018). Fault detection technique for test cases in software engineering. *International Journal of Engineering & Technology*, 7(1), 53-56.

[19] Ramírez, A., Feldt, R., & Romero, J. R. (2022). A Taxonomy of Information Attributes for Test Case Prioritization: Applicability, Machine Learning. *ACM Transactions on Software Engineering and Methodology*.

[20] Li, Q., & Pham, H. (2021). Modeling software fault-detection and fault-correction processes by considering the dependencies between fault amounts. Applied Sciences, 11(15), 6998.

[21] Seth, S. C., & Agrawal, V. D. (1989). On the probability of fault occurrence. In Defect and fault tolerance in VLSI systems (pp. 47-52). Springer, Boston, MA.

[22] Al-Ghuwairi, A. R., Eid, H., Aloran, M., Salah, Z., Baarah, A. H., and Al-oqaily, A. A. (2016). A mutation-based model to rank testing as a service (TaaS) Providers in cloud computing. *In Proceedings of the International Conference on Internet of things and Cloud Computing* (p. 18). ACM.

[23] Rosenblum, D. S. (1995). A practical approach to programming with assertions. *IEEE transactions on Software Engineering*, 21(1), 19-31.

[24] Saini, P., Jain, A., Singh, V., & Kaur, R. (2023). Effective Test Case Prioritization Framework Using Fuzzy Logic for Cloud-Based Applications. In Proceedings of Seventh *International Congress on Information and Communication Technology* (pp. 569-576). Springer, Singapore.

[25] Siddiqui, T. and Ahmad, R. (2016). A review on software testing approaches for cloud applications. Perspectives in Science, 8, 689-691.

[26] Wang, H., Yang, M., Jiang, L., Xing, J., Yang, Q., & Yan, F. (2020). Test Case Prioritization for Service-Oriented Workflow Applications: *A Perspective of Modification Impact Analysis. IEEE Access*, 8, 101260-101273.

[27] Rosenblum, D. S. (1992, June). Towards a method of programming with assertions. In Proceedings of the 14th international conference on Software engineering (pp. 92-104).

[28] Katherine A. V. and Alagarsamy K., (2012). Software testing in cloud platform: a survey. *International Journal of computer applications*. 46(6), 21-25.

[29] Srivastava, P. R. (2008). Test case prioritization. *Journal of Theoretical & Applied Information Technology*, 4(3).

[30] Defect Detection Efficiency, Software Testing Fundamentals,https:/ /software testingfundamentals.com/defect-detection-efficiency/