Abstract—Software testing is the major process in software development life cycle. So more importance is given to the testing process. Due to the large size of code and complexity testing becomes a most tedious job. Whenever the modification is done in the software the testing process is repeated so we need to rerun all the test cases so that the time taken for the testing process is increased. Instead, we can give priority to the test cases and we can select the test cases depends upon the priority given to the test case. The priority is given depends upon the code coverage. There is many existing algorithm for performing test case prioritization and selection. In this paper we propose an algorithm for prioritizing the test case. The proposed algorithm will improve the testing process by finding the faults earlier. So that the cost is also reduced.

Keywords— Test Case Prioritization, Code Coverage, Testing Cost Reduction, Test Case Selection.

I. INTRODUCTION

SOFTWARE program which will have a set of instructions for the system that performs the corresponding task. Each set of program will have their own functionality depending upon its type. Program will process the data for the user and will produce the corresponding output. The software is classified into two major categories [2]. Software Testing plays a major role in the development process of the software. Testing is performed in software to satisfy the user requirements. So that the we can able to deliver a quality software to the user. The budget that is allocated for Software testing is high. Testing is performed in Analysis part, Design part and Coding part that means testing is performed throughout the development process. The Testing function will include planning, the creation of Test cases and then execute the test cases to find out the errors and then display the result of testing process. The testing process can be done manually or by using testing tools. The modification is performed in the software depending upon the request that is given by the user. The user can give the change request depending upon the modification that is to be performed in their organisation or depending upon the poor performance of the software.

The request is processed and the modification is made depending upon the user needs and also getting the feedback from the experts. After the modification is performed the testing process is performed to check whether the modified part make any impact on the existing part of the software. Retesting or Regression is the process of repeating the Testing job with the same set of test cases and additional test cases that are generated after modification is done [1]. Regression testing is used during the development phase and maintenance phase of a software product to assist software-testing activities and guarantee that quality is achieved through various releases of the software product. The Regression testing also called as Retesting which once again test the software with the existing test cases and the new test cases that are generated depending upon the change that is performed on the software. So that we are repeating the testing process by rerunning all the test cases this will take extra time for testing.

Our major goal is to minimize the test suite so that the time required for the testing will be reduced. To perform Test Suite minimization the amount of testcases is to be reduced by giving priority to the Test cases that is present in the test suite. Certain Testing tools are available for performing regression testing. There are many algorithms used for test case prioritization such as greedy, additional greedy, hill climbing, Additional 2 Greedy Algorithm, Heuristic Algorithm and Genetic Algorithm. Each algorithm is having their own functionality. The algorithms will differ in terms of the performance. In this paper we propose an test case prioritization algorithm for prioritizing the test cases and will provide an efficient set of test cases for checking the modified functionality.

II. RELATED WORK

Regression testing is an important part of the software maintenance that is performed on the software that is modified to ensure that the modifications do not affect the modified portion of the software. As regression testing is performed frequently in software maintenance, it will consume a larger percentage of cost in maintenance phase. The selection of test cases is done by algorithms build control flow graphs for module and the control flow graphs to select test cases that is used to check the code that is modified from the initial code.
James A. Jones and Mary Jean Harrold proposed new algorithms for Prioritizing the reducing the test suite [3]. Regression testing will perform test on the modified software to give assurance that no new faults are caused due to modification in to the existing code. Regression test selection will reduce the budget for regression testing by executing only a minimum of the test cases from existing test suite. Whenever software is modified for performing regression testing the process will be take more time if all the Test cases are executed. The code coverage method is used for many fault versions of programs written in C language for early fault detection. The results specify that test case prioritization can improve the fault detection time of test suites. Artificial intelligence techniques are getting popular to work for test prioritization. Walcott et al. has used genetic algorithm for test case ordering and the fault can be detected much earlier by using the coverage information. Kim and Porter [4] also presented a technique, called as “history-based prioritization” that will gather information from previous testing operations to select the set of test cases that is to be used for evaluating the new version of the program. But this technique is not sometimes considered as a “prioritization technique” in the sense done in the literature because it imposes no ordering on test cases. Ordering of tests is the essential characteristic for the definition of prioritization. This approach selects a subset from a test suite, using history data to find out which test cases can be selected, and this process is defined as regression test selection method. Test case prioritization has been applied to improve regression testing effects as specified in their paper. Fayoumi et al. [5] has used ant colony optimization and rough set theory concepts is used to find high quality test case of unit test for object oriented source code. This approach used method call, passing arguments and control flow dependency graphs. A hybrid novel framework was proposed by inspiring the natural ant. Circulation and exploring best test case value had been done through Ant colony pheromone matrix Rough set is used as stopping criteria rule in proposed model. Bayesian network (BN) approach has also been used for prioritization. An empirical study on five java objects indicates the effectiveness of feedback mechanism of BN approach in terms of early fault detection. Particle swarm optimization (PSO) is optimization technique of swarm intelligence paradigm. In author has used test case coverage and used PSO to assess the best possible position of test cases in search swarm in modified software units. Existing test case priorities and fitness of test cases were used as parameters for new priorities of test cases. PSO was found to be more effective in term of time and cost than greedy algorithm. Test case prioritization has also been used to reduce the quality assurance cost as well as for minimizing the effort needed for fault detection. The problem with decreasing fault detection effort was that it may cause the information loss, as a result of which debugging cost gets increase. So it was a big challenge to reduce the quality assurance cost which includes both the testing and debugging cost while minimizing the loss of diagnostic fault information. Author has proposed the on-line greedy diagnostic prioritization approach that uses the observed test outcome to determine the upcoming test case. In this approach better tests were those tests which maximize the reduction of diagnostic cost at each step on average [6].

III. PROPOSED WORK
Regression Test Selection algorithm based on code coverage considers executing test cases which covers modified lines of code. Let C be the previous source code and M be the modified version. TC be a set of code coverage based test cases to test C. When C is modified to M the objective is to find TC*, which is a subset that covers all modified lines of code at the earliest. If two or more test cases, have same number of modified lines and their values also matches then consider test case that has few lines of codes other than modified lines. The input that to be given to the algorithm are the initial code, the initial code after the change is done and history of the test cases and the output is the Reduced Test suite. The algorithm which will assign a value for the test case and the value will be calculated depending upon on the change request that is given by the customer, the impact of the change that is done, the implementation and execution. Customer will provide system requirements, the priority for the requirement and failure details. Developers will rank the requirement detail according to its complexity level. Analyst or the system engineer records the requirements, and the priorities and any modifications in the requirements. Maintenance Engineer resolves the field failures defects and links the failure back to the requirements impacted. Tester provides test cases for each requirement, map the requirement to its test case, and executes the test cases. The following are the steps for the algorithm:

- Initially it will generate the value for each test case depending upon the factors.
- Comparison is done and the priority is given for the values
- Both the process comparison and assigning priority is done for all the test cases
- The values is added for all the test cases
- If the value of the test cases are same then the test case is selected depending upon the change requirement that is given by the user.

The concept is explained in the fig.1. Continuous failure are caused by the fault that is coming along the regular execution, and more effort should be needed to detect such faults [7,8]. The customer satisfaction can be increased, and this process id done by giving priority to customer needs depending up on the importance [9,10]. Implementation complexity refers to individual measure of amount of difficulty perceived by the developers of the requirement by the development team. Amland carried out an investigation to determine that the functions with greater McCabe complexity are those with high number of faults [11]. severe defects that will result in increase of the customer costs and it will be more when compared to solve the same problem in the requirements time [12].Requirements is measured as the number of times the development cycle of a requirement has been changed with
respect to when the requirement was first introduced. The customer Requirement may change depending up on their business changes. There is set of group members whose role is to check the change request that is given by the user and depending upon the importance the request is checked and the decision is made to finalise the request. The request is selected and processed depending upon the priority. And if the fault is detected at the earlier stage so that the quality of the software is improved. So the test case is selected in such a way to detect more number of faults at the earlier stage. The number of lines of code that is to be checked by the test cases is identified and the test case is listed with the coverage data.

We have done some experiment on two application projects to measure the effectiveness of our proposed prioritization algorithm. The details about the project is provided in the documentation of these projects. We selected use cases and test cases and asked the customers to rate these use cases. A value based requirement prioritization tool was used to rank these requirements [9]. In first project our proposed algorithm has detected 80% faults while random ordering produces 60% of faults, in second project our proposed algorithm has detected 70% faults while random ordering produces 52% of faults which again shows significance of our findings.

V. CONCLUSION

The proposed Algorithm introduces test case prioritization technique with the help of the value calculated. The proposed algorithm works for both requirement and testing. The Experiment is done for projects. Test case values were being used to rate the test cases. Algorithm was compared with random prioritization technique on two Application projects and it describes the effectiveness of algorithm for early rate of fault detection. We are currently working to see the effect of proposed algorithm with other techniques. Additionally the proposed algorithm is tested on limited data set. It can be validated by taking large size projects having larger set of use cases and test cases, so that it will be suitable for all type of projects with larger complexity and length.

REFERENCES


**Fig. 1 Architecture**

IV. RESULTS

In our work we have specified the effectiveness of our proposed algorithm for early fault detection by using AFD. APFD metric has been used for calculating the average rate of fault detection per percentage of test suite execution. The APFD is to be calculated by having the average of the faults that is detected during the execution of the test suite. We have taken two sample projects and the detail about the project is described in the Table I

<table>
<thead>
<tr>
<th>Details</th>
<th>Proj 1</th>
<th>Proj 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
<td>Application</td>
<td>Application</td>
</tr>
<tr>
<td>Modules</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Test cases</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>Complexity</td>
<td>Moderate</td>
<td>Moderate</td>
</tr>
</tbody>
</table>

A. Pravin received the B.E degree in Computer Science & Engineering from Bharath Niketan Engineering College, Madurai Kamaraj University, Madurai, India in 2003 and M.E degree in Computer Science & Engineering from Sathyabama University, Chennai, India in 2005. Where he is currently working towards the Ph.D degree in Computer Science & Engineering at Sathyabama University, Chennai, India. He works currently as an Assistant Professor for the Department of Computer Science and Engineering at SRR Engineering College, Chennai and he has more than 7 years of teaching experience. He has participated and presented many Research Papers in International and National Conferences. His area of interests includes Software Engineering, Data mining and Data warehouse.