Chapter

The Fundamental of TCP Techniques

Pritee Nivrutti Hulule

Abstract

Strategies for prioritizing test cases plan test cases to reduce the cost of retrospective testing and to enhance a specific objective function. Test cases are prioritized as those most important test cases under certain conditions are made before the re-examination process. There are many strategies available in the literature that focus on achieving various pre-test testing objectives and thus reduce their cost. In addition, inspectors often select a few well-known strategies for prioritizing trial cases. The main reason behind the lack of guidelines for the selection of TCP strategies. Therefore, this part of the study introduces the novel approach to TCP strategic planning using the ambiguous concept to support the effective selection of experimental strategies to prioritize experimental cases. This function is an extension of the already selected selection schemes for the prioritization of probation cases.

Keywords: test case prioritization (TCP), final total effort (FTE), average effort (AE)

1. Introduction

In testing, part of Regression testing in the maintenance phase is the process of retesting the updated software to ensure that new errors have not been introduced into earlier validated code. In addition, the regression tests should take as little time as possible to perform a few test cases as possible. Due to its costly nature, several techniques in the literature focus on costs.

These are:

i. Re-run everything

ii. Minimization/reduction of the test case

iii. Selection of the test case

iv. Prioritization of the test case

v. Hybrid approach.

This document focuses on the techniques of prioritization of the test case. Testers may now want to increase code coverage in test software at a faster pace, increase or improve their reliability in software reliability in less time, or increase the speed at which test suites detect failures at that moment. System during the regression tests. The main problems with code-based prioritization techniques are that they focus only on the number of errors detected and therefore treat all failures in the same way [1–4].

2. Test cases prioritization

Testing software or applications is the most important part of the "Software Development Life Cycle" (SDLC). It plays a very important role in the quality and performance of the software and ensures that the final product is as per the client's requirement. Placement Priority is the expansion of software testing, which is used to determine the "critical test cases". Software testing is done to detect bugs and errors in its operation, depending on how the performance and quality of the software are continuously improved.

These preliminary test cases are determined by a variety of factors depending on the need for the software, which is provided for test cases to perform other processes. By prioritizing test cases, (testers and developers can reduce the time and cost of the software testing phase and ensure that the product delivered is of different quality) [3–6].

2.1 What is test prioritization?

The term "test prioritization" refers to the accountable and difficult part of testing that allows assessors to "manage risk, plan tests, estimate cost, and analyze" which tests will work in the context of a particular project. This process is known as Test Case Prioritization, which is the process of 'prioritizing and planning' test cases. These methods are used to run test cases that are very important to reduce time, cost, and effort during the software testing phase.

In addition, the prioritization of the test case helps in regression testing and improves its effectiveness Developers of forensic trial software can get help to fix bugs earlier than possible otherwise. In addition, to determine the prioritization of test cases, different factors are determined according to the need of the software.

In this way, inspectors can easily run test cases, which have a very high value and provide errors with previous defects. Also, the improved error detection rate during the test phase allows for faster response of the test system.

The major issues of code-based prioritization techniques are that they focus only on the number of faults detected and hence treats all faults equally. In practice, all faults cannot be treated the same. Therefore, there may be situations where the presence of an error is less important but its coverage of the requirements is important. The prioritization of needs-based assessment addresses those issues by providing the most relevant case-based assessment of service-based services. In addition, Testing does not guarantee error-free software is a process of verifying software compared to user descriptions and their requirements major problem with the specifics of Test Case Prioritization is based on specificity and that there is no effective way to measure the performance of selected Test Suits [3].

Strategies for the promotion of probation cases make probationary cases subject to certain conditions. Prioritization of test cases can serve a variety of purposes.

The purpose of these priorities increases the likelihood that they will meet a particular goal more closely than they would otherwise have done at random. The Test Case Prioritization problem was officially described by Rothaermel and she learned nine TCP techniques. Among them, four relied on coding and two relied on early detection of errors. This was compared with no priorities and random prioritization strategies. The right category can only happen in books that are likely to 'benefit. Tests show an early detection of error with the greedy algorithm and additional greed in the code detection process. To measure the purpose of the experiment Rotermel defined metrics, the APFD rated an average of% of errors found over a fraction of the test suit (%) made. Its values are between 0 and 100 and the higher the value the better the error detection [3].

2.2 Intelligence techniques

2.2.1 Uncertainty and metaheuristics

The essence of this study is that it can be used in both the White-box test areas or the black box test environment. This approach aims to redesign test cases according to the extent of the alleged violation from the source code. The process of prioritizing probation cases using the absurd concept to improve the performance of a given assessment suit in violation of the evidence and further prioritization of probation cases. Anwar et al. conducted a study comparing various experimental precautionary measures and used the ambiguous notion of making good use of an experimental suit which is why the testing process backfired. Risk assessment of software needs is a major factor in improving software quality. Propose a new risk assessment approach using a sophisticated professional program to improve the effectiveness of TCP in the review process. All of these studies demonstrate the development of strategic priorities for testing. As time changes, the nature of these subjects changes. The proposed approach is a systematic study that minimizes theoretical aspects and drives research into a practical context. Although many strategies have been proposed many strategies are limited to code-based methods and focus on detecting a high number of errors. According to a study conducted by Catal (2013) on Test Case Prioritization Techniques, the highest number of strategies proposed so far is Coverage (code) based (40%) and minimal value is given to known costs (2%) and distribution-based (2%) strategies.

2.3 Benefits of proposed methodology

The major issues of code-based prioritization techniques are that they focus only on the number of faults detected and hence treats all faults equally. That is removed from the system.

There may be some cases where the existence of fault is not so important but its requirement coverage is that is performed in the proposed technique.

This part of the study is an extension of the selected selection schema and provides a Model of Priority Testing Priority Strategies based on three factors:

i. Service delivery,

ii. Efforts

iii. Complexity

In selecting the schema identification of project features/features that need to be done to identify TCP strategies that include high project attributes and therefore requirements. Selection Schema assumes that the required tools are the same. Once the strategy has been identified the next step is to differentiate those strategies based on high integration and a small experimental effort again, with difficulty. To calculate the effort this study uses the same basis as used by Krishnamoorthi. Therefore, the experimental effort represents the average number of test cases required for a specific functional evaluation process. Methodology plays a role in the evaluation of experimental efforts. According to research, the difficulty can be taken from a scale of 1–10 which is usually defined by the engineer and analyst. Therefore, this part of the study achieves high coverage with minimal difficulty and experimental efforts [2, 7].

3. Phases of TCP

Stage-1 uses the priority matrix proposed in 'the Selection Schema'.

Stage-2 experiences the difficulty of a particular method compared to the availability of its requirements.

TCP strategy tension is measured on a scale of (1–10) based on research. Calculating the effort of a particular method above number 2 is used. According to the formulas, the effort will come on a scale of (0–1). In this way, we will get used to multiplying by 10 so that we consider the same amount as coverage and weight. In the final stage, the classification is done with the Fuzzy Rule-based system.

The output of this program is sorted into the following upcoming sets:" Select, Medium and, Discard". Input variables, as well as output variables, can take values between (1–10). In this case study, triangular membership functions are used for mapping random and flexible input sets during fuzzification as well as for making dynamic output and complex sets during defuzzification. The input variables are written in three non-linear sets each: Low, Middle, and Top various purposes. The priority of the test case can be explained.

Given: In the test, suit provided S of the given system (X; XS, set of S) permissions;

f, function from XS to real numbers. Problem: Get S'є XS to (6S ") (S"єXS) (S's's ') [f (S') ≥ f (S ")].

In this definition, 'XS' is a collection of existing combinations to prioritize test cases for test 'S', and f is an objective function. For example, testers may wish to increase code coverage in software under test at a faster rate, increase or improve their confidence in software reliability in the short term, or increase the rate at which test suits find errors in that system during deferred testing.

The Test case Prioritization problem was officially described by Rothermel and he learned the nine TCP techniques discussed. Four of them were based on coding and two were based on the early detection rate. This was compared with no priorities and random prioritization strategies. The appropriate category is only possible in the text that is almost impossible to achieve. Tests show an early detection of error with the greedy algorithm and additional greed in the code detection process. To measure the purpose of the experiment Rothermel explained the metric, the APFD measuring an average of % of errors found over a fraction of the test suit (%) made. Its values are between 0 and 100 and the higher the value the better the error detection.

Think of a test suit T with several test cases; F is a set.

of m faults detected with test T T. TFi is the first test case in T'(one of T's orders) indicating error i. Thereafter T "s APFD is defined by the following equation [1, 4, 5, 8, 9]:

$$Average\ Percentage\ of\ Fault\ Detected = 1 - \left(\frac{TF1 + TF2 + \cdots TFm}{nm}\right) + \left(\frac{1}{2n}\right) \quad (1)$$

4. Methods

This proposed law-based program has a total of "17 rules" and is reviewed frequently based on expert knowledge. These rules are based on the following experts.

4.1 Motivation and contribution

In previous research, the researcher focuses on factors that I use, but they are not focused on the most important part 'time'. I work on that, and also how many testers are using the system, that tester priority is" low, medium or high" that things also captured and it generates the graph on basis of that.

4.2 Research findings

1. With the selection of any process the most important factor is the installation of project signs and therefore the installation of requirements.

2. Efforts are determined and calculated once the requirements have been met. Therefore, strategies should be chosen with "high coverage and few attempts" to achieve those features.

3. Complexity also plays an important role because only complexity determines the effort of a particular method (**Table 1**).

4.3 Method analysis

There Are Four Factors:

i. Requirement coverage

ii. Efforts

iii. Complexity

iv. Time

4.3.1 Input (three inputs)

1. Relevance of selected TCP Techniques based on maximum requirement coverage.

Rule#	Rule hypothesis	Class (rule outcome)
1	If (rel = low && comp + high && effort = low)	Discard
2	If (rel = low && comp + high && effort = medium)	Discard
3	If (rel = low && comp = high & & effort + high)	Discard
4	If (rel = low && comp = high && effort = high)	Discard
5	If (rel = high && comp = medium && effort = medium)	Select
6	If (rel = high && comp = low && effort = medium)	Select
7	If (rel = high && comp = low && effort = low)	Select
8	If (rel = high && comp = medium && effort = low)	Select
9	If(rel = high && comp + high && effort = low)	Select
10	If (rel = medium && comp + high && effort = low)	Moderate
11	If (rel = medium && comp + high && effort = medium)	Moderate
12	If (rel = medium && comp + high && effort = high)	Discard
13	If (rel = medium && comp = high && effort = high)	Discard
14	If (rel = medium && comp + high && effort = medium)	Moderate
15	If (rel = low && comp = high && effort + medium)	Discard
16	If (rel == medium && comp == high & & effort == medium)	Discard
17	If (rel == high && comp == medium && effort == high)	Discard

Table 1.
Rule base for fuzzy based selection of TCP techniques.

2. The complexity of selected TCP techniques

3. Average Effort (AE)

4. Time to execute

Output: Final class: TCP Techniques [2].

4.3.2 Begin

1. Identify input variables (linguistic variables) i.e., "relevance, AE, Complexity" (initialization).

2. Mapping of fuzzy sets to input variables by constructing the membership functions (initialization).

3. Formation of rules to create the rule base (initialization).

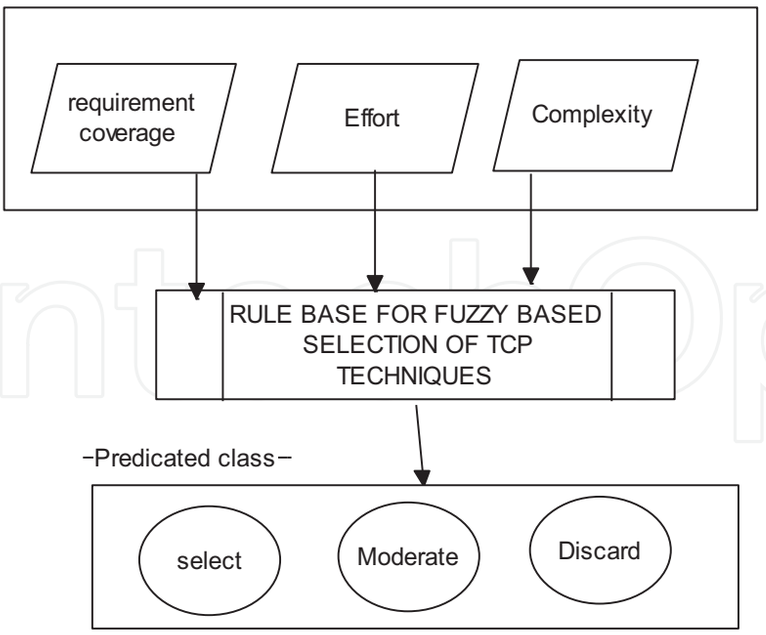
4. Conversion of input data (fuzzification).

5. Assessment of available rules in the rule base (inference).

6. Merge all the results achieved from available rules (inference).

7. Mapping of output data (defuzzification) [10].

4.3.3 Architecture



relevant project 'attributes/features' are done to identify TCP techniques covering maximum project attributes consequently requirements.

The various stages of the proposed approach are as follows:

Stage-1 Identifying project features in terms of relevance and hence the coverage of requirements.

Stage-2 Identify the complexity of testing techniques.

Stage-3 calculating testing effort.

Stage-4 classifies TCP techniques using fuzzy inference.

Stage 5. Time to execute each technique. Selection of any technique most important factor is time to perform an execution [2, 11].

4.4 Technical feasibility

Technical feasibility deals with the study of performance and numerous constraints like availableness of "resources, technology", the risk concerning development that might affect the capability to attain an adequate system. It identifies if the technology used is companionable or not with the recent system.

Following are some technical issues are:

- The system needed a large set of 'MySQL' database connectivity.
- This system technically supports different development tools like Eclipse, Net beans, etc.

4.5 Economic feasibility

Economic analysis is the generally used methodology for estimating the efficiency of a recent system. The procedure of cost analysis is to establish the advantages and savings that are looking ahead from a system and evaluate them with their costs.

Time-Based: On a single click management can create any report.

Cost-Based: There is no need for any type of training to use the software or tools. For managing this tool there is no need for any investment. The software is freeware so there is a minimal cost [2].

4.6 Operational feasibility

Operational feasibility is the activity of however well a projected system solves the issues. It receives the benefits of the opportunities known throughout the scope definition and the way it satisfies the needs known within the requirements analysis part of system implementation. If the user is aware of the all technicalities used for the system and the user having in detail knowledge about the system, then there are no difficulties at the time of implementing the system. Therefore, it's assumed that the user will not face any downside once handling the system implantation. Getting acceptance from users is the major difficulty for any developer at the time of developing any software tool. This system does not have any major problems. The small problem is that some time system gets slow due to the server and makes us wait for results. But it will automatically generate results fast. This is not a permanent problem of this system.

Case Studies for Development and Implementation:

1. With the selection of any process the most important factor is the installation of project signs and therefore the installation of requirements.

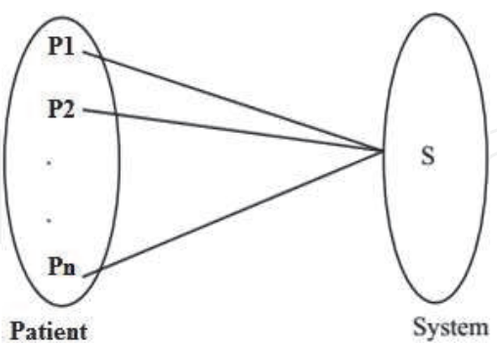
2. Efforts are determined and calculated once the requirements have been met. Therefore, strategies should be chosen with high coverage and few attempts to achieve those features.

3. Complexity also plays an important role because only complexity determines the effort of a particular method.

4. Time: Time to use TCP.

5. Mathematical model

A] Mapping Diagram.



Where,
P1, P2 … . Pn = Tester.
S = System.
B] **Set Theory.**
S = {s, e, X, Y, ϕ}.
Where,
s = Start of the program.

1. Authentication

L = Login, UN = User name, PWD = Password.

To access the facilities of system, TCP Classification.

X = Input of the program.

2. Identify INPUT as

Input should be 4 factors.

X = {S1, S2 … … Sn}.

Where,

S1, S2 … … Sn = No. of Factored selected by Patients.

3. Identify Process P as

P = {DF}.

Where,

DP = TCP Classification rules

4. Identify Output Y as

Y = {Bc1 … …Bcn}.

Where,

Bc1 … …Bcn = Select, discard, moderate class.

According to condition selected factor parameter TC will classify in one Class φ = Success or failure condition of the system [4].

6. Failures

1. A huge database can lead to more time-consuming to get the information.

2. Hardware failure.

3. Software failure.

7. Success

1. Search the required information from available rules.

2. The user gets results very fast according to their needs.

3. The mathematical model above is NP-Complete.

8. Our project is NP-complete

Our project goes into NP-Finish because at some point it will give the result. With the resolution problem, so that it will provide a solution to the problem during

the polynomial period. A collection of all decision-making problems the solution to which can be provided in the polynomial period. Functional requirement.

8.1 User

- Tester login to the system.
- Update profile
- Add tcp info for classification
- View predicted class of tcp with four parameters
    i. Requirement coverage,
    ii. Efforts
    iii. Complexity
    iv. Time. View own TCP.

8.2 Admin

- Admin will log in to the system.
- Admin activates tester.
- Admin can view all TCP Classification classes.
- Admin can view all TCP Classification graphs with 3 categorize.

9. Software quality attributes

9.1 Capacity

The capacity of a project according to data is very less.

9.2 Availability

All functionality will work properly.

9.3 Reliability

The system is reliable for classifying large data.

9.4 Security

User when login to the system that time users Mail Id and password match accurately.

9.5 DFD diagram

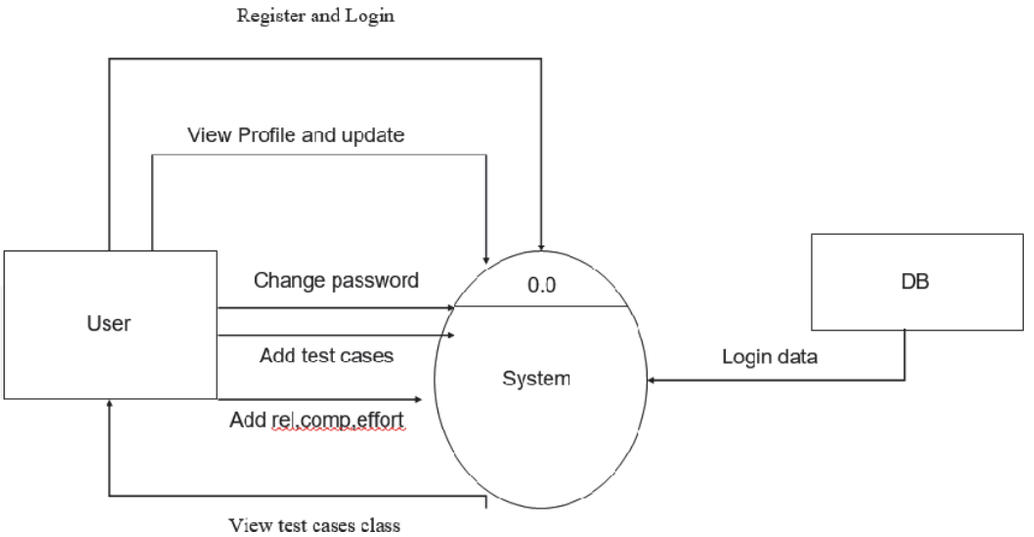


Fig-Data Flow Diagram.

System Use Case Diagram:

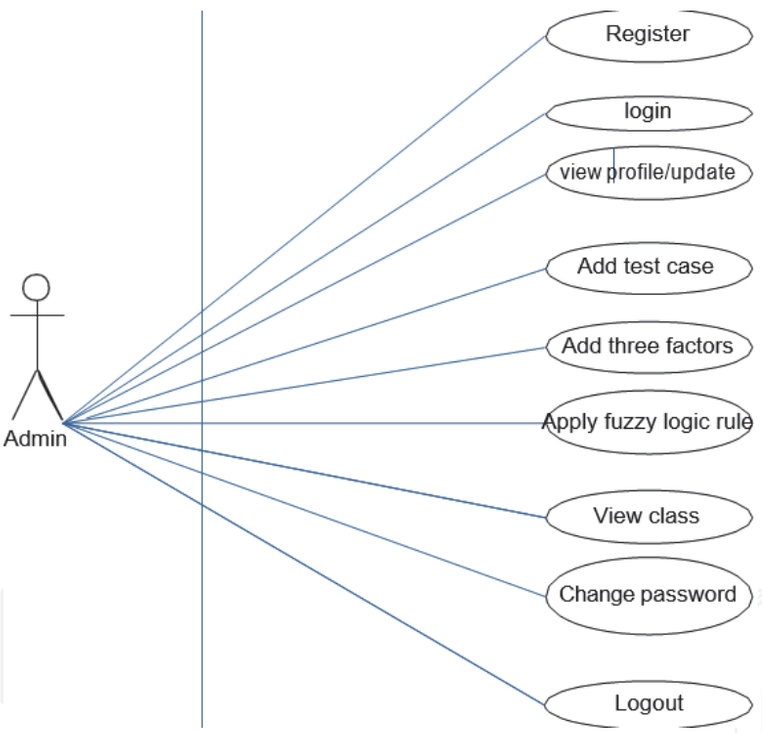


Fig-Use Case Diagram.

Class Diagram:

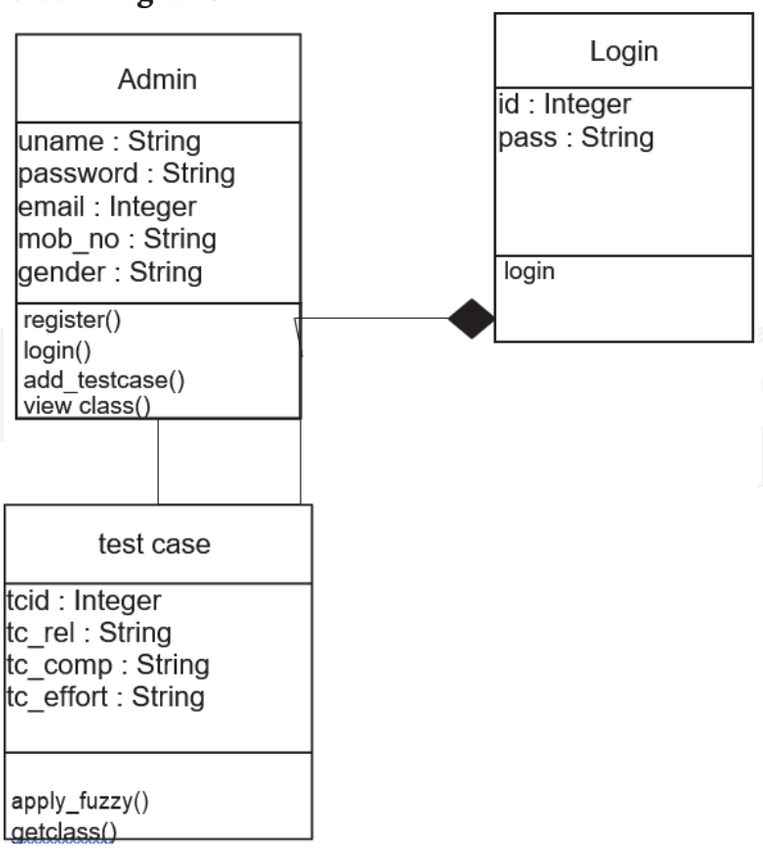
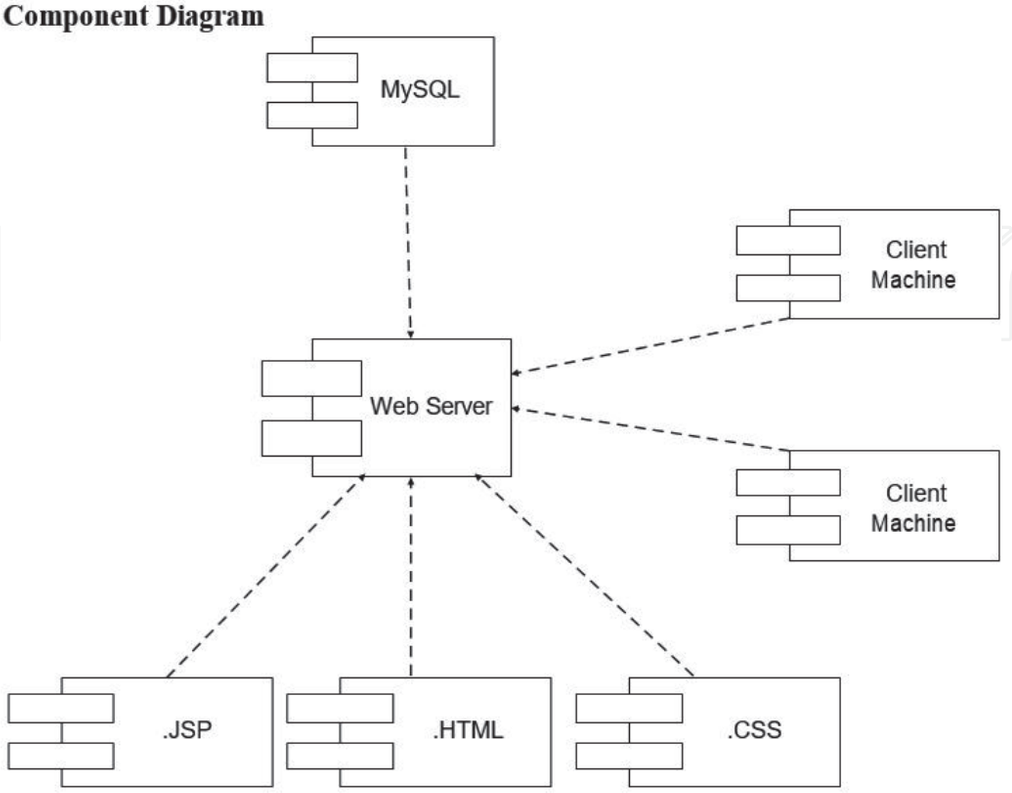


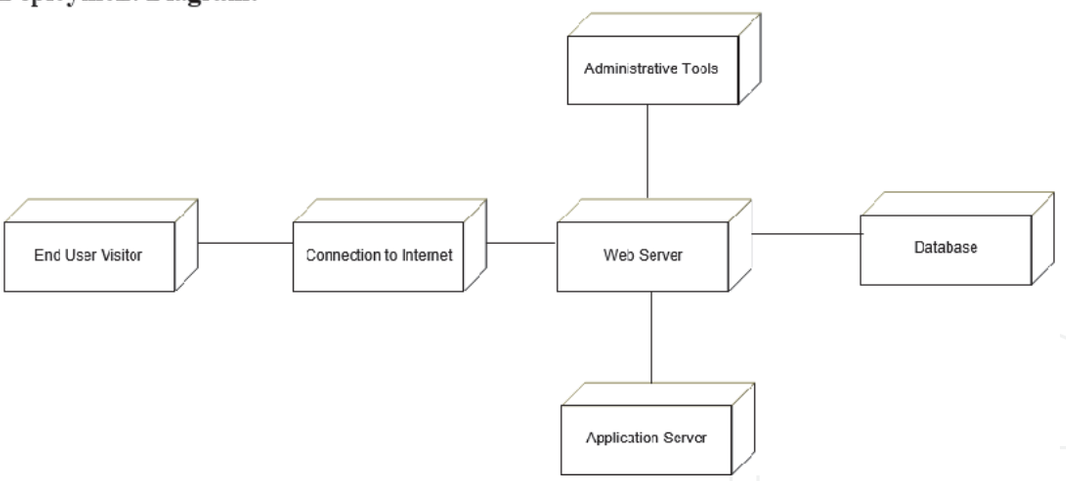
Fig-Class Diagram.

10. Architecture Modeling

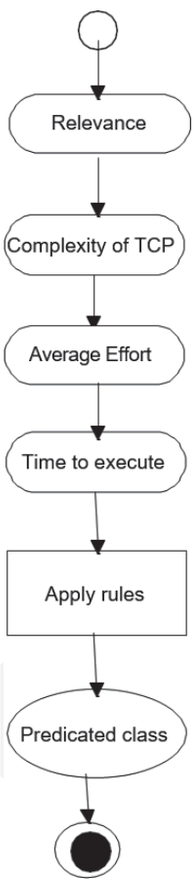
Component Diagram



Deployment Diagram:



11. The design process for quality software



11.1 Implementation approach

Describe the overall test method that will be used to evaluate the product of the project.

There are many ways such as:

- Black Box check
- White Box test

Here we have used the "Black Box test" method. In Black Box Testing we simply provide input to the system and test its output regardless of how the system processes it.

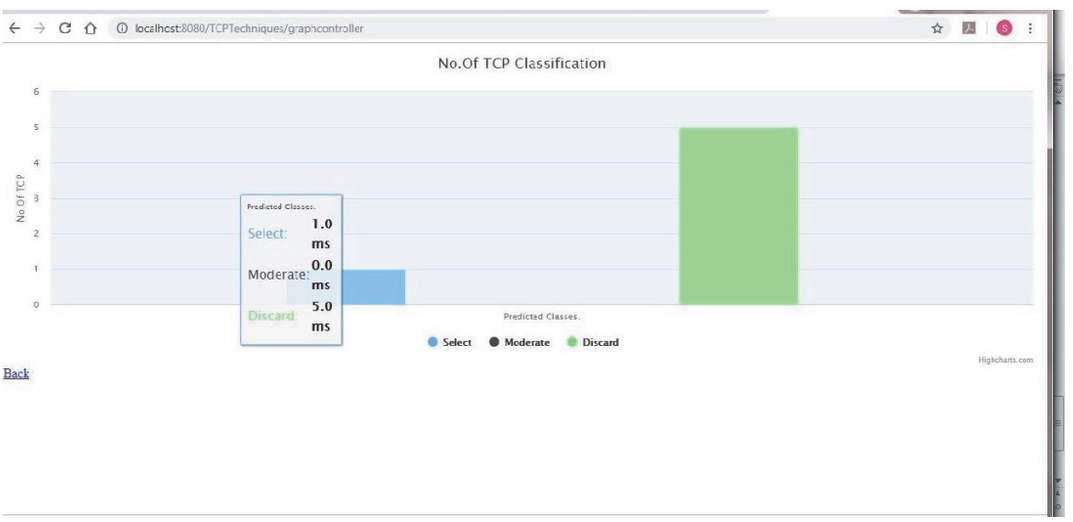
11.2 Passport test or test failure terms

Where the actual results and expectations are the same the test will be passed. When the actual results and expectations are different the test will fail.

Entry method: as soon as we have a need, we can start testing.

Exit method: when the disturbance level falls below a certain level, we can stop the test.

11.3 Implementation with screen output



Present the system in this research we have proposed a novel-based technique for the classification of TCP techniques using Fuzzy Logic. "This work is an extension of the already proposed selection schema for test case prioritization techniques". with the help of tester login and that tester priority of use the factor, it can generate the graph and also, it how much time required for the execution of the test case it calculates. Model for selection of test case prioritization technique based on 4f factors:

i. Requirement coverage

ii. Efforts

iii. Complexity

iv. Time

12. Future scope

"The system will work for other projects testing like ERP.TCP techniques will enhance performance by using another solution to prioritize test cases".

IntechOpen

Author details

Pritee Nivrutti Hulule
Computing Engineering, Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune, India

*Address all correspondence to: hululepritee9@gmail.com

References

[1] Duggal G, Suri B. Understanding Regression Testing Techniques. COIT; 2008

[2] Sujata KK, Purohit GN. A schema support for selection of test case prioritization techniques. In: Fifth International Conference on Advanced Computing & Communication Technologies (ACCT '15). 2015. pp. 547-551

[3] Vegas S, Basili V. A characterization schema for software testing techniques. Empirical Software Engineering. 2005; **10**(4):437-466

[4] Yoo S, Harman M. Regression testing minimization, selection, and prioritization: a survey. Software Testing, Verification, and Reliability. 2012;**22**:67-120

[5] Tyagi M, Malhotra S. An approach for test case prioritization based on three factors. International Journal of Information Technology and Computer Science. 2015;**4**:79-86

[6] Kumar V, Sujata K, Kumar M. Test case prioritization using fault severity. International Journal of Computer Science and Technology (IJCST). 2010; **1**(1):67-71

[7] Chen GY-H, Wang P-Q. Test case prioritization in specification-based environment. Journal of Software. 2014; **9**(8):205-2064

[8] Elbaum S, Malishevsky A, Rothermel G. Test case prioritization: a family of empirical studies. IEEE Transactions on Software Engineering. 2002;**28**:159-182

[9] Miranda B, Cruciani E. 2018 Copyright held by the owner/ author(s), FAST approaches to scalable similarity-based test case prioritization. 2018

[10] Silva D, Rabelo R, Campanhã M, Neto PS, Oliveira PA, Britto R. A hybrid approach for test case prioritization and selection. In: IEEE Congress on Evolutionary Computation (CEC). 2016. pp. 4508-4515

[11] Alakeel AM. Using fuzzy logic in test case prioritization for regression testing programs with assertions. The Scientific World Journal. 2014;**2014**: Article ID-316014