International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9s
DOI: https://doi.org/10.17762/ijritcc.v11i9s.7453
***Article Received**: 16 May 2023 **Revised:** 03 July 2023 **Accepted:** 31 July 2023*

Requirement based Test Case Prioritization for System Testing

Harish Kumar[1], Vedpal[2], Umesh Kumar[3], Naresh Chauhan[4]
[1]Department of Computer Engineeing
J C Bose University of Science & Technology, YMCA
Faridabad, India
htanwar@gmail.com
[2]Department of Computer Applications
J C Bose University of Science & Technology, YMCA
Faridabad, India
ved_ymca@yahoo.co.in
[3]Department of Computer Engineeing
J C Bose University of Science & Technology, YMCA
Faridabad, India
umesh554@gmail.com
[4]Department of Computer Engineeing
J C Bose University of Science & Technology, YMCA
Faridabad, India
nareshchauhan19@gmail.com

Abstract— System Testing encompasses a large number of test cases, which may not be able to get executed due to constrained time, budget and limitation of the resources. Therefore, the test cases must be prioritized in some order such that the critical and most required functionality can be tested early. In this paper, a hierarchical approach for system test case prioritization based on requirements has been proposed that maps requirements on the system test cases. This approach analyzes and assigns value to each requirement based on a comprehensive set of twelve factors thereby prioritizing the requirements. Further, the prioritized requirement is mapped on the highly relevant module and then prioritized set of test cases. To analyze the effectiveness of this approach, a case study of income tax calculator software [1] has been taken. The existing as well as the proposed approach were applied and analyzed on this software. The results show the efficacy of the proposed approach in terms of fault detection and severity early.

Keywords- System Testing, Test case prioritization, fault severity,tcp.

I. INTRODUCTION

Test case prioritization techniques organize the test cases in a test suite by ordering in such a manner that the most critical test cases are executed first thereby increasing the effectiveness of testing. The prioritization techniques [2] provide a way to find out more bugs under resource constrained environment and thus improve the reliability of the system quickly. Moreover, as faults are revealed earlier, software engineers have more time to fix the bugs and adjust the project schedule. Many prioritization techniques have been proposed for prioritizing the system test cases based on requirements. However, the requirements only in consideration cannot include critical test cases. The implementation complexity and test case complexity may also affect the test case prioritization. Though Hema Srikanth [3] has included the developer perceived complexity for implementation factor but it is only a scaling assigned by developer explicitly. There may be lot of complexities and issues in design and code of the mapped requirements. All these factors should also be considered while prioritizing the test cases. The researchers have also considered, fault proneness of requirements only in connection with customer-reported failures. But there is need to consider fault-proneness for every requirement with every affected factor. Moreover, the fault proneness associated with mapped code should also participate in prioritizing the test cases.

In this paper, a hierarchical test case prioritization is proposed wherein the prioritization process is performed at three levels given below:

(1) The requirements are first prioritized on the basis of twelve factors by assigning a priority weightage to each requirement.

(2) The highest priority requirements are then mapped to their corresponding modules to get prioritized modules.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9s
DOI: https://doi.org/10.17762/ijritcc.v11i9s.7453
***Article Received**: 16 May 2023 **Revised:** 03 July 2023 **Accepted:** 31 July 2023*

(3) The test cases based on to the highest prioritized module are then put for execution.

II. RELATED WORK

Hema Srikanth et al. [3] considered four factors for analyzing and measuring the criticality of requirements. These factors are Customer-Assigned priority of requirements, Requirement Volatility, Developer- perceived implementation complexity. Based on these four factor values, a Prioritization factor value (PFV) is computed. PFV is then used to produce a prioritized list of system test cases. R. Kavitha & N.Suresh Kumar [4] proposed a method to prioritize the regression test cases considering the following factors: (1) Customer assigned priority of requirements, (2) Developer-perceived code implementation complexity, (3) Changes in requirements, (4) Fault impact of requirements, (5)Completeness, (6) traceability (7) Execution time. Based on these factors, a weightage was assigned to each test case in the software thereby prioritizing the test cases.

Patric Berander and Anneliese Anfrews [5] considered an approach that provides means to find an optimal subset of requirement resulting in trade of desired project scope against sometime conflicting constraint such as schedule, budget, resources, time to market and quality. They also considered requirement prioritization as the basis of the product strategy. Maya Daneva and Andera Herrman [6] proposed a conceptual model of requirements prioritization based on benefit and cost prediction.

Siripong Roongruangsuwan and Jirapun Daengdej [7] proposed a new classification of test case prioritization techniques considering a new test case prioritization method along with practical weight factors like test case complexity, dependency, and test impact etc. Thillaikarasi Muthusamy et al. [8] proposed a technique which prioritizes the test cases based on four groups of practical weight factor such as: customer allotted priority, developer observed code execution complexity, changes in requirements, fault impact, completeness, and traceability. M.Kalaiyarasan and Dr.H.Yasminroja [9] proposed a version specific test case prioritization technique which uses data flow information. The proposed technique considered the fault detection capabilities of test cases for prioritization purpose. They find out four different categories of software modification and use the data flow information for prioritization purpose. Johana Ahmad et al. presented [10] the results that are obtained from the 70 primary studies. They investigated and indentified the factors that are used to prioritize the test cases. Their studies show that the 10 factors that should be used to improve the existing test case prioritization technique. The identified factors are the Fault, redundancy, complexity, frequency, requirements, Time, Distance, Cost, permutations, and others.

Manaswini B et al. presented [11] a test case prioritization technique to perform regression testing. The proposed technique is based on the cat swarm optimization algorithm. For experimental validation the applied the presented technique on some open-source applications likes jtopas and jmeter. They [12] also presented a technique using the shuffled frog leap algorithm. They used the metrics like code coverage, execution time etc.

Remo Lachmann et al. proposed [13] a technique to prioritize the system test cases using the supervised machine learning. They used the black box Meta - Data like test case history. For evaluation of the proposed approach, they used the SVM rank machine learning algorithm.G. Bhavyasri et al. proposed [14] a technique to prioritize the test cases. They used the functional dependency to cluster the test cases. The test cases are prioritized on the bases of the function coverage. Rayapureddy Kalyani et al. investigated [15] that whether the grouping of the requirements helps in improve the effectiveness of the test case prioritization techniques. They used the code scope metric for grouping the requirements. Manoj Kumar Sahu et al. proposed [16] a test case prioritization for regression testing. They used the business criticality value. They also validated the proposed algorithm and found the effectiveness of the presented approach.

Zubair Rashid Bhat et al. presented [17] the hybrid approach of test case prioritization. They used the robust genetic algorithm to enhance the parameter like execution time. Song Wang et al. presented [18] quality aware test case prioritization technique (QTEP). They addressed the limitation of the existing the coverage-based algorithms. They leverage the code inspection technique. They found that the QTEP helps to improve the efficiency of existing TCP techniques. Naresh Chauhan et al. [19] discussed about the role of machine learning in the designing of testing techniques to perform the testing of the software. They found that the testing technique based on the machine learning are affective as compared with the traditional techniques Rongi pan at al. [20] presented review of various literature to select and prioritize the test cases using the machine learning. They found that various machine learning techniques are used to prioritize the test cases by considering the various types of the feature like code complexity, user inputs, execution history etc.

Cristina Maria Tiutin et al. [21] used the neural network classification technique to prioritize the test cases.

The neural network has trained using the different factors like the association between the requirements, tests and discovered

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9s
DOI: https://doi.org/10.17762/ijritcc.v11i9s.7453
***Article Received**: 16 May 2023 **Revised:** 03 July 2023 **Accepted:** 31 July 2023*

faults. Jijo Joseph C George et al. presented [22] a study related to the various techniques and effectiveness to prioritize the test cases. Ali Samad at al. used [23] the multi-objective particle swarm optimization to prioritize the test cases. They considered the various factors like execution time, code coverage etc. Elinda Kajo Mece at al. investigated [24] the various machine learning application used to order the test cases. They reviewed the recent proposed work and introduce the various information like process, metric and measure the effectiveness

A critical review of the work done by the researchers in the direction of system test case prioritization indicates that the following factors have not been considered that may affect the system test case execution:

Developer Assigned Priority: The developer may assign the priority to every requirement on the basis of its importance.

Show Stopper Requirements: These are the critical requirements in the absence of which the software may not work. The developer may therefore assign the priority to these types of requirements.

Frequency of Requirements: It is the frequency of a requirement how much it is being used in the software.

Expected fault: The developer may analyze the causes which may make the software error prone.

Implementation Complexity: It is the criteria how much each requirement is difficult to implement considering technology dependency, interdependency of the requirements, complexity of requirement itself, etc.

Cyclomatic Complexity: It is the logical complexity [1] of a program. The module with higher complexity may lead to complex test cases.

Non DC path: In data flow graph [1] of a program, the non-dc paths which are the path between the definition node and the usage node of the variable wherein the variable is defined more than once are the problematic areas with respect to the use of a variable. Therefore, this factor may also be considered for module prioritization.

This paper considers the above factors and proposes a new technique for system test case prioritization, which is discussed in next sections.

III. PROPOSED WORK

The proposed Hierarchical System Test Case Prioritization (HSTCP) approach starts with analyzing and assigning value to each requirement based on a comprehensive set of twelve factors thereby prioritizing the requirements. After getting the ordered list of requirements, a mapping between the highest priority requirement and its corresponding modules is performed. The modules are then prioritized based on Cyclomatic complexity and non dc path. The weighted prioritized module is then selected for testing. It may be possible again that there are several test cases corresponding to this selected module. For this purpose, the third level of prioritization is applied by prioritizing these several test cases based on four factors. In this way a hierarchical system test case prioritization (HSTCP) technique is proposed and discussed in subsequent sections. In the proposed prioritization process almost every stakeholder viz. the customer, developer, tester, and business analyst participate. The prioritization process includes the following steps (See Figure. 1).

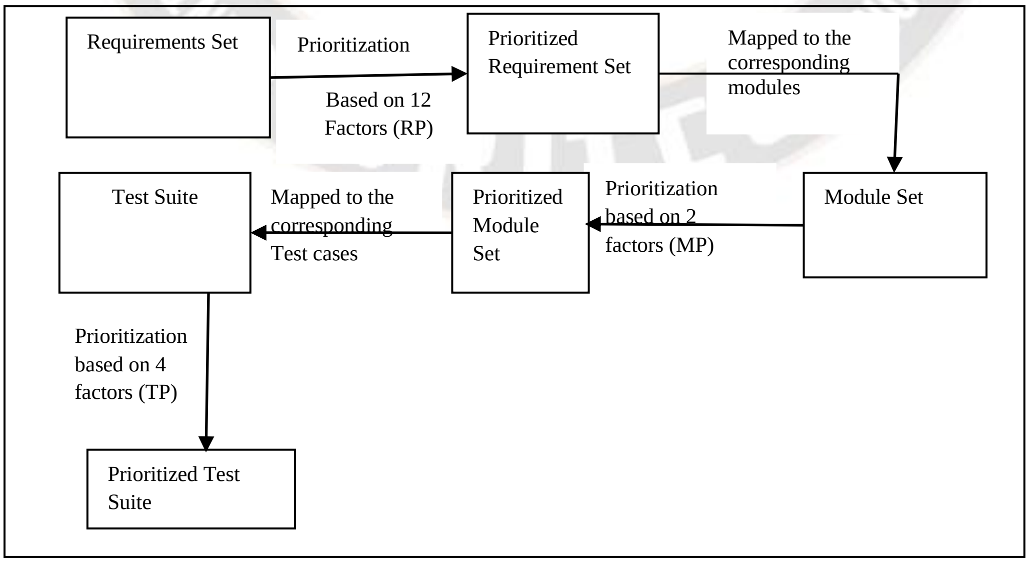

Requirements Set
Prioritization
Based on 12 Factors (RP)
Prioritized Requirement Set
Mapped to the corresponding modules
Module Set
Prioritization based on 2 factors (MP)
Prioritized Module Set
Mapped to the corresponding Test cases
Test Suite
Prioritization based on 4 factors (TP)
Prioritized Test Suite

Figure 1. Process for Proposed HSTCP Approach Based on Requirements

(1) Customer, developer, analyst, and tester assign values to the requirement factors.
(2) Apply the process of the requirements prioritization (RP).
(3) Based on the prioritized requirements a mapping between the requirements and their corresponding modules are performed.
(4) Apply the process of prioritization of the modules (MP).
(5) Tester assigns the value to each factor of test case of the prioritize module.
(6) Apply the process of test case prioritization (TP).
(7) The resulting test suite contains the prioritized test cases.

A. Prioritization of Requirements

There are the various factors on the basis of which process of prioritization of requirements is performed. These factors are in accordance with every phase of SDLC. All these factors have been assigned a priority value between 0 to 10. These priority values are assigned by various stakeholders of the project. The Table 1 shows these factors.

1) Requirements Volatility: Requirement volatility is the frequency of changing a requirement during development cycle of the software.

Reasoning: The most of errors are found during the requirements gathering and analysis phase. If the developers implement the requirement and that requirement changes then developer has to redesign and re-implement the same. Due to reimplementation of requirement, it also increases the fault density in the programs. Studies show [19] that 35 % of the requirements for an average project change before project completion. The requirement with a higher change frequency is assigned a higher priority value as compared to the stable requirements.

2) Customer Assigned Priority: Based on the priority of the requirement, the customer assigns a priority value to each requirement.

Reasoning: Several studies indicate that some requirements of a project are frequently used and some are rarely used. The studies show that approximately half of the software functions are never used. Only 36 % of the software function is always used and most of the faults lie in these functions which are frequently executed. So the customer is involved to know which requirements are very important to him so that these are tested earlier to increase the customer satisfaction. Customer assigns the highest weight to requirement which is very important for him.

3 Implementation Complexity: Each requirement may be analyzed according to how difficult it is to implement. There are various factors considered during requirement implementation. So before assigning a priority value to this factor it is necessary to consider all factors related with that requirement. The priority value for this factor is the sum of the priority values assigned to these factors.

TABLE 1. FACTORS CONSIDERED FOR REQUIREMENT PRIORITIZATION

Sr.No.	Factors	Phase of SDLC	Priority value assigned by
1	Requirement Volatility	Requirement Analysis	Customer
2	Customer Assigned Priority	Requirement Analysis	Customer
3	Implementation Complexity	Design	Developer
4	Fault Proneness of Requirements	Design	Developer
5	Developer assigned priority	Requirement Analysis	Developer
6	Show Stopper requirements	Design	Developer
7	Frequency of execution of requirement	Requirement Analysis	Developer
8	Expected Faults	Coding	Developer
9	Cost	Requirement Analysis	Analyst
10	Time	Requirement Analysis	Analyst
11	Penalty	Requirement Analysis	Customer
12	Traceability	Testing	Tester

There are 3 factors which are taken into consideration as shown in Figure 2. These 3 factors are discussed below.

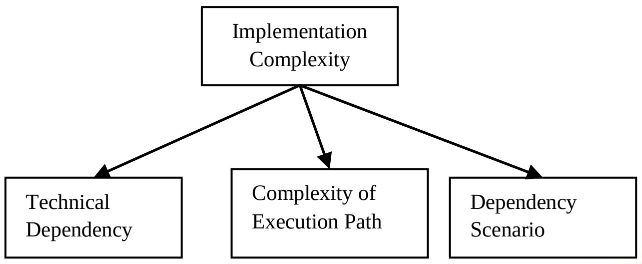


Figure 2. Implementation Complexity Factors

Reasoning: The studies [20] show that more complex is the requirement, more it tends to have faults. So, a priority value is assigned by the developer to this factor.

Technical Dependency: Technology plays a very important role in development of any software. Implementation technique of software is varying from technology to technology. With the selection of suitable technology developer can develop less error prone project within target time and budget. Some time the customer bounds developer to a particular technology. Sometimes the proposed requirements are very complex to implement in selected technology whereas the same requirement can be implemented in other technology without the much complexity and less error. So, this factor is considered for prioritizing the test cases. For this factor a priority value between 0 and 3 is assigned.

Complexity of Execution Path: Sometimes in the project a requirement is very simple to implement thereby its complexity is very low. But to execute that requirement user must follow the complex path of the execution. So, the long path of execution also affects the complexity of requirement. This factor assigned a priority value between 0 and 3.

Dependency Scenario: The studies [21] show that more the dependency between the modules of a requirement higher is its complexity. It means if a requirement is covered by more than one module and the dependency among these modules is high then higher is the complexity of that requirement. For this factor a priority value between 0 and 4 is assigned.

4) Fault Proneness of Requirements: Fault proneness signifies those requirements which are associated with faults or which shows failures in the previous releases of the software. If a requirement in an earlier version of the system has more bugs, then this requirement in the current version is given more weight.

Reasoning: Fault proneness factor is important because the requirements which have shown failures in the earliest release are more faults prone. So, it is important to give more weight to requirements with high fault proneness so that they can be tested on higher priority. This factor is valid for only those requirements which have been implemented in earlier version of software and not valid for the new requirements. So, a priority value is assigned accordingly.

5) Developer Assigned Priority: Developer assigns the priority to every requirement because of the importance of the requirement. Developer assigns the priority value to each requirement ranging from 0 to 10.

Reasoning: Developer plays an important role for successfully completion of a project within target time and budgeted cost. Studies show that more than 50% project are not completed in the target time and cost. Here the developer analyzes each requirement and assigns the weight to each requirement on the basis of that requirement how much it is important for the project. It may happen that lowest priority given by the customer to a particular requirement is very important for the project. So the developer gives a weight to each requirement on the basis how much it contributes towards the success of the project. Larger value of the weight given to a requirement shows it is very critical to the project.

6) Show Stopper Requirements: Show stopper requirement are those requirements based on which software works. Such requirements are given more importance and assigned the priority value accordingly.

Reasoning: In every project there are some core requirements on the basis of which all modules are working. If these requirements are failed then whole project will stop. For example, consider online ticket booking website. By using website user can inquire about the train, see the available seats in a particular train, cancel out ticket, online payments to tickets and book tickets. These are the requirements which are frequently used. Suppose for a moment the online payment system fails, in this case users are not able to book the ticket until customer has not paid for the tickets. So here the online payment system is critical requirement. There may be more than one requirement on which the whole project works.

3.1.7 Frequency of the Execution: In this factor priority value to each requirement is assigned on the basis of its execution frequency. The more priority value is assigned to the requirements which are frequently used. .

Reasoning: In every project there are some requirements which are never executed in product and some requirements are frequently executed. The requirement may be executed directly or may be through the other requirements. Therefore, a priority value is assigned to them on the basis of their frequency of execution. Consider online ticket booking website. By using website user can inquire about the train, see the available seats in a particular train, and cancel the tickets, make online payments to tickets and book tickets are those requirements which are being frequently used. But update the fare of tickets, update the timings of the trains are those requirements which are not frequently used.

8) Expected Fault: This factor identifies the future implementation faults. In this factor developer analyzes the causes which make the software error prone.

Reasoning: The study [22] shows that it is not possible to implement software without faults. The reason that may be responsible for generating the fault should be considered. As the studies show if developer analyzes the fault in the initial phase, then the project will be successfully completed within the time and the budget. The two factors that we are using are shown in Figure 3.

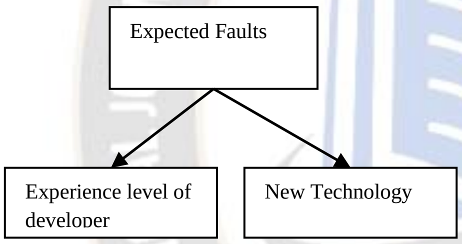


Figure3. Expected Faults

Experience level of developer: The study [23] shows that skills and experience of a developer play an important role in success full completion of a project. Lower is the experience of a developer more is the chance of getting a bug in the implementation of a particular requirement. A developer with lower experience may implements a requirement with higher complexity whereas the experience developer implements same requirements with less error. For this type of requirement, a weight between 0 and 5 is assigned.

New Technology: Sometimes customer bounds the developing team to use a particular platform to implement their requirement then if that technology or platform is never used by the developer then to work on the new platform is difficult for the developer. So, there are more chances of bug in the requirement so a priority value between 0 and 5 is assigned. Higher value is assigned for the very new technology which is never used and medium value which has been used in previous projects and zero value for our current technology used by the developers.

9) Cost: It corresponds to expenditure done to implement the requirements. Here a cost factor is considered for each requirement and a weight between 1 and 10 is assigned. The higher value being assigned to the cost factor shows that cost to implement the requirement is very high.

Reasoning: The software development cost is difficult to predict. The study shows [24] that 45 % projects complete with overrunning the cost. There are many factors which influence the cost of requirements. These factors are: complexity of a requirement, the ability to reuse of the code, amount of testing and the documentation. Generally, the cost is expressed in the term of the staff effort since for the implementation of a product new persons should be hired, trained them, buy new resources, new tools. The cost in software development is related to the number of hours spent by the staff for the implementation of the product. The implementation cost is usually estimated by developing organization.

10) Time: This factor is the most critical factor in software development cycle. Since in every organization there is pressure to complete the product with in specified time. So, the time for every requirement is estimated and assigned the priority value accordingly. A higher value of time factor indicates that it takes higher staff hours to complete the requirements.

Reasoning: In software industry on every product there is constraint to complete a product with in time. Time in software developments is related with number of staff hours. The development time of requirements is influenced by many factors such as degree of parallelism in development, train the staff, need to develop support infrastructure. Time is directly related with the cost. The more is the time to develop the requirement the more is the cost to implement the requirement.

11) Penalty: It is the punishment imposed on organization if they are not able to deliver the complete product within budget in the specified time. Penalty is critical factor in development of a requirement. This factor shows the penalty associated with each requirement. The higher value of penalty shows that they incur a high penalty if failed to deliver the right product. Here a weight between 1 and 10 is assigned.

Reasoning: - In software development process it may be possible that a low priority requirement incurs high penalty if the developer fails to complete the requirement. Penalty factor is associated with each requirement. It also increases the quality of product. If a requirement is not fulfilled then it is possible to evaluate the penalty corresponding to that

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9s
DOI: https://doi.org/10.17762/ijritcc.v11i9s.7453
***Article Received**: 16 May 2023 **Revised:** 03 July 2023 **Accepted:** 31 July 2023*

requirement. High penalty value means high penalty of that requirement.

12) Traceability: Traceability is the factor when a requirement is traceable to its test cases or not.

Reasoning: If there are pre-prepared test cases available then it is very beneficial for the developer organization and if the test case is not available then test cases must be designed for testing the requirements. If there are set of test cases corresponding to the requirements then assign zero priority value to this factor.

For each requirement, based on these 12 factors a Requirement Prioritization factor value (RPFV) is calculated by using Formula 1.

$$\text{RPFV} = \sum_{j=1}^{n} (pfvalue_{ij} * pfweight_{j}) \text{ ---------(1)}$$

Here i represent number of requirements and j represents number of factors.

In Formula 1 the RPFV represents the prioritization factor value for a requirement which is the summation of the product of priority value of a factor and the project factor weight. pfvlaue represents the value for factor for ith requirements and pfweight represents the factor weight for the jth factor for a particular project.

By using the Formula 1, the weight prioritization factor RPFV for every requirement can be calculated. Table 2 shows the prioritization of four sample requirements based on the RPFV for each requirement. In Table 2, R2 has highest RPFV among all the requirements. So, prioritization orders of these requirements are R2, R3, R4, and R1.

TABLE 2. REQUIREMENTS PRIORITIZATION

Factor	**R1**	**R2**	**R3**	**R4**	**Weight factor**
Customer Assigned Priority	8	10	9	9	0.02
Developer Assigned Priority	8	9	9	8	.08
Requirements Volatility	3	0	0	2	0.1
Fault Proneness	0	0	0	0	0.15
Expected Faults	2	3	4	2	.10
Implementation Complexity	3	4	5	3	.10
Execution Frequency	5	10	9	6	.05
Traceability	0	0	0	0	.05
Show Stopper Requirements	0	9	6	0	.2
Penalty	1	4	3	3	.05
Time	3	6	5	4	.05
Cost	4	7	6	6	.05
RPFV	2.25	4.77	4.15	2.47	1.0

The value of the RPFV depends on the value of the pfvalue and the pfweight. The value of the RPFV will vary with a change in the factor weights and the factor value.

As shown in Table 2 weights to each factor are assigned by the stakeholders of the project. The factor weight is assigned by the developer for each factor. Total factor weight assigned by the developer to the all factors should not more than one. In this approach the developer can analyze the complexity of a requirement based on the factor weight assigned to that requirement.

B. Prioritization of Module

In the process of prioritization of module mapping between the chosen prioritized requirement and its corresponding modules are performed. If there is more than one module the modules are prioritized. The criteria for module prioritization are based on the cyclomatic complexity and non dc path. Higher the cyclomatic complexity and non dc path of the module higher is the priority of that module. The test cases of the higher priority module are prioritized first and execute. For each module a module prioritization value (MPV) is calculated by adding the cyclomatic complexity and the number of non-dc paths.

TABLE 3. MODULE PRIORITIZATION

Factors	**M1**	**M2**	**M3**	**M4**
Cyclomatic Complexity	8	4	4	5
Non-Dc path	7	5	6	3
MPV	15	9	10	8

Table 3 shows the prioritization of four sample modules since MPV for each module. The order of prioritization of modules on the basis of MPV is M1, M3, M2 and M4.

C. 3.3 Test Case Prioritization Process

The test case prioritization process is used to prioritize and schedule the test cases corresponding to prioritized modules. In this test case prioritization process, there are some practical weight factors. On the basis of these practical weight factors process of the test case prioritization is performed These factors are test Impact, test case

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9s
DOI: https://doi.org/10.17762/ijritcc.v11i9s.7453
***Article Received**: 16 May 2023 **Revised:** 03 July 2023 **Accepted:** 31 July 2023*

complexity, requirements coverage and the dependency of the test cases as discussed below.

1) Test Case Complexity: Complexity of test case shows that how difficult is a test case to execute. It shows how much efforts are required to execute the test case. After analyzing the complexity of test case, the value of this factor is assigned between the value 1 and 10.

2) Requirement Coverage: This factor shows that how many requirements are covered by executing the test case. This factor is scaled between the value from 1to 10. The higher value shows the maximum requirements being covered by the test case. Higher the number of requirements coverage higher the priority of the test case to be executed first.

3) Dependency: This factor shows the dependency of test case on some pre-requisites. It shows how many pre-requisites are required for each test case before the execution of the test case. The value of dependency factor is assigned between the values from 1 to 10.

4) Test Impact: Test impact is the most critical factor in test case prioritization. It shows the impact of test case on a system if it is not executed. So, this factor assesses the importance of the test cases. Here a value between the 1 and 10 is assigned.

After assigning the prioritize factor value to each factor as discussed above TCWP (Test case weight prioritization) is computed using Formula 2.

$$\text{TCWP} = \sum_{j=1}^{n} (\, fvalue_{ij} * fweight_{j}) \quad \text{----(2)}$$

Where TCWP is weight Prioritization for each test case calculated from the four factors. fvalue is value assigned to each test case, fweight is a weight assigned to each factor.

After calculating the weight of each test case. The test cases are ordered by TCWP such that maximum TCWP gives a test case the highest priority and executed it first.

Consider a set of four sample test cases TC1, TC2, TC3, and TC4 which are to be prioritized.

For these test cases TCWP is calculated by Formula 2 and are prioritized on the basis of the value of TCWP (See Table 4).

TABLE 4. TEST CASE PRIORITIZATION

S.No.	**Factors**	**TC1**	**TC2**	**TC3**	**TC4**	**Weight**
1	Test Impact	4	8	7	9	0.4
2	Test case Complexity	8	7	5	9	0.3
3	Requirement coverage	6	2	4	4	0.2
4	Dependency	7	6	6	8	0.1
	TCWP	5.90	6.30	5.70	7.90	1.0

Now the order of the test case for the execution is TC4, TC2, TC1, and TC3. If the TCWP of the two test cases are same then we pick randomly from these two test cases.

IV. RESULTS AND ANALYSIS OF PROPOSED HSTCP APPROACH

To analyze the effectiveness of A Hierarchical system test case prioritization technique based on requirements approach, it was applied to the income tax calculator software which is used to calculate the tax on the income [1]. The software consists of 1160 lines of code and has nine modules named, Income details non salaried, Income details salaried, Savings, Tax deductions, Male Tax, Female tax, Senior tax and generate tax. All types of bugs like critical, major, and medium and minor bugs were introduced intentionally so that testing can be performed on the software using proposed HSTCP approach. Income tax software is based on following requirements.

- Accept Personal detail (APD)
- Accept income detail (AID)
- Accept tax deduction (ATD)
- Accept Savings and Donatation details (ASD)
- Generate tax detail (GTD)

Now considering the twelve factors for requirements prioritization discussed in Section 3, the corresponding weight values for each requirement was calculated as shown in Table 5.
Based on computation of RPFV the requirements prioritized list of the requirements is GTD, ATD, AID, ASD and APD Now the requirements were mapped to their corresponding modules. The cyclomatic complexity, number of non dc paths and the number of test cases of the modules are shown in Table 6.
In the table cyclomatic complexity, non-DC paths and the number of test cases for testing of each module are shown. Here GTD requirement has the highest priority. There are four modules corresponding to this requirement. On the basis of the values of cyclomatic complexity and non dc paths, the MPV value for Tax module is more as compared to other three modules. So, the test cases of the tax module must be prioritized. Table 7 shows the values for different factors for six test cases and the weight assigned.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9s
DOI: https://doi.org/10.17762/ijritcc.v11i9s.7453
***Article Received**: 16 May 2023 **Revised:** 03 July 2023 **Accepted:** 31 July 2023*

TABLE 5. REQUIREMENTS PRIORITIZATION

Requirements → / Factors ↓	APD	AID	ATD	ASD	GTD	Weight factor
Customer Assigned Priority	8	10	9	9	10	0.02
Developer Assigned Priority	8	9	9	8	10	.08
Requirements Volatility	3	0	3	2	8	0.1
Fault Proneness	0	0	0	0	0	0.15
Expected Faults	2	3	4	2	3	.10
Implementation Complexity	3	4	5	3	6	.10
Execution frequency	5	10	9	6	10	.05
Traceability	0	0	0	0	0	.05
Show Stopper Requirements	0	9	8	0	10	.2
Penalty	1	4	6	3	8	.05
Time	3	6	7	4	6	.05
Cost	4	7	8	6	7	.05
RPFV	2.25	4.77	5.20	2.47	6.25	1.0

TABLE 6. MODULE PRIORITIZATION

Requirements	Module	C complexity	Non dc path	No. of test cases	MPV
APD	Main module			8	
AID	NON salary	8	7	4	15
	Salary	12	10	6	22
ATD	Deduction	16	17	10	33
ASD	Saving	8	5	4	13
GTD	Male Tax	4	0	4	4
	Female Tax	4	0	4	4
	Senior Tax	4	0	4	4
	Tax module	6	0	6	6

TABLE 7. TEST CASE PRIORITIZATION FOR TEST CASES OF TAX MODULE

S.No.	Factors	TC1	TC2	TC3	TC4	TC5	TC6	Weight
1	Test Impact	4	7	7	9	8	7	0.4
2	Test case Complexity	8	7	8	9	8	9	0.3
3	Requirement coverage	0	0	0	0	0	0	0.2
4	Dependency	2	2	2	2	2	2	0.1
	TCWP	4.2	5.1	5.4	6.5	4.8	5.7	1.0

1) *Experimented Results*

The Tables (Table 8 to Table 13) shows the number of the faults detected by the test cases of all prioritized requirements.

TABLE 8 . FAULT DETECTION IN GENERATE TAX DETAILS (GTD) REQUIREMENT

Test ID	Critical Fault	Major fault	Medium fault	Minor fault
1	1	1	0	1
2	0	1	0	1
3	0	1	0	1
4	0	3	0	1
5	0	2	0	0
6	0	2	0	1

TABLE 9. FAULT DETECTION IN INCOME TAX DEDUCTION (ATD) REQUIREMENT

Test ID	Critical Fault	Major fault	Medium fault	Minor fault
1	0	1	0	1
2	0	0	0	1
3	0	0	0	0
4	0	0	1	0
5	0	0	1	0
6	0	0	1	0
7	0	0	1	1
8	0	1	1	0
9	0	0	0	0
10	0	2	4	3

TABLE 10. FAULT DETECTION IN ACCEPT SAVINGS AND DONATION DETAILS (ASD)

Test ID	Critical Fault	Major fault	Medium fault	Minor fault
1	0	1	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	1	1

INCOME DETAIL (AID)TABLE 11. FAULT DETECTION IN INCOME DETAIL MODULE OF ACCEPT

Test ID	Critical Fault	Major fault	Medium fault	Minor fault
1	0	1	0	3
2	0	0	0	1
3	0	0	0	1
4	0	1	0	1

TABLE 12. FAULT DETECTION IN INCOME DETAIL SALARIED MODULE OF ACCEPT INCOME DETAIL (AID) REQUIREMENT

Test ID	Critical Fault	Major fault	Medium fault	Minor fault
1	0	1	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	0	1	0	0
6	0	2	0	0

TABLE 13. FAULT DETECTION ACCEPT PERSONAL DETAIL (APD)

Test ID	Critical Fault	Major fault	Medium fault	Minor fault
1	0	1	1	0
2	0	1	0	0
3	0	0	0	0
4	0	0	0	0
5	0	0	0	1
6	0	0	0	0
7	0	0	0	1
8	0	0	0	0

Table 14 shows the total faults severity of each requirement. Faults severity is calculated using the Formula 3.

Fault severity = 4* no. of critical bugs+ 3* no of major bugs+2* no of medium bugs+1* no of minor bugs -------------------------------------(3)

TABLE 14. NUMBER AND TYPE OF FAULTS DETECTED BY ALL REQUIREMENTS

Requirement	Critical Faults	Major Faults	Medium Faults	Minor Faults	Total Faults severity
GTD	1	10	0	5	39
ATD	0	4	9	6	36
AID	0	6	0	6	24
ASD	0	2	2	1	11
APD	0	2	1	2	11

The fault severity corresponding to various prioritized requirements using HSTCP approach is shown below in Figure 4.

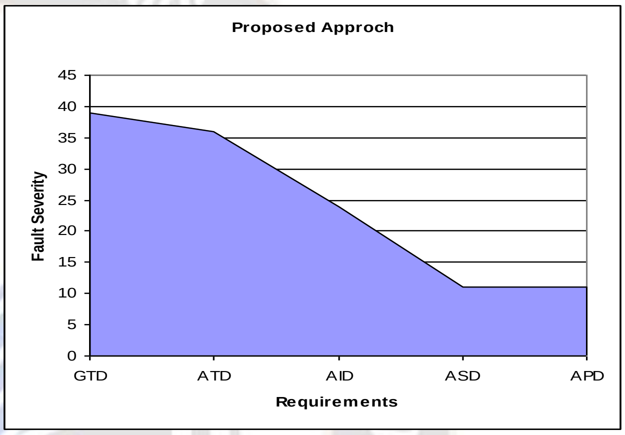


Figure 4. Graph for Proposed HSTCP approach based on requirements

A comparison of the proposed HSTCP approach was also performed with random as well as PORT [5] approach as shown in Figure 5 and Figure 6.

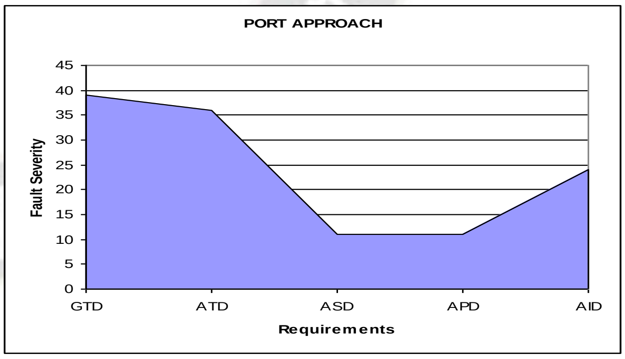


Figure 5: Graph obtained using PORT approach

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9s
DOI: https://doi.org/10.17762/ijritcc.v11i9s.7453
***Article Received**: 16 May 2023 **Revised:** 03 July 2023 **Accepted:** 31 July 2023*

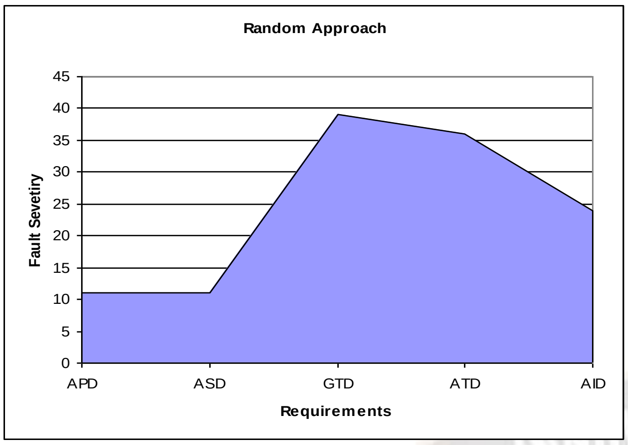


Figure 6: Graph for non – Prioritized test suite

$$APFD = 1 - \frac{TF1+TF2+\ldots\ldots\ldots\ldots\ldots+TFm}{n*m} + \frac{1}{2n}$$

---(4)

where n is the number of faults and m is the number of requirements

By using the formula 4 the APFD for the all approaches of test case prioritization were calculated as given below.

Table 15 shows the faults detected by various prioritized requirements by HSTCP approach.

TABLE 15. PRIORITIZED REQUIREMENTS OBTAINED BY HSTCP APPROACH

Type of Faults	GTD	ATD	AID	ASD	APD
Critical	1	0	0	0	0
Major	10	4	6	2	2
Medium	0	9	0	2	1
Minor	5	6	6	1	2

APFD For proposed HSTCP approach:

$$APFD = 1 - \frac{(16+38+36+20+25)}{57*5} + \frac{1}{2*5}$$

APFD= 1-135/285+1/10

APFD= 63%

Table 16 shows the faults detected by various prioritized requirements by PORT approach.

TABLE 16. PRIORITIZED REQUIREMENTS OBTAINED BY PORT APPROACH

Type of Faults	GTD	ATD	ASD	APD	AID
Critical	1	0	0	0	0
Major	10	4	2	2	6
Medium	0	9	2	1	0
Minor	5	6	1	2	6

APFD For PORT Approach:

$$APFD = 1 - \frac{(16+38+15+20+60)}{57*5} + \frac{1}{2*5}$$

APFD= 1-149/285+1/10

APFD= 58%

Table 17 shows the faults detected by various prioritized requirements by Random approach.

TABLE 17. PRIORITIZED REQUIREMENTS OBTAINED BY RANDOM APPROACH

Type of Faults	APD	ASD	GTD	ATD	AID
Critical	0	0	1	0	0
Major	2	2	10	4	6
Medium	1	2	0	9	0
Minor	2	1	5	6	6

APFD for Random Approach:

$$APFD = 1 - \frac{(05+10+48+76+60)}{57*5} + \frac{1}{2*5}$$

APFD= 1-199/285+1/10

APFD= 41%

2) *Analysis of Proposed HSTCP Approach*

The comparison is drawn between proposed approach, non – prioritized and PORT approach. It indicates that value obtained for proposed approach is more than the previous methods, thereby showing the efficacy of prioritized method. In this way the proposed Hierarchical system test case prioritization technique (HSTCP) based on requirements approach proves to be more effective as compared to other two approaches as shown in Figure 7.

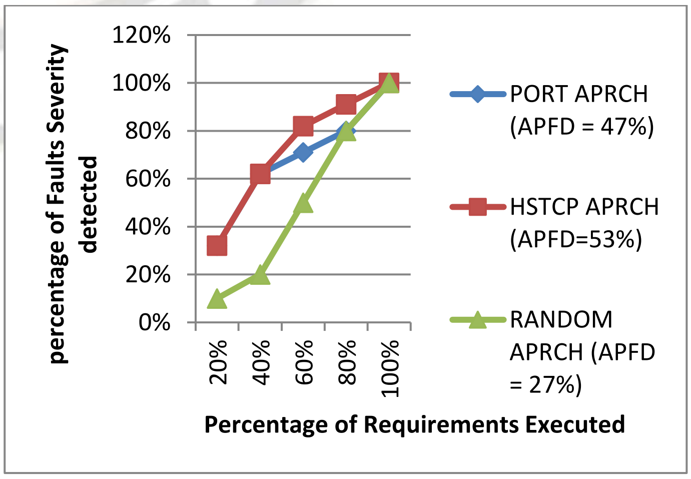


Figure 7 Comparison between Random, PORT, and Proposed HSTCP approach

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9s
DOI: https://doi.org/10.17762/ijritcc.v11i9s.7453
***Article Received**: 16 May 2023 **Revised:** 03 July 2023 **Accepted:** 31 July 2023*

V. IMPLEMENTATION

To implement the proposed approach a tool known as Hierarchical Test Case Prioritization (HSTCP) has been developed in JAVA language. This tool will help in prioritizing the requirements and further the modules and test cases in hierarchical manner. Using this tool, the tester is able to execute the test cases in highly prioritized order, so that test cases may detect critical bugs earlier. Some of the snapshots of the tool developed are shown in Figures 8 to Figure 13.

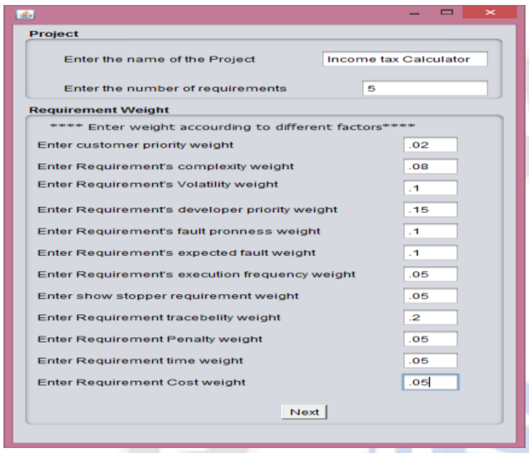

Project
Enter the name of the Project: Income tax Calculator
Enter the number of requirements: 5
Requirement Weight
**** Enter weight according to different factors****
Enter customer priority weight: .02
Enter Requirement's complexity weight: .08
Enter Requirement's Volatility weight: .1
Enter Requirement's developer priority weight: .15
Enter Requirement's fault pronness weight: .1
Enter Requirement's expected fault weight: .1
Enter Requirement's execution frequency weight: .05
Enter show stopper requirement weight: .05
Enter Requirement tracebelity weight: .2
Enter Requirement Penalty weight: .05
Enter Requirement time weight: .05
Enter Requirement Cost weight: .05

Figure 8.

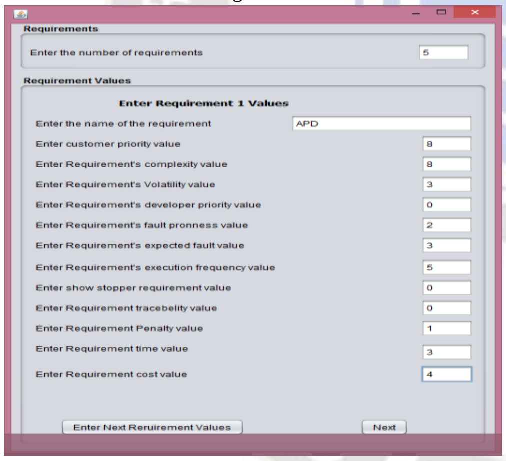

Requirements
Enter the number of requirements: 5
Requirement Values
Enter Requirement 1 Values
Enter the name of the requirement: APD
Enter customer priority value: 8
Enter Requirement's complexity value: 8
Enter Requirement's Volatility value: 3
Enter Requirement's developer priority value: 0
Enter Requirement's fault pronness value: 2
Enter Requirement's expected fault value: 3
Enter Requirement's execution frequency value: 5
Enter show stopper requirement value: 0
Enter Requirement tracebelity value: 0
Enter Requirement Penalty value: 1
Enter Requirement time value: 3
Enter Requirement cost value: 4

Figure 9.

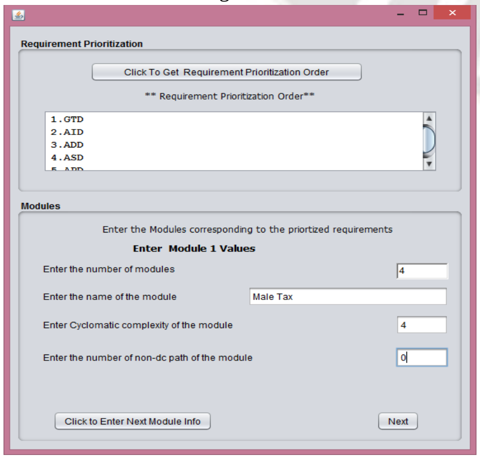
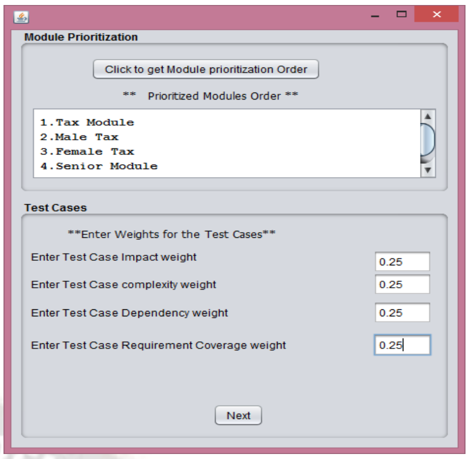

Requirement Prioritization
Click To Get Requirement Prioritization Order
** Requirement Prioritization Order**
```
1.GTD
2.AID
3.ADD
4.ASD
5.APD
```
Modules
Enter the Modules corresponding to the priortized requirements
Enter Module 1 Values
Enter the number of modules: 4
Enter the name of the module: Male Tax
Enter Cyclomatic complexity of the module: 4
Enter the number of non-dc path of the module: 0
Click to Enter Next Module Info
Next

Figure 10.


Module Prioritization

** Prioritized Modules Order **
```
1.Tax Module
2.Male Tax
3.Female Tax
4.Senior Module
```
Test Cases
Enter Weights for the Test Cases
Enter Test Case Impact weight: 0.25
Enter Test Case complexity weight: 0.25
Enter Test Case Dependency weight: 0.25
Enter Test Case Requirement Coverage weight: 0.25

Figure 11.

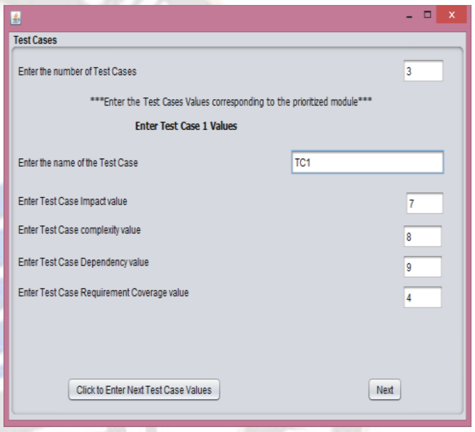
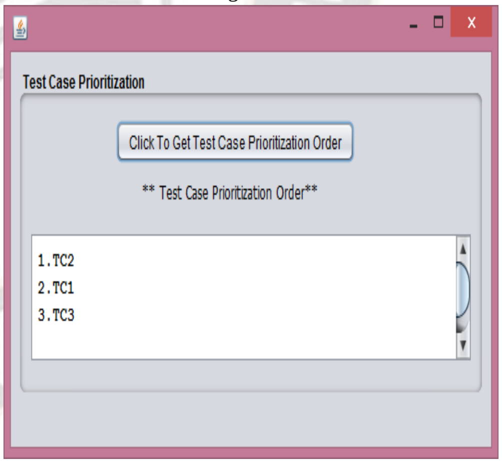

Test Cases
Enter the number of Test Cases: 3
Enter the Test Cases Values corresponding to the prioritized module
Enter Test Case 1 Values
Enter the name of the Test Case: TC1
Enter Test Case Impact value: 7
Enter Test Case complexity value: 8
Enter Test Case Dependency value: 9
Enter Test Case Requirement Coverage value: 4

Figure 12


Test Case Prioritization

** Test Case Prioritization Order**
```
1.TC2
2.TC1
3.TC3
```

Figure 13

VI. CONCLUSION

A hierarchical system test case prioritization technique has been presented in this research paper. The proposed technique maps the requirement to its corresponding design modules and further mapped to the corresponding test cases. This approach can be used to improve the rate of severe fault detection for system testing. An experimental study of income tax calculator software is presented for comparing the effectiveness of proposed approach with previous

***Article Received**: 16 May 2023 **Revised:** 03 July 2023 **Accepted:** 31 July 2023*

approach (PORT) and with random prioritization approach. The experimental results show that proposed new prioritization technique is promising in terms of ordering requirements so that faults are detected earlier in the testing phase. A tool is also developed for demonstrating the proposed HSTCP approach.

REFRENCES

[1] Dr. Naresh Chauhan**, Software** Testing - Principle and Practices, Oxford university press, 2010.
[2] Harish Kumar and Naresh Chauhan, Identifying and analyzing the research challenges in test case prioritization, journal of intelligent computing and application, Serial Publication,2012.
[3] H. Srikanth, L. Williams, J. Osborne, Towards the Prioritization of system test cases, North Carolina State University TR-2005-44, 2005
[4] R.Kavitha, Dr. N. Suresh Kumar , Factors oriented test case prioritization technique in Regression testing, European Journal of Scientific Research ISSN 1450-216X Vol.55 No.2(2011), pp.261-274.
[5] Berander Patrik, Andrews Anneliese, Requirements Prioritization. In: Aurum, Aybüke (Hrsg.);Wohlin, Claes (Hrsg.): Engineering and Managing Software Requirements. Berlin, Deutschland: Springer Verlag, 2005, S. 69-94
[6] Andrea Hermann, Maya Daneva, Requirement Prioritization based on Benefit and Cost Prediction: An Agenda of Future Research, 16th IEEE International Requirement Engineering Conference.
[7] Siripong Roongruangsuwan and Jirapun Daengdej, A Test Case Prioritization Method with Practical Weight Factors, Journal of Software Engg. 4(3): 193 - 214, 2010.
[8] Thillaikarasi Muthusamy, Seetharaman.K, A New Effective Test Case Prioritization for Regression Testing based on Prioritization Algorithm, International Journal of Applied Information Systems (IJAIS) – ISSN: 2249-0868 Foundation of Computer Science FCS, New York, USA Volume 6– No. 7, January 2014.
[9] M.Kalaiyarasan, Dr.H.Yasminroja , Version Specific Test Suite Prioritization using Dataflow Testing, International Journal of Recent Engineering Science (IJRES), ISSN:2349-7157, volume 1 issue 4 April,2014.
[10] Johanna Ahmad and Salmi Baharom, Factor Determination in Prioritizing Test Cases for Event Sequences: A Systematic Literature Review, Journal of Telecommunication, Electronic and Computer Engineering e-ISSN: 2289-8131 Vol. 10 No. 1-4
[11] Ms. Manaswini B, Rama Mohan Reddy A, A Cat Swarm Optimization Based Test Case Prioritization Technique to Perform Regression Testing, International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-8, Issue-1, May 2019
[12] Ms. Manaswini B, Rama Mohan Reddy A, A Shuffled Frog Leap Algorithm Based Test Case Prioritization Technique to perform Regression Testing, International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249-8958, Volume- 8 Issue-5, June 2019
[13] Remo Lachmann, Manuel Nieke, Christoph Seidl, Ina Schaefer, System-Level Test Case Prioritization Using Machine Learning, 2016 15th IEEE International Conference on Machine Learning and Applications
[14] Tyagi, R. ., K. Shastri, R. ., M., K. ., Ramkumar Prabhu, M. ., Laavanya, M. ., & C. Pawar, U. . (2023). Undecimated Wavelet Transform Technique for the Security Improvement In the Medical Images for the Atatck Prevention. International Journal of Intelligent Systems and Applications in Engineering, 11(3s), 211–217. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/2563
[15] G. Bhavyasri , A. AnandaRao, P. Radhika Raju, Enhancing the Performance of Coverage-Based Techniques in Test Case Prioritization, International Journal of Scientific Research in Computer Science, Engineering and Information Technology © 2017 IJSRCSEIT | Volume 2 | Issue 5 | ISSN : 2456-3307
[16] Rayapureddy Kalyani, Padmanabhuni Sai Mounika, Ravipati Naveen, Gnaneswari Maridu, Test Case Prioritization Using Requirements Clustering, International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 15 (2018) pp. 11776-11780 Research India Publications. http://www.ripublication.com
[17] Manoj Kumar Sahu, Aloka Natha, Test Case Prioritization for Regression Testing, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 3 Issue 8, August 2014
[18] Prof. Amruta Bijwar. (2016). Design and Analysis of High Speed Low Power Hybrid Adder Using Transmission Gates. International Journal of New Practices in Management and Engineering, 5(03), 07 - 12. Retrieved from http://ijnpme.org/index.php/IJNPME/article/view/46
[19] Zubair Rashid Bhat, Mudasir Ahmed Mutto , An Hybrid Approach of Test-Case Prioritization: Review , IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 3 Issue 8, August 2016 ISSN (Online) 2348 – 7968 | Impact Factor (2015) - 4.332”
[20] Song Wang, Jaechang Nam, Lin Tan, QTEP: Quality-Aware Test Case Prioritization, Foundations of Software Engineering, Paderborn, Germany, September 4–8, 2017 (ESEC/FSE’17).
[21] Vedpal and N. Chauhan, "Role of Machine Learning in Software Testing," 2021 5th International Conference on Information Systems and Computer Networks (ISCON), Mathura, India, 2021, pp. 1-5, doi: 10.1109/ISCON52037.2021.9702427.
[22] Rongqi Pan, Mojtaba Bagherzadeh, Taher A. Ghaleb, Lionel Briand “Test case selection and prioritization using machine learning: a systematic literature review” Empirical Software Engineering (2022)
[23] Cristina Maria tiutin, Andeer Vescan “Test case prioritization based on neural networks classification“AISTA 2022: Proceedings of the 2nd ACM International Workshop on AI and Software Testing/Analysis
[24] Jijo Joseph C George, D. Peter Augustine “Automation of Test

Case Prioritization: A Systematic Literature Review and Current Trends" Journal of Theoretical and Applied Information Technology, 15th February 2023. Vol.101. No 3 ISSN: 1992-8645

[25] Taylor, D., Roberts, R., Rodriguez, A., González, M., & Pérez, L. Efficient Course Scheduling in Engineering Education using Machine Learning. Kuwait Journal of Machine Learning, 1(2). Retrieved from http://kuwaitjournals.com/index.php/kjml/article/view/121

[26] Ali Samad, Hairulnizam Bin Mahdin, Rafaqat Kazmi, Rosziati Ibrahim, Zirawani Baharum "Multiobjective Test Case Prioritization Using Test Case Effectiveness: Multicriteria Scoring Method" Hindawi Scientific Programming Volume 2021, Article ID 9988987

[27] Elinda Kajo Mece, Hakik Paci Kleona Binjaku "The Application of Machine Learning In Test Case Prioritization - A Review" EJECE, European Journal ofElectrical and Computer Engineering Vol. 4, No. 1, January2020

[28] J. Michael Spector, M. David Merrill, Jan Elen, Handbook of Research on Educational Communications and Technology, Springer Publisher, 2013.

[29] Luca Ferrari, Deep Learning Techniques for Natural Language Translation , Machine Learning Applications Conference Proceedings, Vol 2 2022.

[30] Jean Arlat et.al, Comparison of Physical and Software Implemented Fault Injection Techniques", IEEE Transactions on computers, VOL. 52, NO. 9, September 2003.

[31] A K Munns and B F Bjeirmi, the role of project management in achieving project success, International Journal of Project Management Vol. 14, No. 2, pp. 81-87, 1996.

[32] Michael Bloch, Sven Blumberg & Jurgen Laartz, Delivering large-scale IT projects on time, on budget, and on value, October ,2012.

[33] K.K. Aggarwal, Yogesh Singh, Software Engineering, New Age International (P) Ltd., 2001.