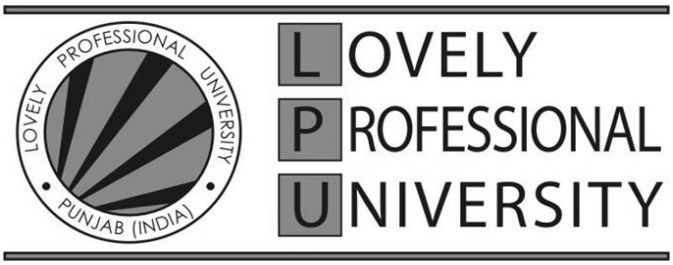

Hybrid Technique for Enhancement in Regression Testing

A Dissertation submitted

By

Neha Dwivedi

To

Department of Computer Science and Engineering

In partial fulfilment of the Requirement for the

Award of the Degree of

Master of Technology in Computer Science and Engineering

Under the guidance of

Er. Dalwinder Singh

(Assistant Professor)

May 2015

ABSTRACT

Regression Testing is the important and irreplaceable part of the test life cycle. Regression testing is executed whenever there is any modification taking place in the system. In order to validate the functionality of the system and to deliver quality product, regression testing plays an enriched role. Regression testing requires a lot of time and resource, hence in order to increase the efficiency of regression testing, test case prioritization technique is widely used. Taken into consideration the large and complex project, we have used the approach of the functional dependencies and the model based technique to prioritize the test cases as they provide the advantage of detection of the fault in the early phases of the development life cycle. In this hybrid technique, the model based approach provides an easy way to detect modification in the project, whereas the dependency approach is used to identify the dependent functions. Functional value of each function is calculated on the basis of its importance and the number of function it affects. Finally, test cases are prioritized according to the descending order of the total functional value.

CERTIFICATE

This is to certify that ***Ms. Neha Dwivedi*** has successfully completed ***M.Tech*** dissertation titled **"*Hybrid Technique for Enhancement in Regression Testing*"** under my guidance and supervision. To the best of my knowledge, the present work is the result of her original investigation and study. No part of the dissertation has ever been submitted for any other degree or diploma.

The dissertation is fit for the submission and the partial fulfilment of the conditions for the award of **M.Tech Computer Science & Engineering**

Signature of Advisor

Date: **Name: Er. Dalwinder Singh**

ACKNOWLEDGEMENTS

This work was carried out during the period of August 2014 to May 2015 at the **Department of Computer Science and Engineering, Lovely Professional University, Phagwara, Punjab.**

I would like, as a first to give my most sincere thanks to my supervisor, **Mr. Dalwinder Singh** for his guidance through all the steps of this work. I really appreciate his support and encouragement to keep me at the right track of doing things. More over without his help, I would not be able to complete my project.

I thank to all my friends, who were always ready to help me. All my friends encouraged me a lot.

I thank the marvellous and excellent work atmosphere at **Lovely Professional University, Phagwara, Punjab.**

I finally acknowledge **my parents and my family** who really supported me from time to time, took care and motivated me, even being at a distance.

Phagwara, 08/05/2015

Neha Dwivedi

DECLARATION

I hereby declare that the dissertation entitled **"*Hybrid Technique For Enhancement In Regression Testing*"** submitted for the **M.Tech** Degree is entirely my original work and all ideas and references have been duly acknowledged. It does not contain any work for the award of any other degree or diploma.

Date- 08/05/2015

Investigator- Neha Dwivedi

Registration No.-11300066

TABLE OF CONTENTS

Contents **Page No.**

LIST OF TABLE

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Software Testing

Software testing is the process of validating and verifying the program or application. It involves checking the proper functionality of the system. It also confirms that the project meets, the technical and business requirement as expected and can be implemented with the same characteristics. Testing procedure includes comparison of the expected outcome with the actual outcome, which helps in detecting the errors which may occur during the development of the project [3].

Any application must not result is severe failures, in order to ensure that, defects should be identified at the starting stage of the software testing. It can be very expensive in the future or in the later stages of the development. Hence testing ensures the quality of the product and this quality product when delivered to the customers helps in gaining their confidence.

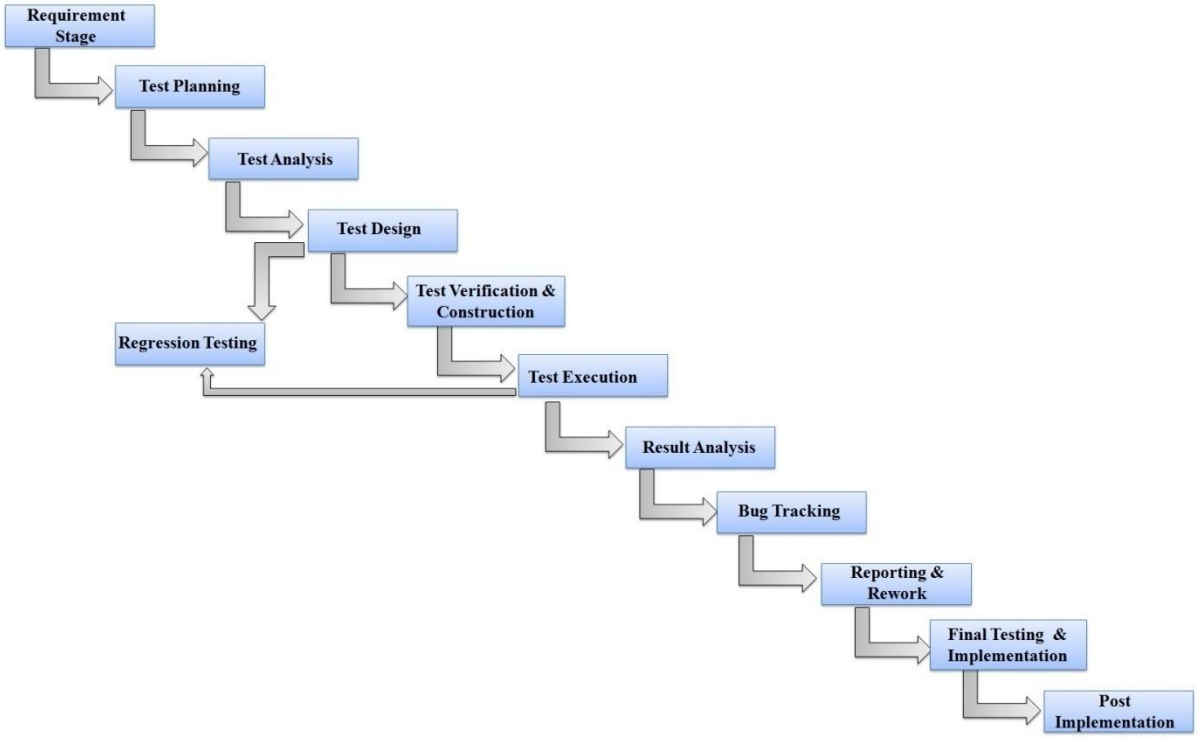

Figure 1.1: Software testing life cycle

Two important terms of testing are:

Validation: software validation is the dynamic process. This validation method is used to

ensure that software satisfies all the requirements, in order to come out with an excellent material. This validation is done during and end of the developmental stages of the software testing.

Verification: software verification is the static method. Verification tests those conditions which are given at the starting of the development phase, that has been satisfied or not.

1.2 Testing techniques classification

The testing can be classified into the following forms:

1. Black box testing and white box testing.
2. Manual testing and automatic testing.
3. Static testing and dynamic testing.

1.2.1 Black box testing and White box testing

1.2.1.1 Black box testing

Black box is a type of testing in which tester need not have any knowledge about the internal working of the software. It is also called as the functional testing. Tester draws test cases on the basis of the functional specification of the software. This type of testing does not require any knowledge of the implementation and it can be done along with the development.

Figure 1.2: Black-Box Testing

Types of Black box testing:

a. **Boundary value analysis :**

   In this type the different boundary condition are tested. Firstly the invalid and valid boundary is identified. On the basis of the boundary condition, the test cases are built and then the software is tested on those test cases. It is very easy to apply and provide good result in terms of the fault detection.

b. **Equivalence partitioning:** In equivalence partitioning, the software application is divided into various partitions. The division of these partitions is on the basis of the similarity of data usage, or the similar functionality. Then the valid and invalid test cases are made to test those conditions. The invalid condition should give wrong result and valid condition should provide correct result.

c. **State transition testing:** State transition testing is mainly conducted to test the transitional change in the system. Whenever any action is taken to transfer from one state to another, those conditions have to be tested and it also requires a graphical presentation.

1.2.1.2 White box testing

White box testing is also called as the structural or the code based testing. In this type of testing the tester should have knowledge of the internal working of the code. It is also called as the clear glass testing, as all the internal code, loops, the conditional branches of the code are known to the tester. Different test cases are tested to validate the code. It is done in the unit or the integration level.

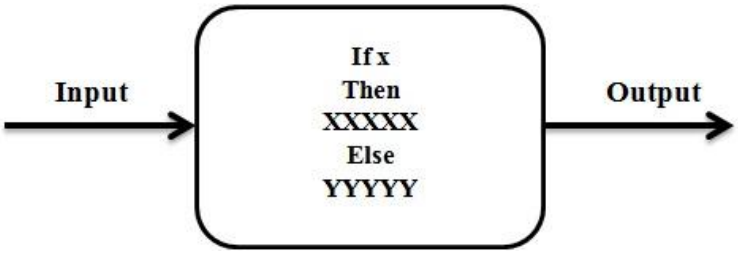

Figure 1.3: White- Box testing

The different type of the white box techniques are:

a. **Statement coverage:** in this type of testing the test cases are built so that it just covers all the statement of the code. This type of testing is done at the time of the unit testing. The main aim is to cover each and every statement of the code.
b. **Branch coverage:**
   In this testing the aim is to ensure that each one of the possible branch from each decision point is executed at least once and thereby ensuring that all reachable code is executed.
c. **Condition coverage:** the conditional testing is to test all the condition of the software. Test cases for testing both the valid and the invalid condition are made. Condition coverage is also known as Predicate Coverage in which each one of the Boolean expression have been evaluated to both TRUE and FALSE.
d. **Path coverage:** in this type of testing all the paths which leads to the final decisions are tested. In this testing all possible control paths are tested, including all loop paths. The test cases are prepared based on the logical complexity measure of a procedural design.

1.2.2 Automatic and manual testing

In manual testing the testing of the software is done manually that means without using any automatic tool or the test scripts. In this testing the tester acts as the end user and performs the testing to identify the bugs and unexpected results. In manual testing the test cases are generated and testing is done as per the test plan. The main drawbacks of the manual testing is that it is time consuming, less reliable, risky and can provide the incomplete coverage.

In automatic testing the testing is done with the help of automated tools and the test scripts. This tests the quality of the software completely. Automatic testing is more reliable, faster and requires less labor.

1.2.3 Static and dynamic testing

Static testing is also called as the specification testing. Static testing is done when the software is not in execution. This is used to test the requirement specification, design specification, coding and complete system specification. In this the tester can pretend like the

customer to check the specification of the system. Specification testing can also be used to check the various attributes of the system such as completeness, correctness, consistent, relevant, feasible and testable.

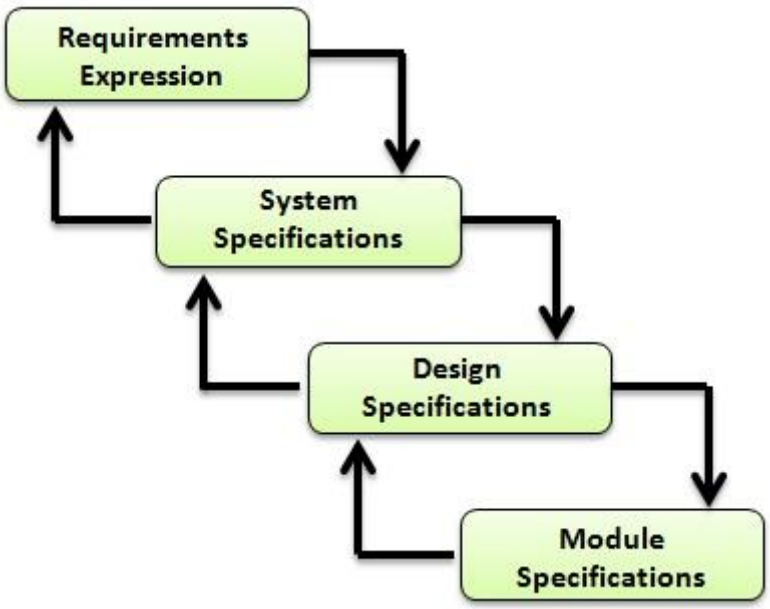

Figure 1.4: Using Specifications in Static Testing

Dynamic testing is used to test the software when it is in execution. In dynamic testing the software is tested to test that software is operating as expected and satisfies to work according to all the specifications necessary. The actual results of the test cases are compared with the expected result in order to resolve the problems in the software.

1.3 Approaches of Software Testing

There are two types of approaches which are followed by the software testing. These approaches are as following:

1. Top-Down Approach
2. Bottom-Up Approach

1.3.1 Bottom-Up Approach

Bottom–up is one of the approaches of software testing. It comes under integration testing. It starts with the unit modules such as programs and complete until the final phases are reached. The control flow moves to upward direction. Drivers which are used in the bottom up approach are the unit module [22]. In this approach, testing is conducted from sub module to main module. Drivers are used to simulate the main module. It is a temporary program and is also known inductive reasoning. The driver passes the appropriate data to the lower level module, at the each stage of bottom up integration. The units at the higher level are replaced by the driver. Figure 1.5 shows M5, M6 are the lower level module and M3 acts as the driver.

Advantages of Bottom-Up Approach:

The main advantages of Bottom-up Approach are as following:

1. Faults are occurring at the bottom of the module only.
2. Test results are easily observable.
3. Test cases are generated easily.

Disadvantages of Bottom-Up Approach

Disadvantages of bottom-up approach are as following:

1. Driver modules are required.
2. Only after the addition of last module, the program can exist.

1.3.2 Top-Down Approach

It is the second type of testing approach. In this approach we start from top level and move towards bottom level. The process module is then created and is broken down into sub modules. These sub modules are further tested and broken down into super sub modules. Stub is used in it as a temporary program.

Testing starts from the top and then proceeds to its child units. Other lower level nodes that may be connected should be created as a stub. The addition of lower level code, will replace stubs with actual components. This testing can be performed either by breadth first or depth first. Figure 1.6 shows that M5 and M6 are the lower level nodes.

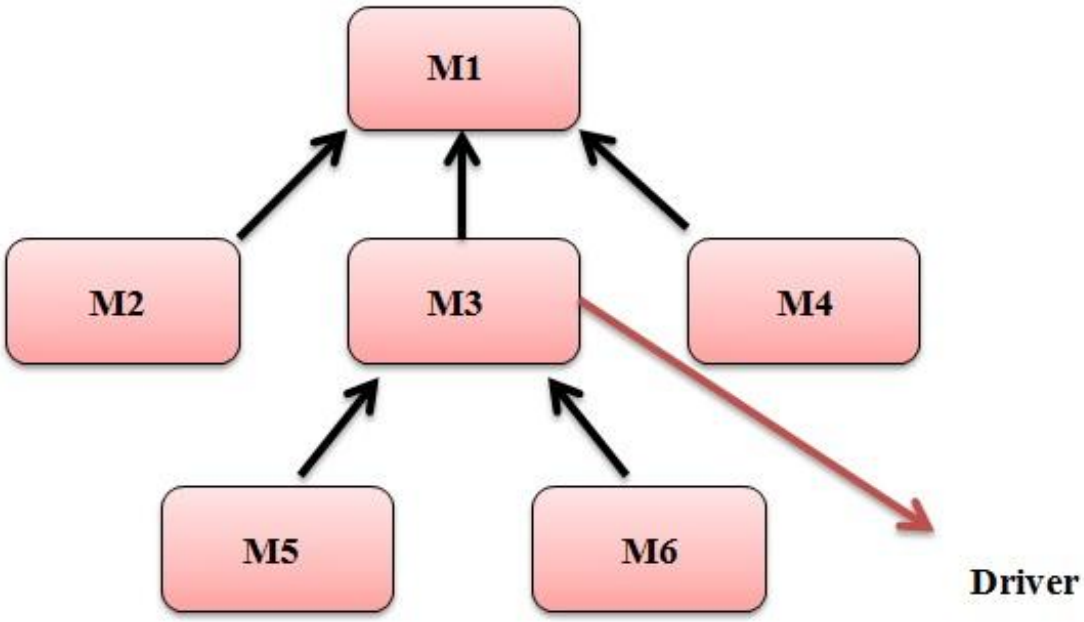

Figure 1.5: Bottom-Up Approach

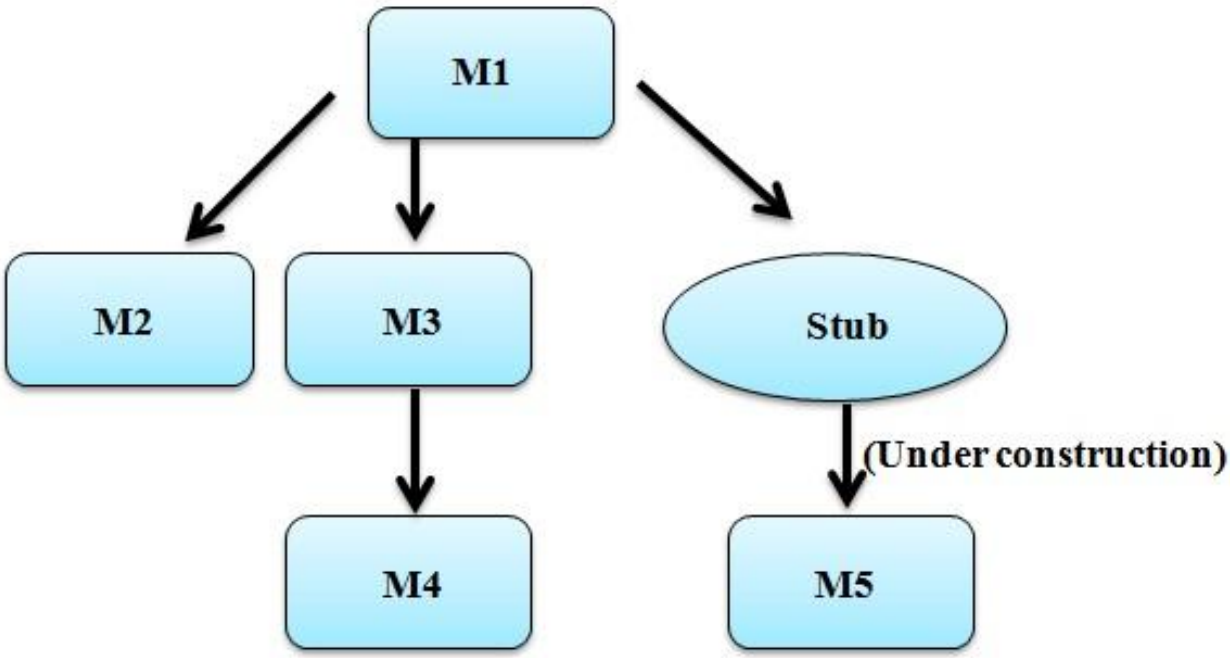

Figure 1.6: Top-Down Approach

Advantages of Top-Down Approach

The main advantages of top-down approach are as follows:

1. Test cases are easily created after the I/O functions are added.
2. If faults are found out at the early stage, then it is useful.

Disadvantages of Top-Down Approach

Disadvantages of top-Down Approach as follows:

1. Stub is complicated to produce at early stage.
2. Observation of test output is difficult to produce.
3. Testing and design are overlapped.

1.4 Testing in Software Maintenance

There are several phases in the software maintenance process. These phases are:

1. **Problem identification and modification testing**

   This is the first phase, in which the modification request is the entered.

   The request is send by the user, manager or customer to do the modification. In this phase it is decided that whether to do the modification or not.

2. **Analysis phase**

   In the analysis-phase, preliminary plan for design, implementation, testing and delivery will be built according to the conducted analysis. This analysis contains both the feasibility and the detailed analysis. It defines the requirement of the change and the test plan for testing.

3. **Design phase:** In the design phase strategy is designed to do the modification. For making these it requires the current document, current system and the output of the analysis phase. It identifies the modified part and plans the test strategy for the regression testing.

4. **Implementation phase:**

   This phase includes the implementation of the code of the modified function. It also conducts the unit testing, integration testing and the testing of the modified code.

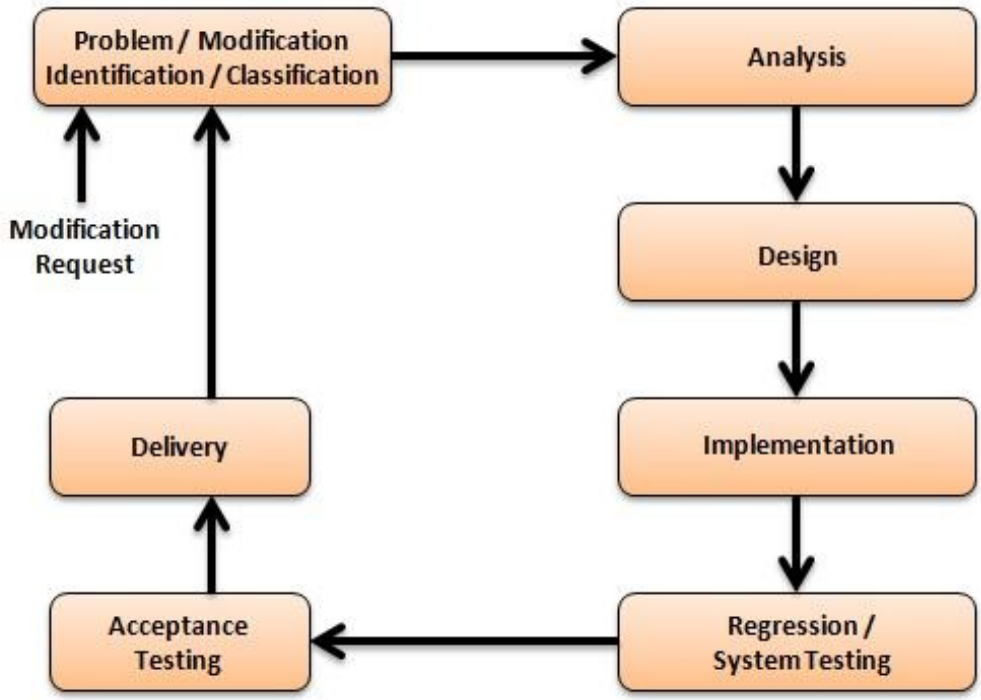

Figure 1.7: Software Maintenance Process

5. **Regression testing phase**

   Regression testing is that phase, in which the system is verified in order to ensure that due to the any modification in the code, the new faults have been added into the system or not.

6. **Acceptance testing**

   It is a type formal testing for the user needs, requirements, and economic processes that manage to determine whether or not a system satisfies the acceptance criteria. This is done to enable the user, customers or other authorized entity to determine whether or not to accept the system.

7. **Delivery**

   This is the last phase of the software maintenance phase, during which the new modified software is ready for the installation and the operation.

1.5 Types of testing

a) Unit testing: It is performed on a single unit or group of units that are somehow related to each other in any manner. It is performed by the programmer itself after the development of a unit or group of units. The model used in this testing is "white box model".

b) Integration testing: It is performed on combined components of product. Sometimes it happens that components work correctly before they are combined and produce errors when the combination of components takes place. So, it is also necessary to test the components after they are combined and is referred to as integration testing. The models used in this testing are "white box" and "black box" model.

c) Functional testing: It is performed to check the functionality of the system that whether the system is performing its function, what it is intended to perform. The model used in it is "black box" model. In this type of testing the input is given to each function and the output of the each function is examined. The internal structures of the functions are rarely considered.

d) **Acceptance testing:** In this type of testing, customer checks for output that whether the system is able to meet his requirements. Also, customer checks whether the purpose of the system is resolved or not. The model used in it is "black box" model.

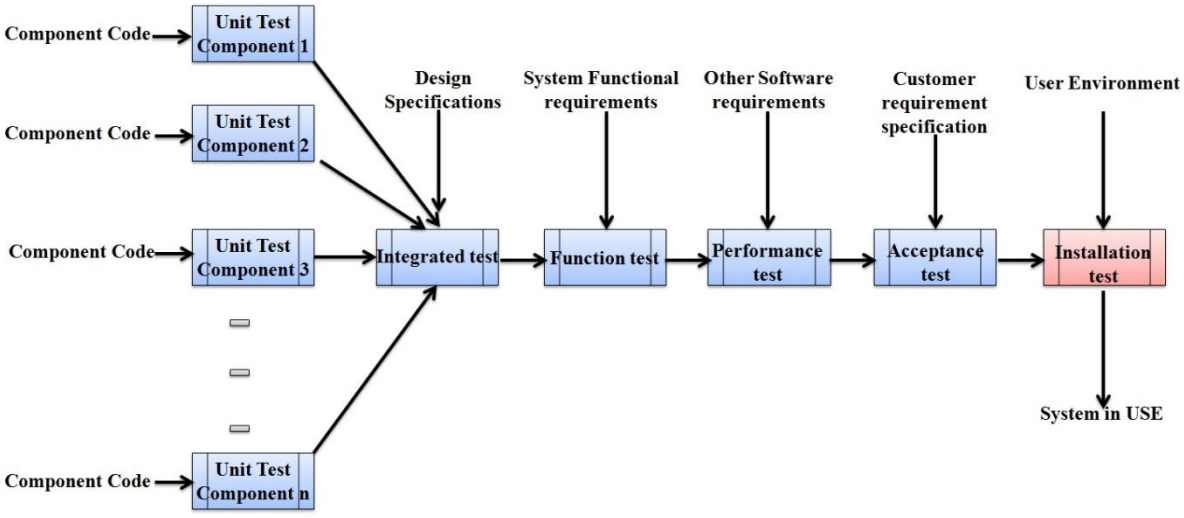

Figure 1.8: Types of testing

e) System testing: It is performed on a complete or full system and the software is put under different environments. It can only be performed after the complete implementation of the system. Stress testing, performance testing and usability testing are performed on it, in order to check the system: under stress (load), performance (speed and accuracy) and usability (user friendliness) respectively. The model used in this testing is "black box" model.

f) Regression (retest) testing: this type of testing is performed in order to validate that, whenever there is any modification or advancement made in any part of code, function or component it will not induce any defect in the functionality of the other parts of the software. [13] This process is very tedious, time consuming and costly affair but it is very important for improving the quality of the software. In order to increase the efficiency and reduce the cost, there are mainly three techniques used. Those are test case minimization, test case selection and test case prioritization. The model used in it is "black box" model. [3]

g) Alpha Testing: Alpha testing is done at the end of the whole process. Virtual type environment is created for this type of testing. It is in-house testing which do not require any type of test plan. It is done from the release of the product. It is uncontrolled type testing. Minor changes can be done in it.

h) Beta testing: It is performed by the outsiders and users. Sometimes the errors can arise due to some uncertain condition which is unexpected and can be felt by the user itself. The model used in it is "black box" model.

i) Usability Testing

It is user friendly application testing. In this type of testing new user can easily understand its functionality. Proper user help is provided in each level of testing. It provides effective functionality for the end users.

j) Load Testing

Load testing also comes under performance testing. It is done to determine whether the system is able to handle the given number of users or not. It is done only to check the

performance of the system i.e., is it working well or not. It checks the behavior under heavy loads and inputs. In web application testing, is used to check where the performance of the system degrade and at which point it fails.

k) Stress Testing

Stress testing is a performance testing which is done above than the normal volume and with the speed greater than the normal speed. It is applied on a system beyond the maximum design load. System is tested beyond the limit of required specifications and checked when it fails. It tells the behavior of the system with the increase of user cases.

l) Sanity Testing

It is a testing which is done at the initial phase to check whether the new version of software is good enough to perform the major testing later. If the software crashes at the earlier stage then further testing will not be able to perform over it.

m) Security Testing

It is also known as penetration testing. It is used to protect from the unauthorized users and hackers. It ensures that only authorized person can access the function of these systems. It is used to find out the weakness of the system, which further might arise and be a major harm for the system.

n) Reliability Testing

Reliability testing is used to find out the point where the system fails and remove it before it is deployed. Its aim is to restart system after verification from major crashes. Estimation Model is used to analyze the present reliability and predict the future reliability with the help of it. Decision will take place with the help of estimation model in reliability testing, whether the software will release or not by developer and whether it will adopt by users or not.

Essential elements of the testing are:

1. **Test Strategy:** It defines the amount of testing and types of testing which gives the best results to find the defects. It also defines which types of tests should be

conducted at each testing levels. It ensures the acceptable and cost limit of testing.

2. **Test Plan:** Test Plan is like a document which defines the test activities, scope, resources, approach and tasks etc. in a testing. It is a formal process. It helps to define the feature of testing; how testing can be done. It is a part of project tasks.
3. **Test Cases:** A test case has an input, an action performed and an expected result as an output. The main goal of test cases is to find out errors. A test case contains knowledge that includes the test case name, objective, background, steps to take, input and expected results. To check whether the application is working according to the given specification then its expected results are matched with the actual results
4. **Test Environment:** It allows the system to configure and install new products according to its implementation. It provides the right equipment for testing the software.

1.6 Regression testing

Regression testing is that part of the test life cycle where program are tested in order to find that whether the changes or the modification which are made in the program do not affect the function which are not supposed to be affected. In regression testing those test cases which are tested earlier are supposed to be tested again. Testing of test cases over and over again is very time consuming and involve lot of budget. Many companies have to pay lot of expenses on the testing of the code. Moreover if the quality of testing is not good it hampers the quality of product.

1.6.1 Importance of regression testing

Regression testing plays an important role when the developer, while fixing the bug is so concentrated about fixing that bug that he is not able to view the changes which are affected by that fixing it. Regression testing also checks for the locations which are faulty in the program. This helps in successful modification of the programs.

1.6.2 Techniques for regression testing:

- **Reset all:** in this technique all the test cases of the existing project are re executed to

verify that whether the modification affected those part or not, which are not to be affected. This is very time consuming and costly process.

- **Test case prioritization:** In this type of technique, the test cases are prioritized according to some criteria. The test cases are prioritized such that the rate of fault detection is higher, the cost and time spent should be economic. Various techniques which are used by the researchers are based on the requirements, cost of testing, past history of fault detection, functional importance of test cases and on the basis of the coverage of the code.
- **Regression test selection**: In this technique the test cases are selected which are needed to be tested and are necessary so that, the unnecessary cost and time can be avoided.
- **Hybrid Approach:** In this approach the combination of test case prioritization and the regression test case selection technique is used. Hybrid approach selects the test cases and then they are tested to the priority.

1.7 Model Based Testing

Model based testing is an automated type of black box testing approach. Model based testing provides an innovative and advance method for test case generation and test case evaluation. The test cases are built at the early stages of development based on the model. Model provides an actual overview about the requirement, specification and behavior of the system. Specific industry standard are there which governs the generation of the model that can be further enhanced by experience standard [14].

Model based testing is very flexible in nature, changes in the requirement can be simply implemented by updating the model and then quickly generating test suits. Unlike other conventional testing process in which editing of test suites is very tedious and error prone. Moreover model based testing reduces the cost, time and increases the effectiveness of the test life cycle.

Software models can be of two types:

a) Structural model: It defines different units and the static relationship that exists between those units. Example: Class diagram, Use case diagram.

b) Behavioral model: It defines different units and the dynamic relationship that exists between those units. Example: Sequence diagram, Activity diagram

1.7.1 There are five main steps of MBT process:

1. **Model the software under test (SUT) or its environment**. Construction of an abstract model is the first step of model based testing (MBT) process. It mainly focuses on the key aspect of the system to be tested based on the specified requirement. It should be very simple behavioral model, without much detail. [15]

2. Creating abstract test cases from the model:

Next step after the generation of the behavioral model is the abstract test case generation from that model. To complete this step it is very important to decide a criteria for test case selection because there can be infinite number of possible tests. For example in code coverage criterion, the model has to focus on to cover all transition in finite state machine. [15]

3. Execution of the abstract test cases after concretization of test cases: After the generation of the abstract test cases, the next step is the conversion of those abstract test cases in to the executable concrete tests. This process includes writing of adaptor code and its execution to maps each abstract operation to the lower level SUT interface. This helps to connect abstract test with the concrete SUT by adding details which are not included in the abstract model. The advantage of this two layer approach is that we can use any language for writing test cases in any environment. The test can be reused in different test execution environment by simply changing the adapter code [15].

4. Execution of the test cases and verdict assignment:

After the successful generation of the test script, it is the time for the execution of the tests. Tests are executed in either offline or online mode. In on line mode the test cases are executed as soon as they are produced and result are recorded in MBT tool. With offline mode firstly the whole test script is produced and then they are executed.[15]

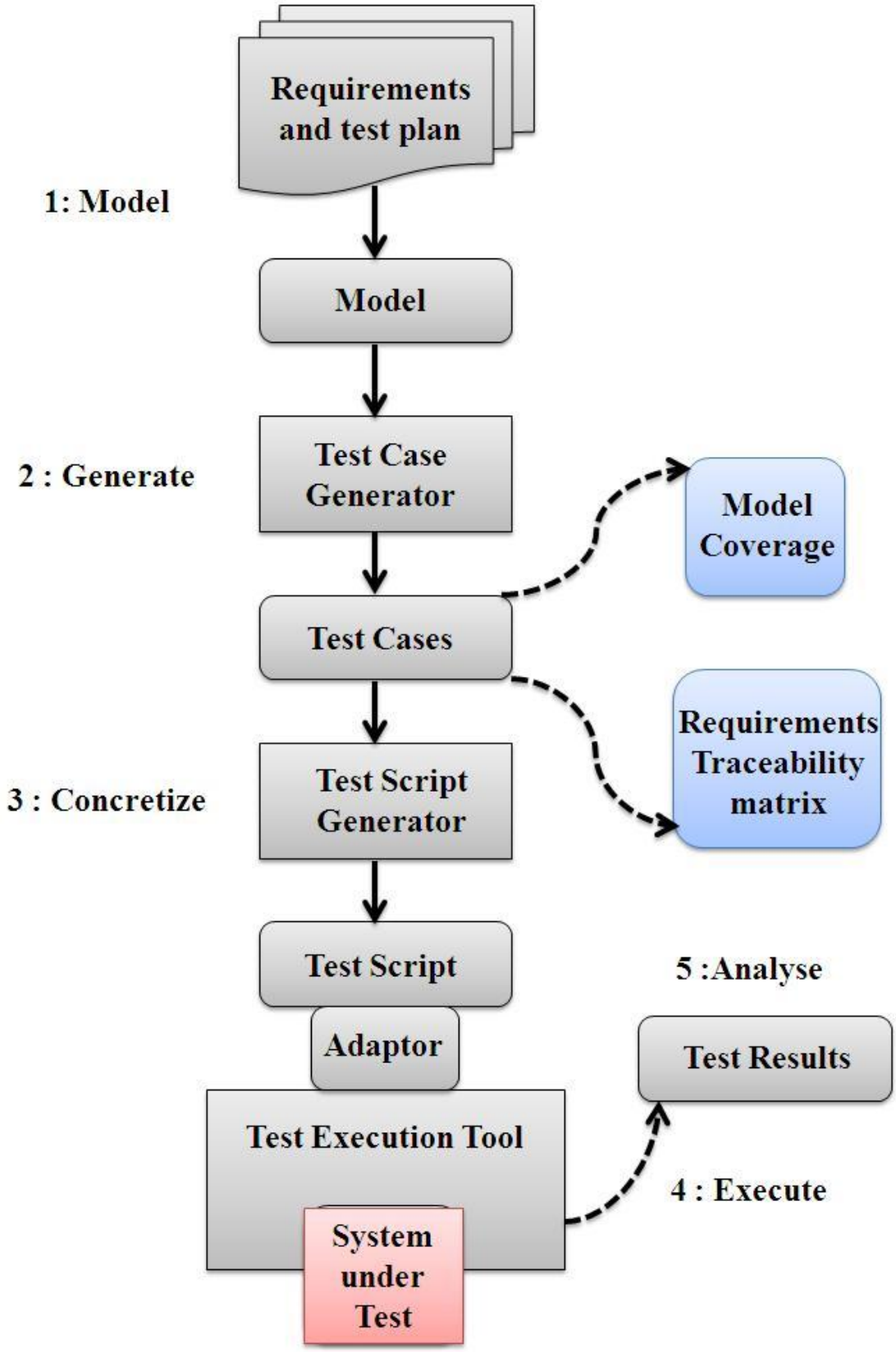

Figure 1.9: Model based testing process [15]

5. Analysis of the test results:

The last step of the MBT process is the analysis of the result and determination of the causes of each test case failure. For each test case, the cause of the failure and corrective steps are suggested. The failure of test case may occur due to the fault in software under test (SUT) or test case itself which occur due to the fault in behavior of the system or in the adaptor. Feedback about the validity of the whole model is given in this step. [5]

Advantages:

- **Reduction in cost:** It decreases the cost as well as effort, since the faults can be detected in early phases.
- **Reduction in time:** due to the automation process the time of testing has greatly reduced. Model based testing can find six times more faults as compared to other testing methods.
- **Documentation is easy:** Documents can be easily made, as understanding is clear through the models. These documents can further be used to train the new team members came in development and testing groups.
- **Assets can be reused:** The assets can be reused while regression testing is being performed. When any modifications are performed on the system, the model can be easily changed accordingly.
- **Good understanding:** It is easy to understand through models rather than theoretical description.

1.8 Test Case Prioritization

Test case prioritization is one of the efficient techniques of regression testing. It aims at scheduling the execution process of test cases in such a way, which improves the efficiency of regression testing. Whenever there is any modification in the software the regression testing has to be done in order to validate the functionality of the software, which is a costly affair and requires a lot of money and time. In order to reduce the burden of regression testing, prioritization of test cases is done. [6]

There can be infinite number of ways to prioritize the test cases. However it is impossible to consider all the different ways to prioritize test cases in a system. Hence we have to select

particular criteria or the combination of different criteria in accordance with the structural and behavior of the system.

Techniques of Test Case Prioritization:

1. Customer Requirement-Based Prioritization Techniques:

This type of test case prioritization technique mainly focuses on the requirement of the customer along with the system complexities. The test cases are prioritized on the basis of customer assigned priority. [4]

2. **Coverage-based techniques:** This type of testing technique is called structural or white box testing; this technique mainly focuses on the code of the system or the internal logic of the system. The priority is assigned on the basis of the code coverage of the test cases. Those test cases which cover more code area are given higher priority than the others, in order to cover all the logical operation in less time. [9]

3. **History-based techniques:** This type of testing technique considers the knowledge of history as the basis of prioritization. In this technique a traceability matrix is built which maps the number of fault detected to the tests cases. The test cases which detect higher number of fault are assigned higher priority in the list of test cases. [5]

4. **Cost Effective-based techniques:** This type of test prioritizing technique focuses on the cost required to perform regression testing. It also takes the resource utilization in to consideration. Those test cases which have high severity, requires less resources and are assigned higher priority. [2]

As we discussed earlier that the regression testing is a very important phase of the testing life cycle, which is very much in demand due to the rapid growth of the software industry. It delivers high quality product in less time with reduced cost and required functionality. Functional dependency is another criterion which can bring improvement in the test case prioritization and early detection of the faults. [11]

1.8.1 Dependency Approach

A large software product consists of large number of subsystem or components, which further contains programs to a specific task. In software engineering these subsystems or

components are called as scenarios. This scenario depicts the behavior or sequence of action which is performed by the system. These ordering of the scenario show the dependency and the interaction among the scenario. Action of one scenario is dependent on another scenario and that cannot be executed before the completion of the action of the dependent scenario. [22]

On the basis of scenarios, test cases are generated. There are mainly two types of test cases:

1. Independent test cases: these are the type of test case which is executed independently without the concern of the other test cases.

2. Dependent test cases: are those types of test cases whose execution cannot occur freely without concerning the dependent test cases. The execution of test case is only possible when it had finished the execution of the all the dependent test cases.

The diagram in Figure 1.10 shows that C_1 and C_2 are two independent components. S3 to S10 are dependent subcomponent. S3 is dependent on C_1 directly and S6 is indirectly dependent on C_1. On the basis of the direct and indirect dependencies there are two types of dependency structure. One is open dependency structure and other is closed dependency structure. This shows the dependency structure of the test cases and helps in assigning the priority to the test cases.

1. **Open dependency structure:** In this type of dependency structure, the dependency

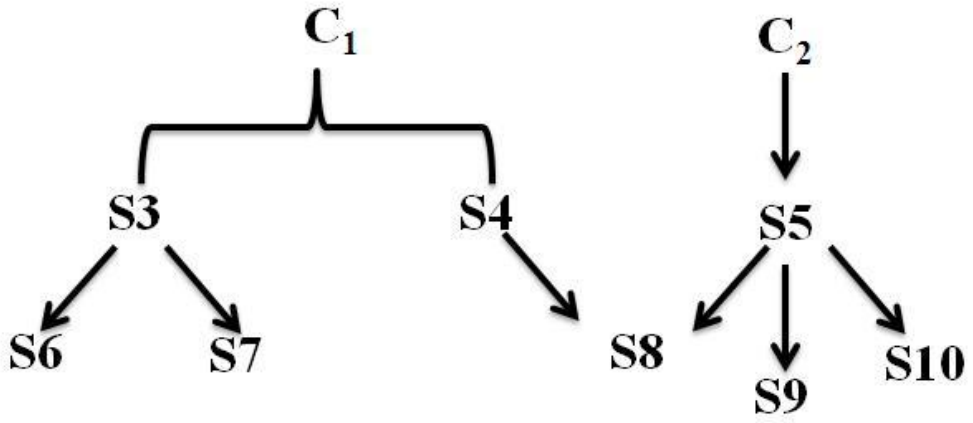

Figure 1.10: An example of Dependency structure. [22].

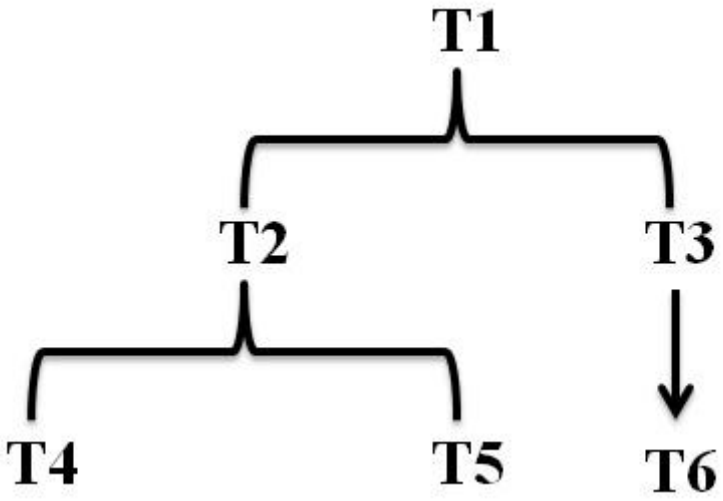

Figure 1.11: Dependency structure of the test cases [22]

between the test cases is such that the execution of one test case should take place before another anywhere in the program. Not necessarily immediately before that. For example as shown in the Fig 1.10 after the execution of the C_1, S3 and S4 remain open for execution and execute anywhere in the program.

2. **Closed dependency structure:** In this type of structure, execution of one test case should be executed just before another test case. The dependency between them is not open. For example Fig 1.10 shows the closed dependencies where the node C_1 and C_2 are executed in order, then the execution of S3 and S4 take place, and node C_1 would have to be executed again.[22]

1.8.2 Prioritization of the Test Cases Using Dependency

Complexity of the large system is greater due to the large amount of coupling and the interaction between the parts of the system. For the early detection of the fault it is recommended to find out those parts of system, which have higher interaction. [22] Interaction among the system indicates the dependencies of the function in the system. Therefore prioritization of the test cases in the order of the weights of the dependencies will result into the early detection of the faults in the system. [7] In order to find the priority of the functional dependencies, a technique called as dependency structure prioritization (DSP) is used.

CHAPTER 2
LITERATURE REVIEW

Shifa-e-Zehra Haidry and Tim Miller [22], suggested that using the dependencies has conducted a study on prioritization. Their study suggested that, existence of large coupling makes the prioritization technique more efficient. Test-case prioritization is done based on dependencies, as the dependencies can be observed. This leads to the ordering of test cases. The test cases having more dependents will be executed first. Resources can also be used efficiently with the help of prioritization. Otherwise, the test cases get no resources even which are important to execute. They suggested that dependency can be open or closed. Different algorithms can be used to detect both kinds of dependencies. Basically, the test case.

Shin Yoo and Mark Harman [23], they suggested in their paper and focused on test suite minimization, test case selection and test case prioritization. They suggested that the work on these three techniques is strongly associated to each other. The function of the test suite minimization is to check out for the test plans. Test suite minimization ensures the duplicity, by decreasing the size of the test plans. This leads to the test suite minimization. Test case selection process selects the test cases from a typical set of test cases. The priorities to different test cases are provided by test case prioritization technique. These test cases will be run in the order of their priority. Higher priority test case will run first and then the test case with lowest priority. Dependencies and prioritization of test cases the main basis of this paper.

Amitranjan Gantait [1] has used model transformation approach in order to provide test cases commencing activities. The author used a technique which generates test cases from extended activity models. This technique will first create a halfway model, and after its creation, test cases get generated from it. The author also invented a technique to prioritize those test cases. This technique is based on the possibility of twigs (branches) at the node which is having any decision in the model of activities. ATM as a sample example has been put forward for checking the effectiveness of the proposed approach. This has proved the

efficiency of the approach.

Harald Cichos and Thomas S. Heinze [9], suggested an approach for systematically integrating the test cases rather than randomly making the pair and merging them aimlessly. The size of test suites can be decreased by this technique. In this technique resemblance among the test cases is identified and then based on those resemblances a pair of test cases is generated. After creating the pair based on the resemblance, integration is performed with which the size will get decreased. A model checker has been used for analytically creating the test cases.

Xiaobo Han, Hongwei Zeng and Honghao Gao [24], based on heuristic models implemented a new technique of prioritization. This technique obtains two types of information from the models. The heuristic models help to detect the defects in earlier stages. This earlier detection of defects serves by reducing the error rate and because of this the average percentage of the fault detection gets increases. This technique has proved the better early rate of the detection of faults. This will lead to the quality product.

Quart-ul-an-farooq, Mohammad Zohaib Z. Iqbal, Zafar I Malik and Matthias Riebish [17] studied an approach of state based regression testing (START). START, an eclipse based tool was used for this approach. Dependencies exists is various states gets treated by START. START makes it possible to care for the dependencies, during the modifications are taking place. UML 2.1 class diagrams and the state machines are taken into account by START. The efficiency of this technique was proved by a case study on the "student enrolment system". This shows the test suites reduction. START works fine even when it is integrated with other testing tools. Parsor, comparator and test suite analyser are the main parts of START. XMI v2.1 format is used as an input in START.

Siripong Roongruangsuwan and Jirapun Daengdej [19], has focused mainly on the test case prioritization. They suggested that the current test case prioritization techniques are unable to make order of test cases when the priority of those test cases is same. They also suggested that test suite prioritization should also be taken into account. Hence they invented the two

techniques. First technique, which handles the test cases having the same priority. The second technique will efficiently prioritize the test suites rather than test cases.

Hema Srikanth, Lauri Wlliams, Jason Osborne investigated PORT[10] (Prioritization of requirements for test) technique, which is based on various prioritization factors like: In Customer assigned priorities on requirements factor, priority will be assigned by the customer instead of any technique, developer or tester. Requirements volatility will check that, how many times the requirements have been changed during the development of the programs. Fault proneness is basically the probability of a requirement. This can result into a fault which further can get converted into failure. Complexities of implementation are the complications faced by the development team during the implementation.

Xuan Lin [24] concluded that Prioritization of test cases is the act of providing the order in which test cases must be run. These results in the better utilization of the resources and also the test cases, which urgently needs to get executed. It takes the priority into account to run the test cases. The test suites which are redundant get removed from the test cases. Since they are not eliminated, hence they will take more time to get removed and also take more resources in comparison. Test complete tool provides an environment for automating the testing. This tool basically is made for the unit testing and functional testing but it also can be used for regression testing.

Swarnendu Biswas and Rajib Mall [18], have described test suite management. The authors suggested that in order to provide more efficient functionality, during the modifications being made in the original test suites, there is a requirement of test suits to be managed. It can be done by putting some new test cases in, while some of the test cases are needed to get out of the suite. This in and out process require management. The process of again using the test cases which have been used earlier, before the modifications were made is known as test case reuse. The reuse of the test cases reduces the effort and money.

Gregg Rothermel, Roland H. Untch and Mary Jean Harrold [6], have proposed the technique for prioritizing the test cases with the help of execution information. Firstly test case

prioritization is done on the basis of the code which is covered. Secondly, prioritization is done on the basis of the code coverage which has not been covered earlier. Thirdly, prioritization is done on the basis of the capability of discovering the defects. Practical implications of these techniques have been done on various test suites. Increased average percentage rate of fault detection (APFD) has been found which reflects the efficiency of these techniques. The main drawback of these techniques is that they are very costly. Hence further research is going on to reduce the cost of the factor also.

Prateeva Mahali and Arup Abhinna Acharya [16], have studied the model based optimization and prioritization for the test suites in regression testing. Their methodology suggests using the genetic algorithm, the optimization of test cases is first performed and then prioritization is performed. This can result in producing the better results and the successful regression testing. The authors also found that there is reduction in the cost and time with these methods. Case study of a shopping mall was taken into account in order to provide results which are better in comparison. They also have represented the system under test by an activity diagram in UML 2.0.

B Bhattad and A Kothari [3] have studied the concept of fault detection in the database of the system. They used the different techniques which can be used for the detection of the defect in the database. They also explained the various types of the defects which can be their missing attributes, broken links and have prioritize them according to the severity level of the defect. Authors made the design for the testing of the database which can help in saving the cost and the time of testing.

Gregg Rothermel,Sebastian Elbaum, Alexey Malishevsky, Praveen Kallakuri, Brian Davia [7] has described a new tradeoff which can be used for the selection of test suites. They have considered the cost-benefit as the major factor for the grouping of the test suites. The test granularity is considered along with different methodologies of regression testing. This provides a better trade off and cost effective testing.

H Cichos, and T.S. Heinze [8] has described a technique which is used to reduce the test suites size. The authors described, that there are certain cases where the test suites size

exceeds the cost of the execution of the test case. They have also proposed the methodology which suggests that in order to reduce the size of the test suits; we can use the concept of test pairing. In this, the test cases are paired using some concept and then they are merged. Merging of the test cases reduced the overall size of the test suites which in turn reduces the overall cost of the regression testing.

H Do and G Rothermel [9] have described different types of test case prioritization approaches. The early detection of faults can help in the reduction of the cost of regression testing. The authors want to describe these test cases in order to have an advantage over the cost reduction. The authors describes that most of the test case prioritization technique is based on the self seeded fault. However in the real scenario the faults are emerged because of the mutation. Hence in order to study the effectiveness of the prioritization he has conducted the controlled experiment. Their study shows that the ability of the prioritization to detect defect is relative to the fault induced in the test cases. The authors have also compared the prioritization on the basis of fault seeded and mutated fault and performed an empirical studies.

J Ryser and M Glinz, [11] has described a kind of dependency diagram and the notation which are the advancement in the UML diagram. These are used to depict the functionalities of the project as in the form of the scenario. Each scenario has some relation with another scenario and they are interlinked. The interaction between the scenarios was shown in the form of the diagram with dependencies in them. The authors also explained that how these dependencies chart can be useful in the testing of the product. This will be beneficial to reduce the cost of the regression testing.

M. J. Harold, J. A. Jone, T. Li, and D. Liang [13] have described a technique which is used to select the test cases from the test suites and overall reduces the size of the software. The authors have used the RETSET algorithm which is very effective in reducing the overall reduction of the test cases from the test suites. They have used this technique to test the program of java which is very common. Their work is also useful for testing java program which adds the external libraries from outside. Their technique can be safely applied for the selection of test cases for the java project. Selection of the test cases reduces the test suites

and thus makes the regression testing faster and within the budget.

X Lin [24] has described the result of his survey on the gap between the techniques of regression testing in the research, as well as in the industries. The author has noticed that there is a huge gap which exists between the industry and the research related to the regression testing. The author has explained that regression testing is very crucial for industry and it requires regression testing to be carried out. Their important modification and good testing is always needed for good quality product. The author also has explained that this gap between the companies and the research will help in inventing new techniques which will be helpful in enhancing the efficiency of regression testing.

CHAPTER 3
PRESENT WORK

Regression testing is used to test the various component of the program. It is used to verify that after the modification or the bug fixing process, the other functions are not affected. Regression testing is a very essential part of the test life cycle and it cannot be avoided in any of the product development process. However sometimes the regression testing can be very time consuming and can require lots of cost. In order to make the regression testing effective there is a requirement of test case prioritization techniques. Implementing regression testing with the effective test case prioritization technique provides a higher rate of fault detection. If maximum numbers of the faults are detected in the early phases of the development, then the quality of the product will be good and customer will be highly satisfied. In order to make the test case prioritization efficient, the hybrid approach of model based testing and dependency is used.

3.1 Problem Formulation

In the recent years, much research is already been done on the various aspects of the software engineering. Regression testing is one of the most important areas of the research which is required by many of the industries as they are moving towards the agile development. Large and complex project needs to be tested again and again, after the modification or the fixing of the fault. Hence there is a need to make regression testing efficient so that it can detect fault at the early stages of development. Test case prioritization is one of efficient technique which can prioritize test cases and can provide higher rate of fault detection. Many techniques were used by various researchers but they have their own limitations. The problem is that they are based on previous fault history of test cases. This leads in allotting the least priority to those test cases, which are extremely important for the product development as compared to those which are less important. To overcome this problem a technique, on the basis of functional importance of the test cases and their related functional test cases has to be formulated.

3.2 Objectives of the research

1. **To propose a hybrid technique for test case prioritization:** There are many

algorithms which has been suggested by various researchers. Our aim is to study, analyze those techniques deeply, and understand the limitations and advantages of those techniques. Finally we proposed a new hybrid technique that may provide better results in terms of rate of fault detection.

2. **To implement the hybrid technique for the test case prioritization and compare it with previous technique:** The aim of the research is to implement the proposed hybrid technique and to compare it with old techniques of test case prioritization.

3.3 Research methodology

Test case prioritization is the technique which is used to make the regression testing efficient. The main aim of the test case prioritization to is to find out maximum no. of the fault in the less amount of time i.e. in the early stages of the development. Moreover at the same time the budget needs to be taken care, as the companies do not want to spend much of their investment in testing and cannot compromise on that. Hence we will study the different techniques which are used in the past for test case prioritization. This can lead us to come out with the new technique which is more efficient as compared to the exiting one. The research methodology which we followed during the research was as follows:

1. **Study and analysis of techniques of test case prioritization.**

Study and analysis by various researchers of the different techniques of test case prioritization were first on the basis of past fault finding, second on the basis of the chronological order- a) It prioritize the test cases on the basis of the fault detected by the test cases in the past history. This means we need to maintain the list of fault detected in the past. b) There can be such a case, where a test case which has detected less number of fault in history but may be very important for the development. Hence due to that it has given the lower priority in the order, and any test case which is not so important but has higher number of fault detection value and is given a higher prioritized value, third on the basis of the cost of the test cases and fourth on the basis of user requirement. After the study of all these techniques our analysis showed that there are some limitations in each of them if take into consideration.

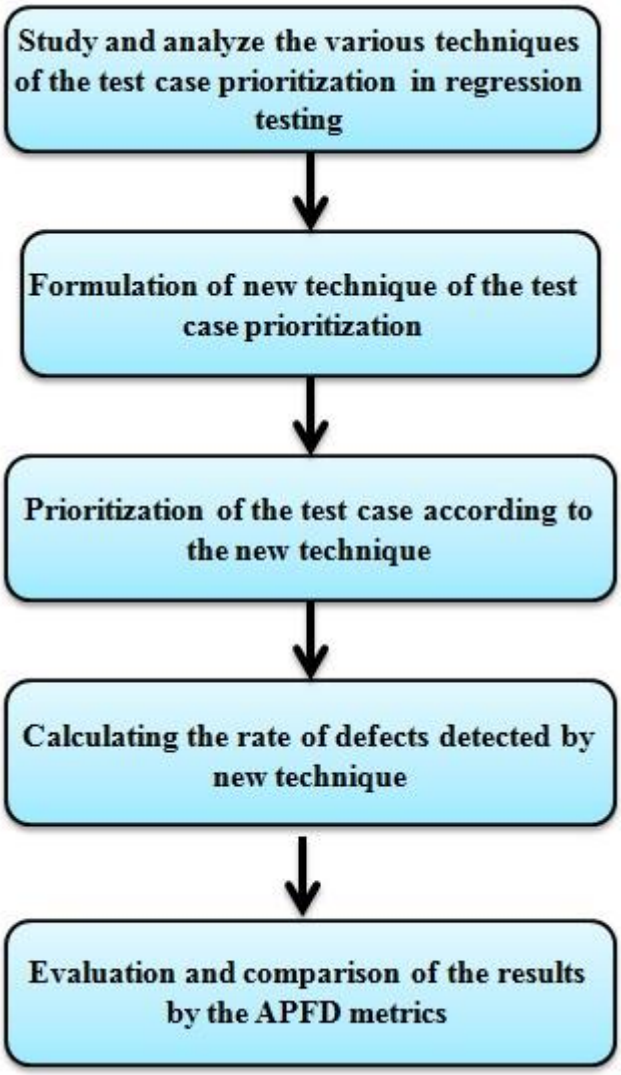

Figure 3.1: Methodology of the research

2. **Formulation of new technique for test case prioritization.**

The formulation of the new technique will provide better result than the existing technique. The new technique of the test case prioritization is based on the functional importance of the test cases rather than on the fault detection.

Proposed technique:

Steps of proposed technique are as follows:

a. Select the project and maintain the database (i.e., information related to the project,

such as number of function it has, number of time each function has occurred in the project, number of function affected by each function).

b. Prepare the model of the project.

c. Identify the changes which had occurred due to modification in the project. This is done by comparing the modified project database with the old database.

d. Identify the functions which are affected because of the modification.

e. Evaluate Function Values (FV) of each affected function, according to the formula of Function value.

$$\textit{Function value} = \frac{\textit{Number of time function encounter}}{\textit{Total number of factor being affected}}$$

f. Traverse activity diagram with the help of Depth First Search (DFS) to find out the dependent functions.

g. Calculate the Total Functional value (FTV) by adding functional values of each dependent function.

h. Test cases are prioritized according to the decreasing order of the FTV value.

i. Calculate the rate of fault detection by the APFD metric.

Maintenance of the database: in this step we need to maintain the database which has the information about the projects- like project id, project name, the information about the different functions which are used in the project. This database will further be used for finding the changes in the project. Matching of the modified project with the existing project database was carried out; being the next step is to compare the new project with the existing project database. This matching was done in order to find out the affected function due to the changes. After the identification of the affected function, the functional value (FV) of the each function is calculated. Calculation of the functional value of each function, requires two inputs, one is the number of times the function encountered and another is the number of function affected by that function.

After the calculation of the functional value of the each function the next step is to identify the dependent function. That means identification of all those function which are affected due to the change. Finally the total functional value (FTV) of each function is calculated. This is done by the adding up all the functional value of the related function.

3. **Prioritization of the test case according to the new technique**

   After the calculation of all the FTV value the next step is to prioritize the test case. The prioritization of the test cases will be done according to the decreasing value of the FTV.

4. **Calculating the rate of defects detected by new technique**

   Finally after the generation of prioritized list of test cases, the next step is to evaluate the result by APFD metric. Average percentage of fault detection (APFD) metrics calculates the rate of rate of fault detection over the testing period. The higher the value of the result of the APFD metric, the better will be the rate of the fault detection.

   The APFD can be calculated by the following formula:

   $$APFD = 1 - \frac{(TF1 + TF2 + \ldots\ldots\ldots TF\,m)}{mn} + \frac{1}{2n}$$

   Where, T is the test suite which is under the evaluation.

   m = number of the fault which is contained in the program p.

   n = number of the test cases to be tested.

   TFi = it is the first test which has discovered the fault i

5. **Evaluation and comparison of result by using the APFD metrics.**

   Result of both the algorithm will be compared with each other to analyze that our technique is efficient or not.

Case study of shopping website:

A case study of the online shopping website been taken as an experimental model. In this case study, by the concept of model based testing we designed a model of online shopping website. This model displays its various functions. In every industrial project there is a continuous change which is required for the betterment of the product. Every modification, which has been introduced in the product, needs to be tested for regression testing. The

model will help in the identification of the function which requires changes and those functions which are affected because of that change.

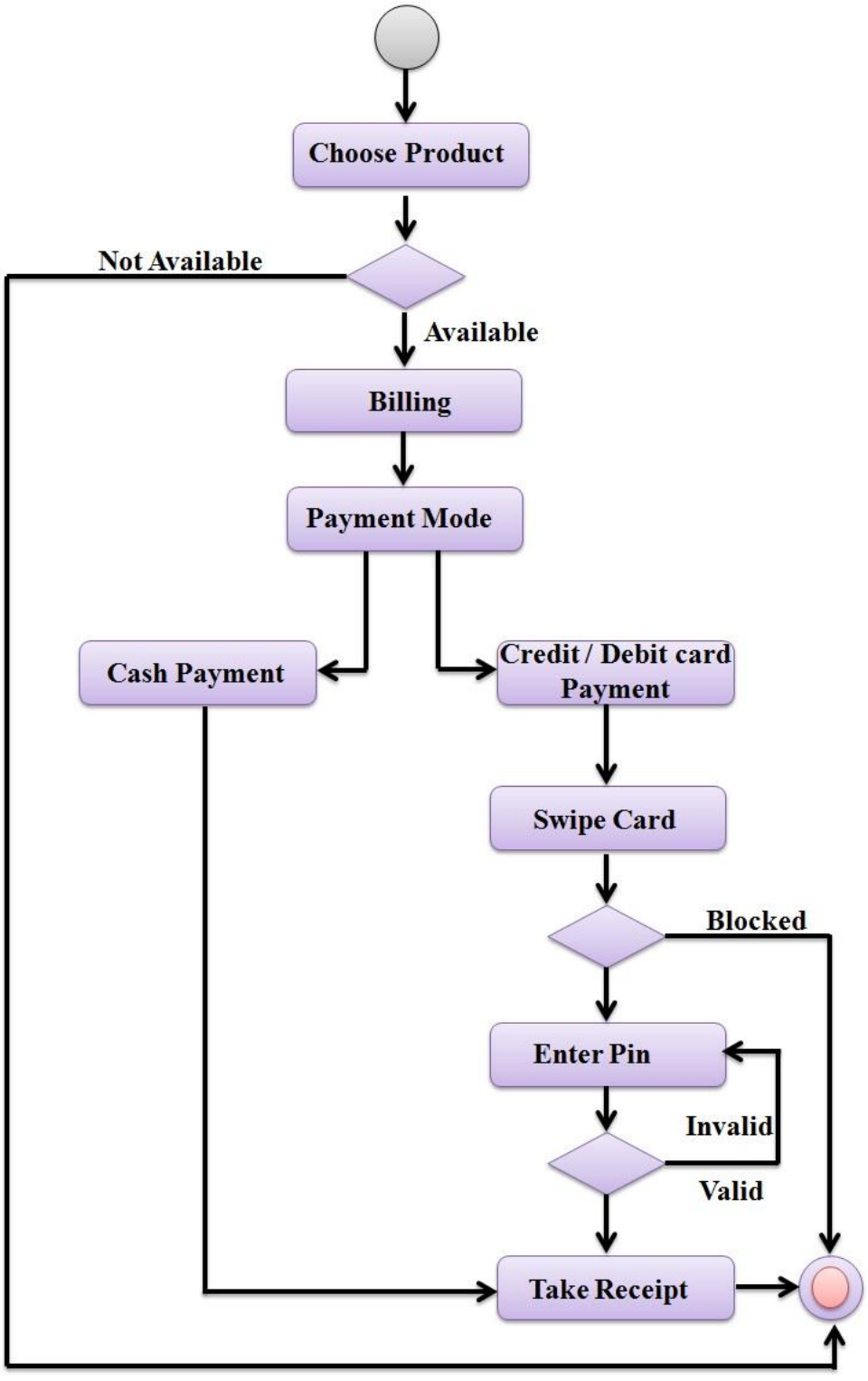

Figure 3.2: Activity diagram of shopping website

CHAPTER 4
RESULT AND DISCUSSION

In the previous studies it has been reported that, different techniques are dependent upon the number of fault detected in the past history by the test cases. These test cases were prioritized on the basis of the greedy algorithm. The higher the number of fault detected in the past history, the greater will be its priority. This algorithm has certain problems associated with it. Firstly it is dependent on the past and secondly it has not considered the importance of the function. However there can be such test cases which have higher importance value because of the criticality of the function. This was just because it has discovered fewer faults in past, it have provided less priority.

Another algorithm which is used for test case prioritization is the genetic algorithm. Genetic algorithm is based upon the fitness value, the chromosome no. and mutation. This prioritizes the test cases based upon the estimated fitness value. In the proposed technique the test cases were prioritized based on the calculated Total Functional Value (FTV). This leads to much better result of the rate of the fault detection. The FTV value is based upon three important aspects. 1) The importance of the function. 2) The number of the function associated with its function, that provides greater functional coverage and 3) it does not need the information of the fault detected in past history, which means it is not dependent on past history. The model which is generated is itself sufficient to provide all information. Calculated FTV value is more reliable than the estimated value of the genetic algorithm. It provides the higher rate of the fault detection. The higher rate of fault detection has been shown in this study. The calculation and results are proved by APFD metric. The model which is generated will help in the identification of the modified function. After the successful identification of the modified function, the functional value of each function is calculated.

4.1 Tool used

MATLAB (Matrix Laboratory) is developed by MathsWorks. MATLAB is a multiparadigm computing environment and it is also a fourth generation programming language. MATLAB allows manipulation of matrix, function, data plotting, algorithm implementation, and user interface creation. MATLAB is an interface which can easily be integrated with other

programs written in different languages such as C ,C++, JAVA, FORTRAN and Python. MATLAB is primarily used for the numerical computing, but there is an optional tool box which uses the MuPAD symbolic engine, that allows access to symbolic computing capabilities. Simulink, act as an additional package which provides graphical multi domain simulation and Model -Based Design for dynamic and embedded system.

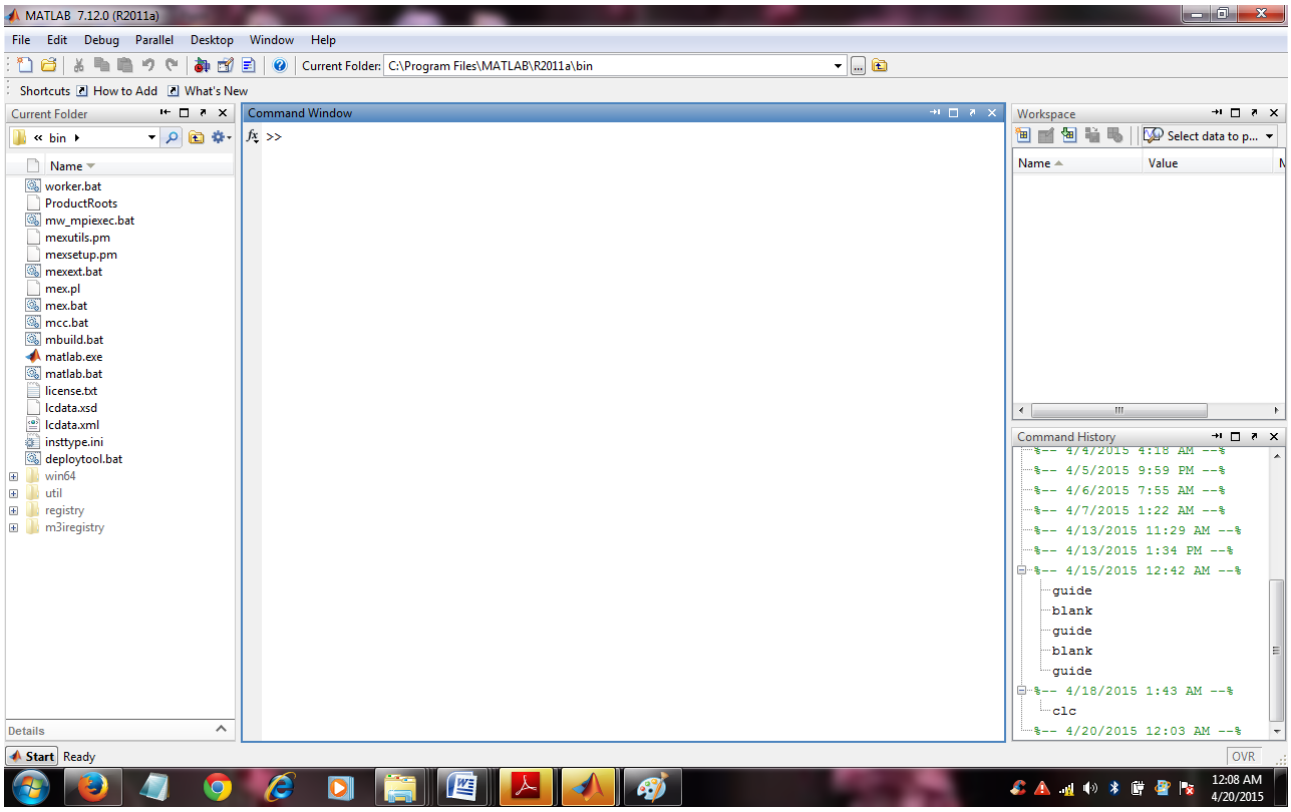

Figure 4.1: MATLAB working environment

4.2 Results

The solution to the problem is represented by the snapshots of the implementation. The implementation has been done in the MATLAB.

The Figures shown in the next sections, shows the snapshots of the implementation which was carried out in this study.

This is the online shopping project which has been selected to represent the implementation of the research.

However the research work can be implemented with any application.

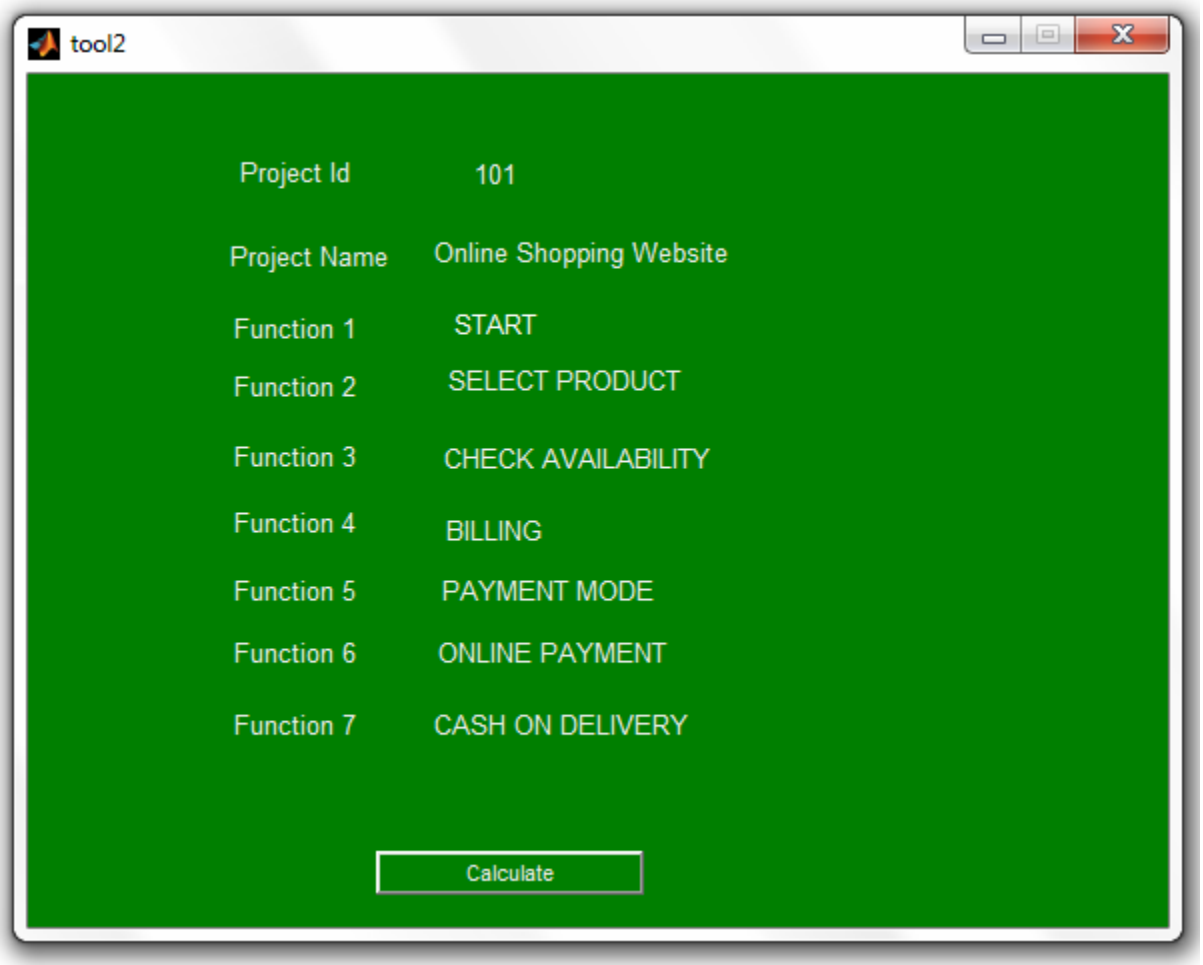

Figure 4.2: Snapshot of functions of the project

The Figure 4.2 represents the project id, name of the project and the functions which are numbered from 1 to 7. These functions have been identified from the model which has been generated to study and it depicts the various important functionalities of the projects. The functions which are shown above have been modified to make enhancement in the project. When there is any modification in the system, the regression testing has to be done. Using our technique we will find out the prioritized order of the test cases of these functions. The prioritized order will be such that that, those that test cases which have higher importance value, will be given a higher priority in the prioritized order of test cases.

After the display of these function which requires some changes. We have calculated the functional value (FV) of each function.

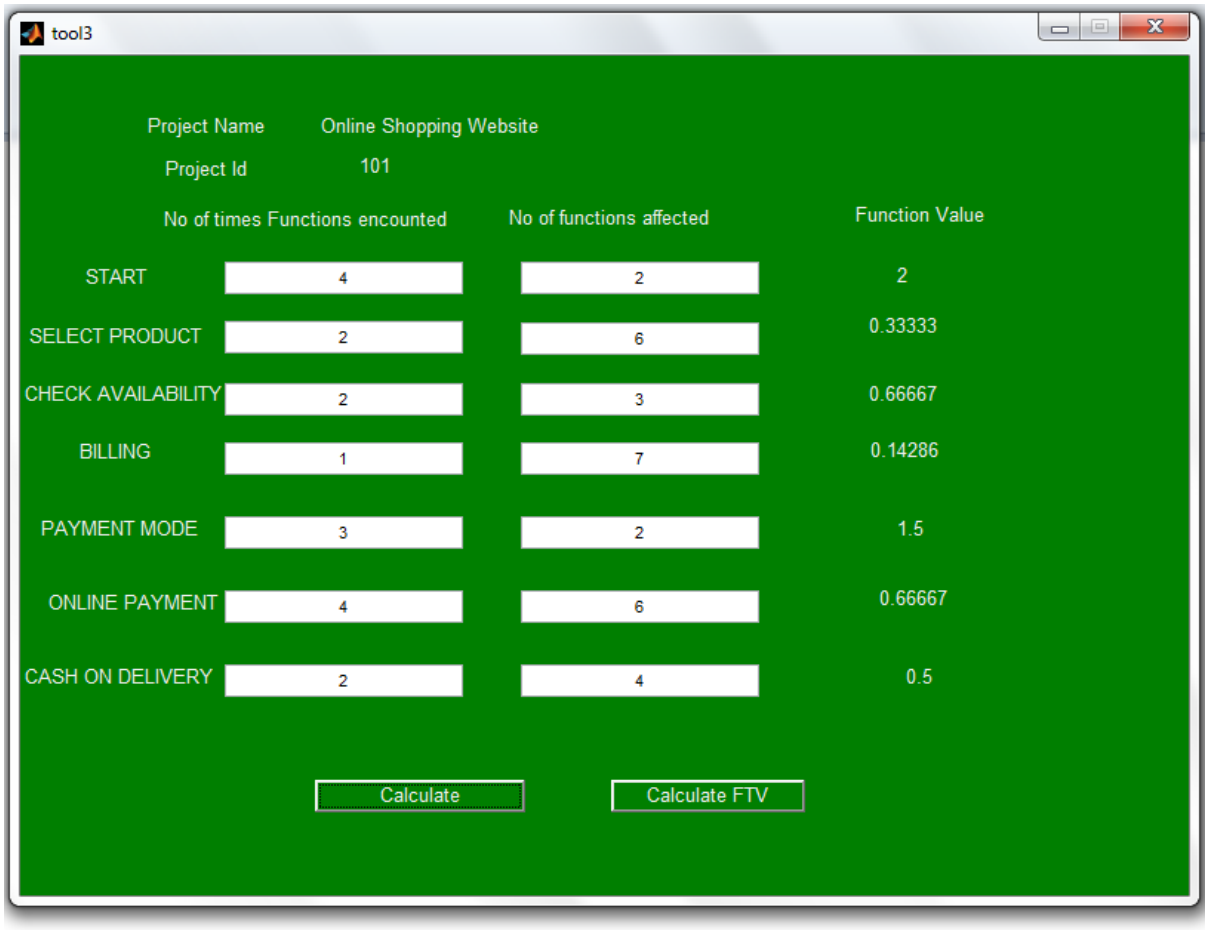

Figure 4.3: Snapshot of dependency value calculation

The above Figure 4.3 shows two values, which are given by the user as the input. The first value is the number of times the function encountered that represents the importance of the function in the project and second is the number of functions affected which provides the greater functional coverage. On the basis of two values the functional value (FV) of each function is calculated by the given formula.

$$\textit{Functional value} = \frac{\textit{Number of times function encountered}}{\textit{Number of function affected}}$$

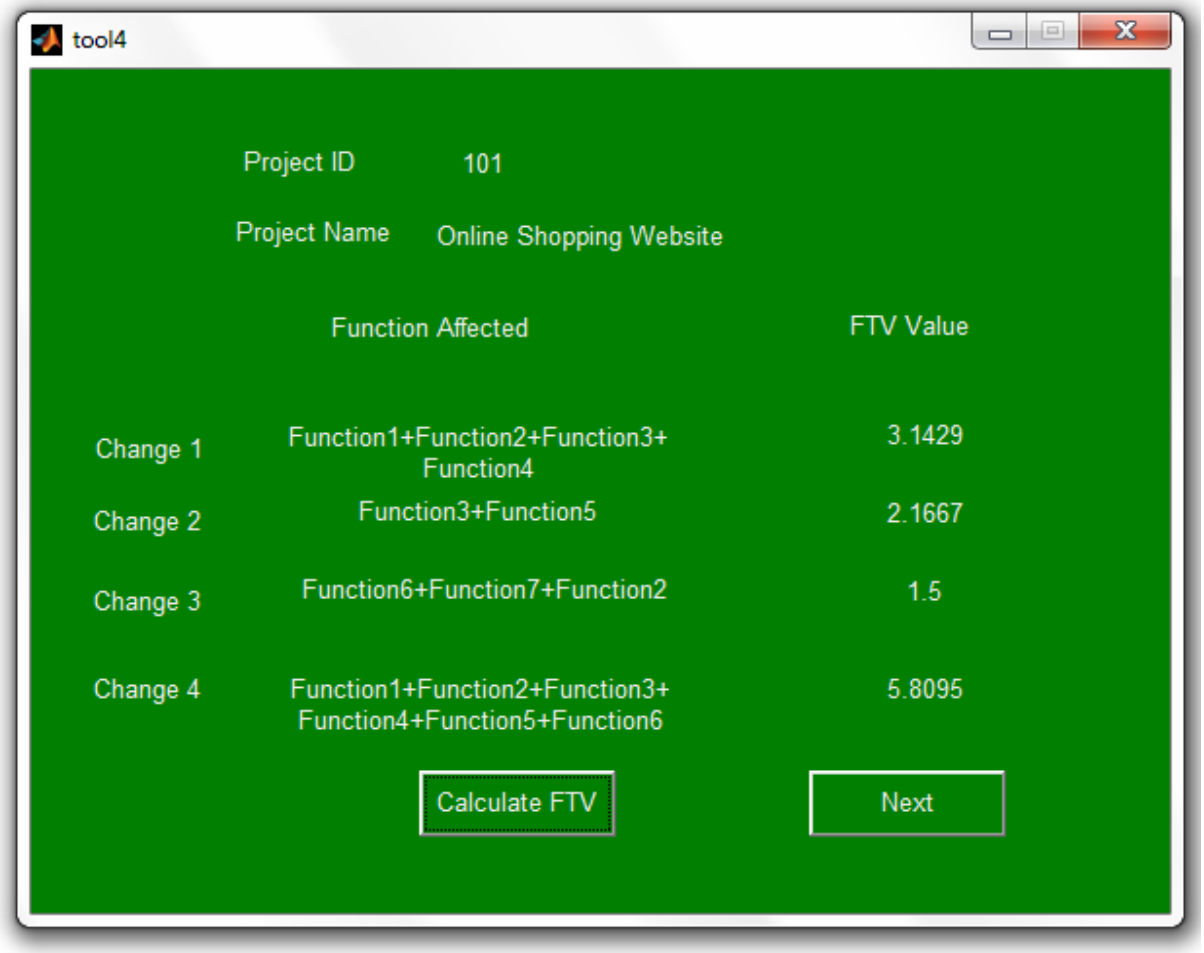

Figure 4.4: Calculation of FTV value

In this Figure 4.4 represents the changes and due to that change the function which is affected are also defined. On the basis of the affected functions, the final FTV value is calculated.

The Figure 4.4 also suggests that due to change1 the affected functions are Function1, Function2, Function3, Function3, and the Function 4.

FTV are calculated by adding the functional value of each function.

For example:

For Change3 FTV value = 1.49997

This is calculated by (FV of function6+FV of function7+FV of function2)

FTV for change3= 0.66667+0.5+0.3333 = 1.49997

In the Figure 4.5 the values of the changes are shown those values are the values of the corresponding FTV of the change which are calculated by the adding the FV of the various

Figure 4.5: Test case values

dependent function. According to these FTV values the test cases are prioritized. The higher the FTV value, the greater will be the dependency in the function and it will have higher importance value. Hence it will be tested first in order.

In the Figure 4.5 the different changes which affects the project has been shown. Corresponding to each change, the test cases of the affected function has also been shown. Each change has a particular calculated FTV value associated with it. The test cases of the changes will be prioritized in the decreasing order of the FTV value. The decreasing value of the FTV value indicates that, the test suites which have higher FTV value has more number of dependent functions than that of the test suites which have lower FTV value.

The tool which is generated to compare the fault detection value of the new algorithm and the existing algorithm is shown in Figure 4.6. The graph shown in the tool represents the value

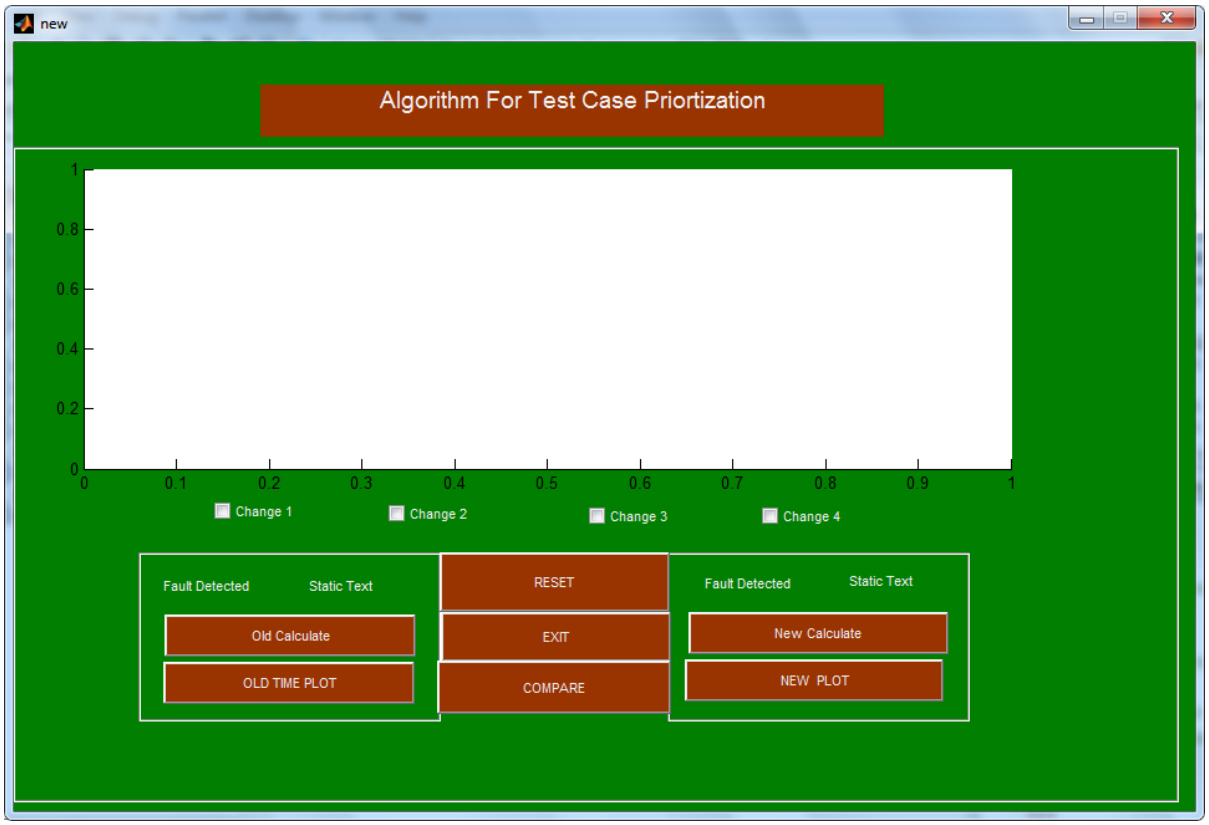

Figure 4.6: Default view of the tool to calculate faults

of the fault detected in graphical form. This tool can be used to represent and compare the values of the fault detection for the multiple changes.

When any of the change is selected, the corresponding fault detection value is represented in the numerical form. Clicking the plot button will result in the generation of the bar graph and the comparison can also be shown in the pie chart form. By comparing we can see that new algorithm is much more efficient than the existing genetic algorithm. New algorithm detects much more faults. RESET button is used whenever we want to test the algorithm for new change and want to compare. EXIT is used to close the tool and exit from application.

The rate of fault detection for the existing algorithm is calculated by the APFD metrics. Clicking the old calculate button, the rate of fault detection of old algorithm is displayed. The old algorithm which we had considered for the comparison is the genetic algorithm. The genetic algorithm is mainly based on the fitness value. Fitness value is an estimate value

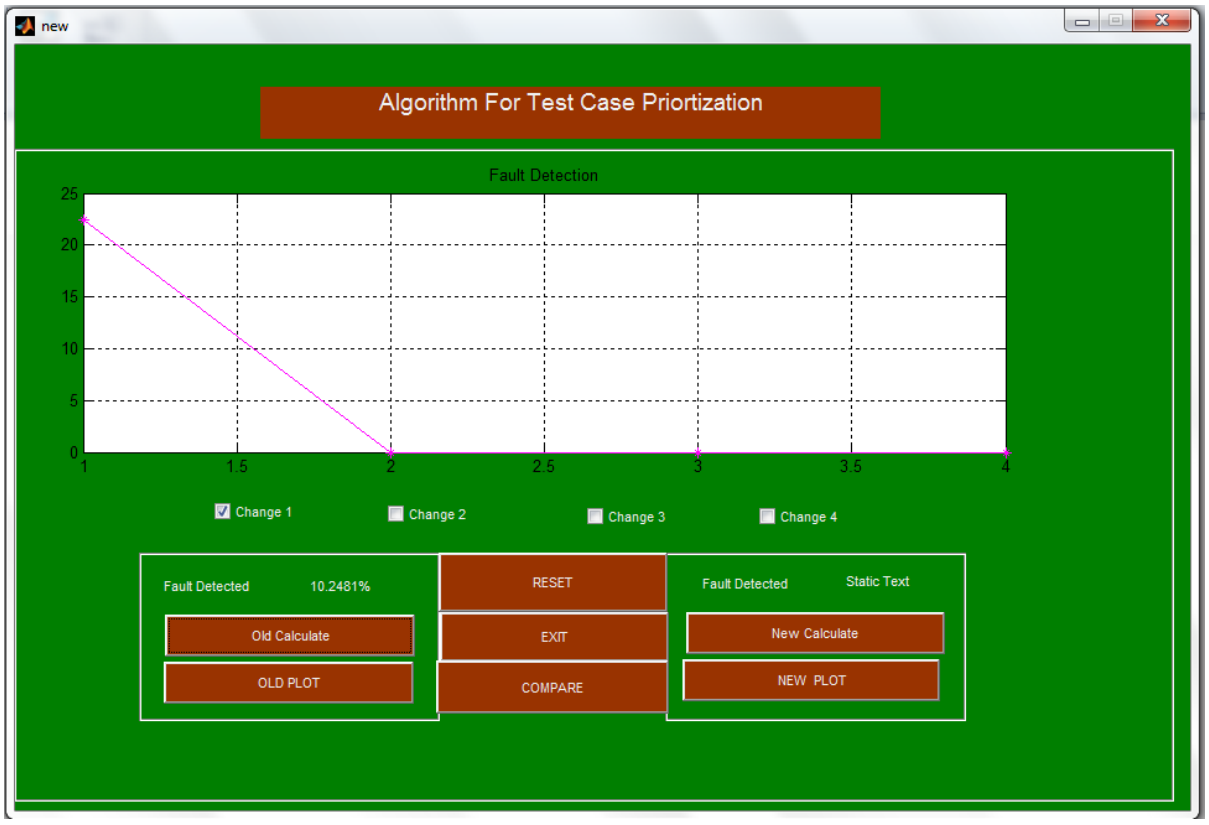

Figure 4.7: Fault detection with existing algorithm

based on the past historical data of the detected fault. Hence the prioritized order which was generated was based on the estimated value, whereas the technique which we used to prioritize the test cases is based on the importance of the function.

The rate of fault detection calculated for the old algorithm is 10.2481 percent for the change 1 as shown in the figure 4.7. For every change the rate of the fault detection can be shown in either the numerical value or in the graphical form. The rate of fault detected by new algorithm is shown in the Figure 4.8 and comparison can be seen between both algorithms.

The Figure 4.8 shows the fault detection rate of existing algorithm i.e., 10.2481 percent. The fault detection value of the new algorithm is 44.3805 percent.

The comparison of the both the algorithm has been shown in terms of the rate of the fault detection. This has been calculated by the APFD metrics. The higher value of the rate of the fault detection by the new algorithm shows that new algorithm is more efficient in detecting

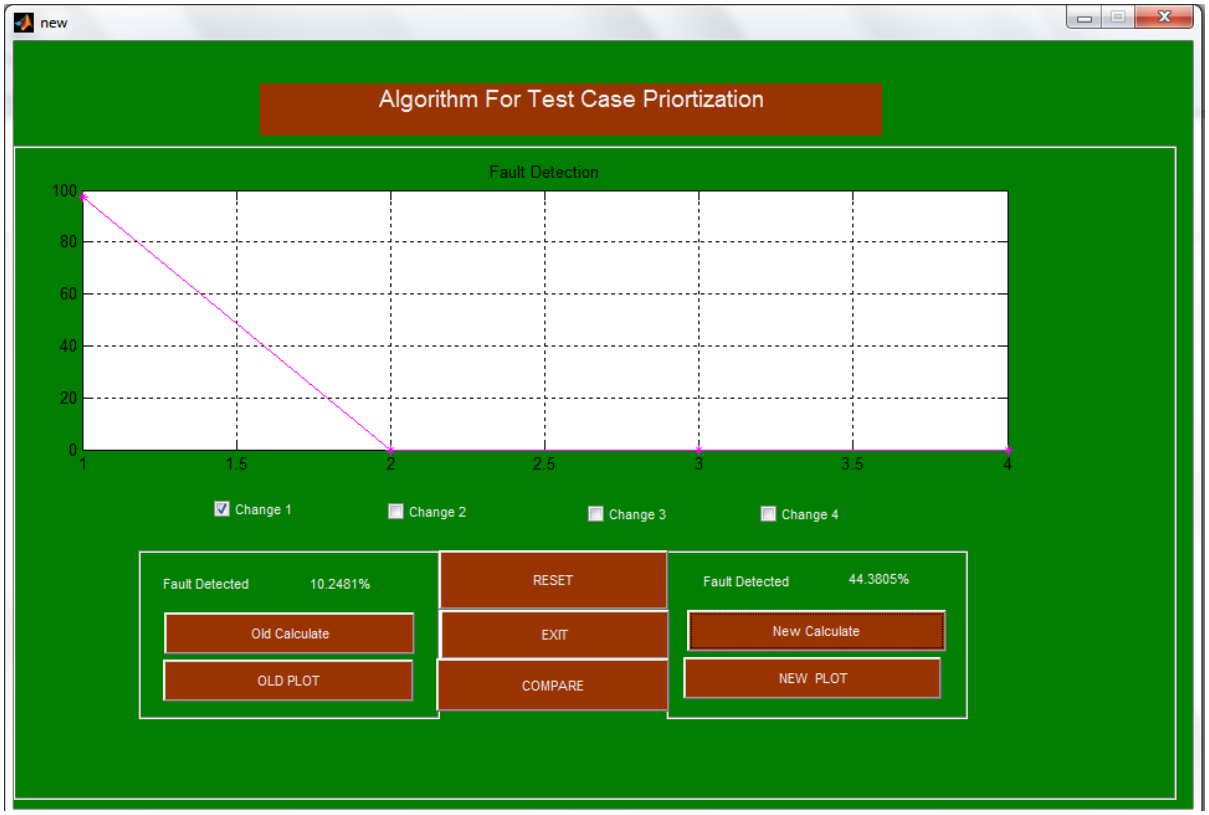

Figure 4.8: Fault detection with new algorithm

faults at the early stages of development. The rate of fault detection can also be represented by the bar graph by clicking on plot new button. The comparison between the two algorithms can be shown by the bar graph and the pie chart on clicking the compare button, which have been shown in the Figure 4.9 and 4.10.

The result has proved that our new algorithm provides much better result as compared to that of the old algorithm. Some of the other advantages of our technique are that it is not dependent on the past history of the fault detection as the old algorithm depends on the old information of the faults detection of the test cases.

4.3 Performance Evaluation

There are seven types of functions are available which are used in the activity diagram. These functions and their descriptions are provided in the following table:

Table 4.1: Function Id and their functionality

Function ID	Functionality
F1	START
F2	SELECT A PRODUCT
F3	CHECK AVAILABLE
F4	BILLING
F5	PAYMENT MODE
F6	ONLINE PAYMENT
F7	CASH ON DELIVERY

The Table 4.1 shows the function which have been modified during the maintenance phase. These are those functions which require the regression testing to be done in order to confirm that, the changes in theses function will not affect the functionality of the other function. In order to carry out the regression testing on the test cases of the functions we need to prioritize them. The prioritization order should be such that, the large number of the defect can be detected as early as possible in the testing phase.

These functions are identified from the activity diagram which is shown in the Figure 3.2. We had considered the application of an online shopping website. The important functions are shown in the Table 4.1. After the identification of the function, the dependent function of these functions has to be identified. The use of the dependency approach to detect the depended function from the model provides the greater functional coverage. Using our technique of prioritizing the test cases is based upon their importance and their tendency to affect other functions.

The Table 4.2 shows the calculated functional value of each function. The function value is calculated by the use of the given formula.

$$\textit{Function Value:} \quad \frac{\textit{Number of times Function}}{\textit{Total number of factor being affected.}}$$

Table 4.2: Calculated function value

Function	No. of times the function encounters	No. of functions affected	FV
F1	4	2	2
F2	2	6	0.3333
F3	2	3	0.6667
F4	1	7	0.14286
F5	3	2	1.5
F6	4	6	0.6667
F7	2	4	0.5

After the calculation of functional value, the total functional values (FTV) are calculated are calculated. CHANGE_1, CHANGE_2, CHANGE_3, CHANGE_4 are the changes which are required in the project and the affected functions corresponding to each change have been shown in the Table 4.3.

Table 4.3: Affected functions and FTV values

No. of changes	Affected Functionality	FTV
CHANGE_1	F1,F2, F3,F4	3.1429
CHANGE_2	F3,F5	2.1667
CHANGE_3	F6,F7,F2	1.5
CHANGE_4	F1,F2,F3,F4,F5,F7	5.8095

The affected functions of the change has been identified by traversing the activity diagram of the project. The greater value of the FTV indicates the larger numbers of the functions which are dependent on each other. The greater will be the FTV value the higher is the chances of the detecting the faults. Hence the test cases are prioritized according to the value of the FTV.

At the end, test cases are prioritized according to the decreasing order of FTV. The test

cases which have higher FTV will execute first and the test cases which have lower FTV value will execute at the last. Hence the test sequence is in order: **T4>T1>T3>T2**

Table 4.4: Prioritized order of test case

Test Cases	FTV
T4	5.8095
T1	3.1429
T3	2.1667
T2	1.5

The Table 4.4 shows the prioritized order of the test cases according to the decreasing value of the FTV. The decreasing order of the FTV indicates that, those test cases which have more dependent function have the higher prioritized value. The dependency approach is used to find out the dependent functions. The use of this approach along with the model based approach provides the greater functional coverage.

After the generation of the prioritized order of the test cases the next step is to analyze the results of our techniques. In order to test the efficiency of our algorithm we used the APFD metrics. The rate of the fault detection of the old algorithm i.e. genetic algorithm was calculated which is based on the operations such as mutation crossover and the estimated fitness value. Hence the rate of the fault detection of the prioritized order of the old algorithm is based on estimated value which is much less then our technique. The comparison of the old algorithm and new algorithm is shown in the Figure 4.9 and 4.10.

The Figure 4.9 shows the comparison of both the existing and the new technique. The graph shows that the new technique has higher rate of fault detection then the existing algorithm. The comparison between the new algorithm and existing algorithm is done by the metrics called average percentage rate of fault detection (APFD) metrics. Each test case has the certain fault detection value which is given as the input in the metrics.

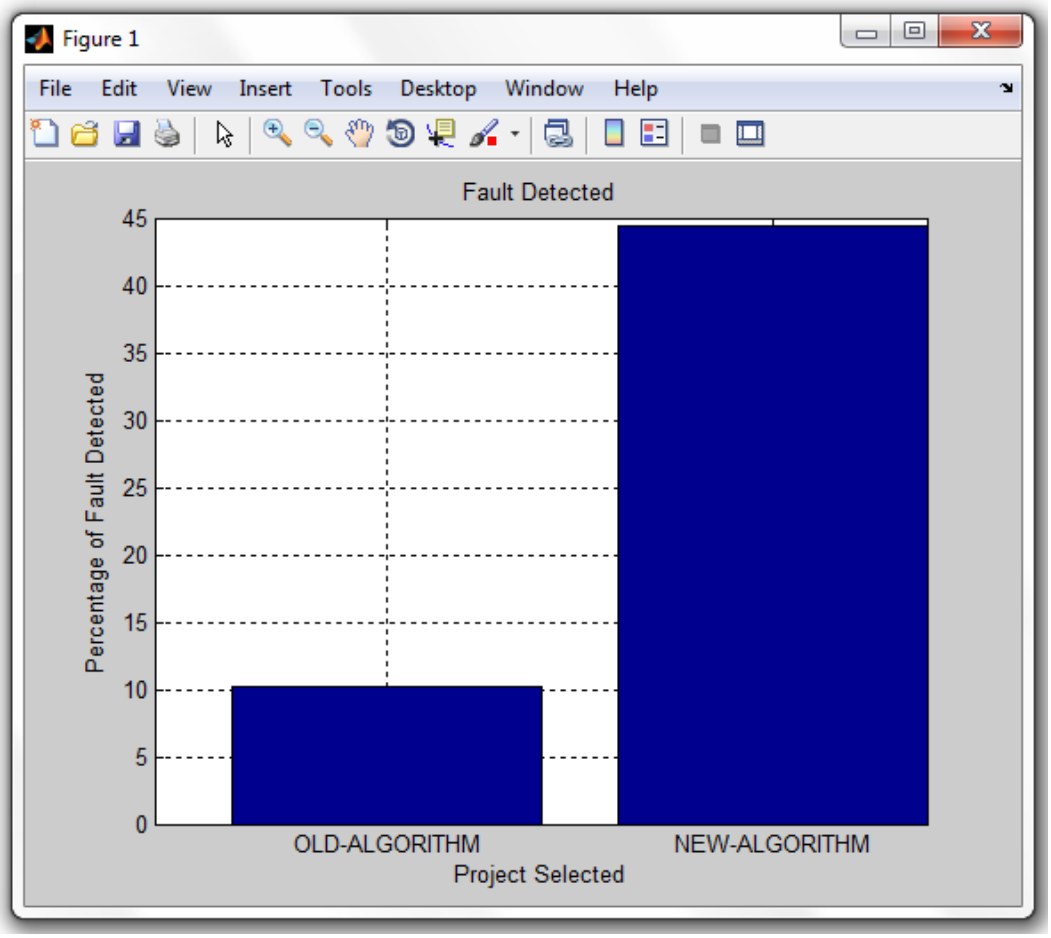

Figure 4.9: Bar graph representation of comparison of new algorithm with the old algorithm

The graph shown in Figure 4.9 justifies that our prioritization order detects the higher number of fault in early stages. Our algorithms have several advantages then that of the old algorithm, not only in terms of fault detection but also in several other ways. Our technique does not require any information from the past history of fault detection as required by the old genetic algorithm and it also provide greater functional coverage.

The Figure 4.10 shows the comparison of existing and new algorithm in terms of percentage of fault detection. The pie chart represents that the new algorithm is efficient in detecting the 81% of the total faults, whereas the old algorithm detects the rest of the total faults in a particular interval of time period. This shows that, the prioritization order which we got as a result of our technique has higher rate of fault detection then existing techniques. The better result is because of the certain important features of our algorithm. This makes it better than

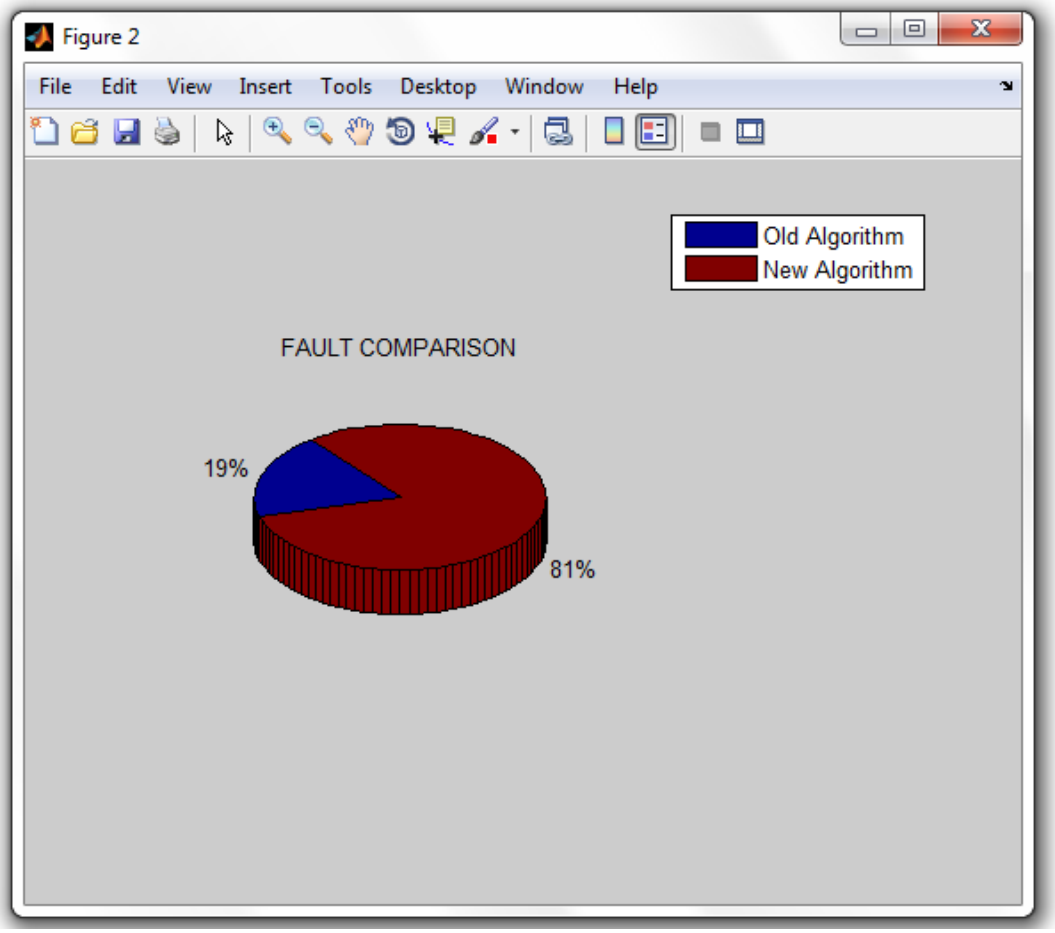

Figure 4.10: Pie chart representation of comparison of new algorithm with the old algorithm

the other old algorithm. Firstly it prioritizes the test case based on certain calculated value unlike genetic algorithm which is based on estimated value. Secondly it is not dependent on the past history of fault detection like genetic algorithm.

CHAPTER 5
CONCLUSION AND FUTURE SCOPE

5.1 Conclusions

Over the past few years many of the prioritization techniques have been introduced and they have their own advantages and disadvantages. We proposed a new technique which has enhanced the efficiency of regression testing. Our proposed technique is a hybrid of two techniques- functional dependency technique and the model based testing technique. Dependency approach has helped us to depict the dependent functions and Model based technique has helped us in generating all those test cases which are important to test the functionality of complex system. Both of these approaches together gave a stronger technique to prioritize test cases. This technique detects fault in the early stages of development and also provide higher rate of fault detection and reduce the time of testing.

5.2 Future scope

Our technique is a hybrid technique in which we used the two approaches the model based approach and the dependency approach. The model is generated in order to depict the functions and find out their importance. In future, we can also extend our approach by adding the non functional features of the project. The dependency approach is used to identify the dependent functions which provide a greater functional coverage.

In future, we planned to test our technique over large number of the projects which includes large number of functions; so that the scalability of the algorithm can be tested .The efficiency of the test case prioritization can also be enhanced by the use of the automated tool for test case generation from the models. The automated tool for computing can also be used.

REFERENCES

[1] A Gantait IBM India Pvt. Ltd Kolkata, India "*Test case Generation and Prioritization from UML Models"IEEE 2011.*

[2] A.G. Malishevsky, G Rothermel and S Elbaum, "*Modeling the Cost-Benefits Trade off for Regression Testing Techniques*", Proceedings of the International Conference on Software Maintenance (ICSM'02), 2002.

[3] B Bhattad1 and A Kothari2 ," *Study of defects test cases and testing challenges in website projects using manual and automated techniques"* Computer Science & Information Technology (CS & IT).

[4] G. Rothermel, R. Untch, C. Chu, and M. Harrold, "*Test Case Prioritization"*, IEEE Transactions on Software Engineering, vol. 27, pp. 929-948, 2001.

[5] G Rothermel and M.J. Harrold, "*A Safe, Efficient Regression Test Selection Technique*" ACM Transactions on Softw. Eng. and Methodology, 6(2): 173-210, 1997.

[6] G Rothermel, R.H. Untch and M.J. Harrold "*Prioritizing Test Cases For RegressionTesting*" IEEE Transaction of Software Engineering Vol. 27, No.10 October 2001.

[7] G. Rothermel, S. Elbaum, A. Malishevsky, P. Kallakuri, and B. Davia, "*The Impact of Test Suite Granularity on the Cost-Effectiveness of Regression Testing*," Proc. 24th Int'l Conf. Sofware Eng., p. 130, 2002.

[8] H Cichos, and T.S. Heinze "*Efficient Reduction of Model-Based Generated Test Suites through Test Case Pair Prioritization*" IEEE 2010 Workshop on Model-Driven Engineering, Verification, and Validation.

[9] H Do and G Rothermel, "*A Controlled Experiment Assessing Test Case Prioritization Techniques via Mutation Faults*", Proceedings of the IEEE International Conference on Software Maintenance, pages 411-420, 2005.

[10] Hema Srikanth, Laurie Williams and Jason Osborne "*System Test Case Prioritization of New and Regression Test Cases*" IEEE 2005 International Symposium.

[11] J Ryser and M Glinz, "*Using Dependency Charts to Improve Scenario-Based Testing*,"Proc. 17th Int'l Conf. Testing Computer Software, 2000.

[12] K.S. Lew, T.S. Dillon, and K.E. Forward, "*Software Complexity and Its Impact on Software Reliability*," IEEE Trans. Software Eng., vol. 14, no. 11, pp. 1645-1655, Nov.1988.

[13] M. J. Harold, J. A. Jone, T. Li, and D. Liang, "*Regression test selection for java software*," in Proc. of the ACM Conference on OO Programming, Systems, Languages, and Applications, 2001.

[14] M Utting and B Legeard, *Practical Model-Based Testing: A Tools Approach***.** Morgan Kaufmann Publishers Inc., 2006**.**

[15] M Utting and B Legeard. *Practical Model-Based Testing: A Tools Approach.* Morgan Kaufmann, 2007.

[16] P Mahali and A.A. Acharya "*Model Based Test Case Prioritization Using UML Activity Diagram And Evolutionary Algorithm*" International Journal of Computer Science and Informatics 2013.

[17] Qurat-ul-ann Farooq ,M Zohaib Z. Iqbal Zafar I. Malik and M Riebisch "A *Model-Based Regression Testing Approach for Evolving Software Systems with Flexible Tool Support*" IEEE International Conference on the 2010, pp. 41-49.

[18] S Biswas and R Mall "*Regression Test Selection Techniques: A Survey*" Informatica 35(2011) 289–321.

[19] S Roongruangsuwan, and J Daengdej "*Test case prioritization Technique*" Journal of Theoretical and Applied Information Technology 2010.

[20] S. G. Elbaum, A. G. Malishevaky, & G Rothermel, (2002), *Test Case Prioritization: A Family of Emperical Studies*. IEEE Transaction on Software Engineering. S. Warshall, "*A Theorem on Boolean Matrices*," J. ACM, vol. 9, no. 1, pp. 11-12, Jan.1962.

[21] Shifa-e-ZehraHaidry and T Miller, "*Using Dependency Structures for Prioritization of Functional Test Suites,*" IEEE Transaction on software Engineering, vol. 39, no. 2, Feb 2013.

[22] S Yoo and M Harman, "*Regression Testing Minimisation Selection And Prioritization: A Survey,*" WC2R 2LS, UK.

[23] X Han, H Zeng and H Gao "*A Heuristic Model-Based Test Prioritization Method for Regression Testing*" IEEE 2012 International Symposium on Computer, Consumer and Control.

[24] X Lin "*Regression Testing in Research and Practice"* IEEE International Conference on. IEEE, *2006.*

[25] Glenford J.Myers, Badgett, Todd M. Thomas, and Crey Sandler. *The Art of Software Testing*, Second Edition.John Wiley and Sons, Inc., 2004.