Functional Dependency based Test Case Prioritization for Regression Testing Using Hill-Climbing Approach

Saloni Ghai[1], Sarabjit Kaur[2]
[1] Research Scholar, Dept. Of Computer Science & Engineering, CT Institute of Technology & Research, Jalandhar, Punjab, India
[2]Assistant Professor, Dept. Of Computer Science & Engineering, CT Institute of Technology & Research, Jalandhar, Punjab, India
Email - salonighai15@gmail.com / er_sarabjitkaur35@rediffmail.com

Abstract: Test case prioritization is basically the process of prioritizing the execution order of test cases to increase the fault detection rate. If fault detection rate is being gradually improved during regression testing then it will provide faster feedback to system developers which ultimately deliver the software earlier. Many of the existing test case prioritization techniques illustrated that test cases can be executed in any order but the concept of functional dependency plays an important role between test cases. In this paper we have described about testing, its types, test case prioritization and its techniques. Regression testing is normally being used to retest the component of a system that verifies that defects are removed or not from the effected part of the software after modifications. This paper enhances the hill climbing approach for test case prioritization by using functional dependency technique. This prioritization approach schedules test cases based upon functional dependency technique .Our approach is quiet efficient as compare to any other technique. Regression testing is quite popular as it helps to make the system error free after modifications.

Keywords: Software testing, Regression Testing, Test Case Prioritization, Hill climbing approach.

1. INTRODUCTION

Software testing is basically a technique of comparing the actual outcome with the expected outcome of the software. The main goal of testing is to check the correct functionality of the system or project. If testing will not be performed with the furnished system then the system may lead to improper results. So it's better to check or test the system, so that we can have excellent results. Testing is an important phenomenon in the software development life cycle process to obtain high quality system that is in terms of product accuracy. It verifies and validates that whether the system is working correctly or not. Regression Testing is a testing that refers to that specific section of the test cycle in which various programs are tested to make sure that changes do not affect other parts of the system. The process of verifying the furnished software in the maintenance phase is commonly known as Regression testing. Time and budget constraints are the major disadvantage of regression testing due to complex process. In regression testing number of regression test cases are re-executed which it is not a practical approach to re-execute every test for each program if once changes occur so it is also an expensive testing process used to detect regression faults. Following are the different types of Software Testing:

Compatibility Testing: The most commonly software failure occurs with the lack of compatibility with the other software and Operating System. Compatibility testing checks how much compatible the software is with other applications.

Regression Testing: When changes to major code occur then regression testing is used to find defects in the code. Regression testing can be classified into:

- **Progressive Regression Testing:** It involves new enhancements and new requirements.
- **Corrective Regression Testing:** It involves minor modifications to the code.

Functional Testing: Function Testing basically verifies the actions performed on the code.

Non-Functional Testing: It is not related with the functions of code. It reflects the quality of the product.

Alpha Testing: Alpha Testing is being done on the developer side. In this Developer performs testing of the software.

Beta Testing: Beta testing ensure that the software is being free from bugs. Beta testing is being done on the user side.

Acceptance Testing: This type of testing is being done by customer in their own lab environment on their own hardware is called as User Acceptance Testing (UAT).

Black-Box Testing: Black-box testing is also known as functional testing. This type of testing ignores the internal functioning of the system. This type of testing is only based upon the output and having no knowledge of internal code.

White-Box Testing: This type of testing includes the detailed examination of interior logic and codes structure. This type of testing is also known as open-box testing or structural testing. This type of testing is based upon an analysis of the internal code.

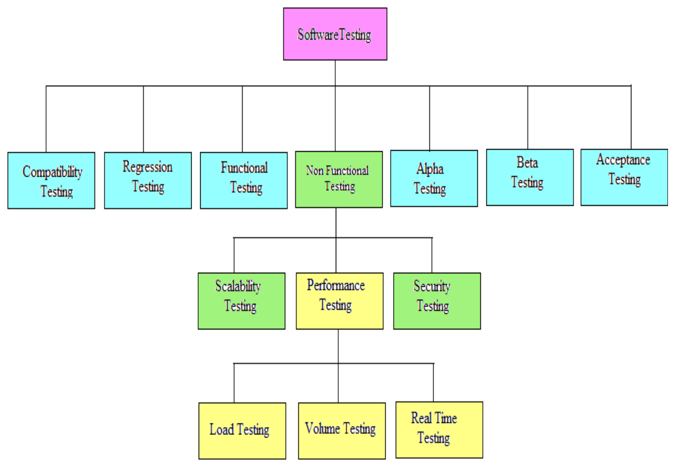


Fig:1 Types of Testing

A. Techniques of Regression Testing

Retest All: This is one of the conventional techniques of regression testing in which all the test cases in the existing test bucket or suite must be re-executed. This is quite expensive technique as compared to others as this technique requires large amount of time and resources.

Regression Test Selection (RTS): Due to certain drawbacks of "retest all" technique, Regression Test Selection has being introduced. In this technique instead of re-running the whole test suite we select a certain part of test suite. This technique also added some new test suites in order to cover those areas that are not covered in the existing test suites. RTS techniques are broadly classified into three categories.

- Coverage techniques: This technique is based upon test coverage criteria. This technique finds coverable parts of the program that have been modified and select those test cases that work on these parts.
- Minimization techniques: It is quite similar to coverage based techniques except that they select minimum set of test cases.
- Safe techniques: This technique does not focus on criteria of coverage as this technique select all those test cases that produce different output with a modified program.

Test Case Prioritization: Test Case Prioritization is a mechanism for arranging a test case in an appropriate manner to increase their effectiveness in order to meet some performance goal and to increase fault detection rate. Therefore we can say that test case prioritization is a technique to prioritize and schedule test cases in a particular order. The main goal to run test cases of higher priority before lower priority test case is to reduce time, cost and effort during software testing phase.

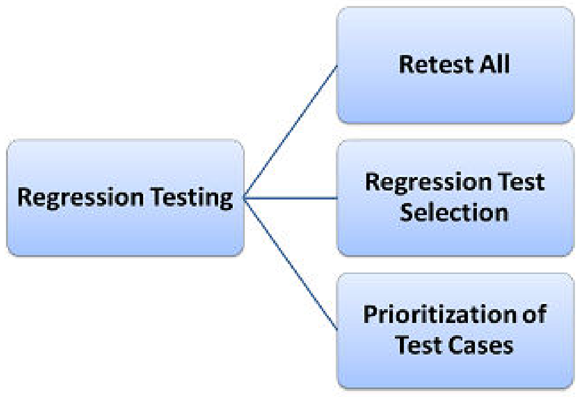


Fig:2 Types of Regression testing

Test case: A test case is set of procedure use to test the software. Test case is a set of condition under which under which a software tester determine whether the application or software system is working correctly or not. To design a test case for particular software the designer must design positive or negative test case for the software. Positive test cases are design to check software under normal condition and negative test case are design to check software at extreme condition. The order of test case execution affects the time at which goal of testing are fulfil. If the goal is fault detection then an improper execution order might reveal most of fault rate which leads to delay in bug fixing activity and the delivery of software.

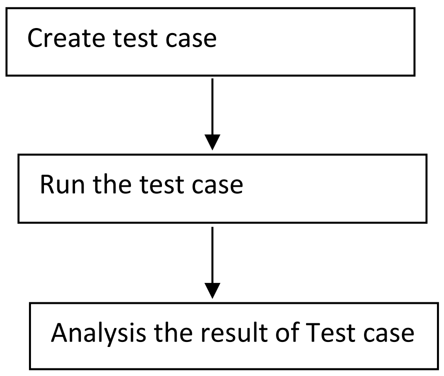


Fig:3. Testing Process

B. Techniques Of Test Case Prioritization

Customer-requirement based technique: In this technique requirement factors of customer are being considered and given some values which is known as weights and based of these values test case weight for requirement is being evaluated. Test cases with high weight value will be executed first following the test cases with lower weight. Certain Customer requirement factors are Customer assigned priority on requirements, Requirement complexity that is change in requirement and Requirement volatility.

Coverage-based technique: This technique is based upon code coverage analysis and the amount of code covered by each test case. In this the amount of coverage is being evaluated and which is used to prioritize the test cases. It is also known as white-box testing technique therefore it is a method that test the internal structure of the software.

Cost effective based technique: This technique prioritizes the test cases on the basis of costs factors such as cost of operations performed by test cases, cost of prioritization, cost of execution, cost involve in validating test cases etc. There are two types of cost namely Direct cost involves test selection, test execution, result analysis where as Indirect cost involves overhead cost and tool development cost.

Chronographic history-based techniques: This technique prioritizes the test cases based on earlier execution history of test cases in current test execution.

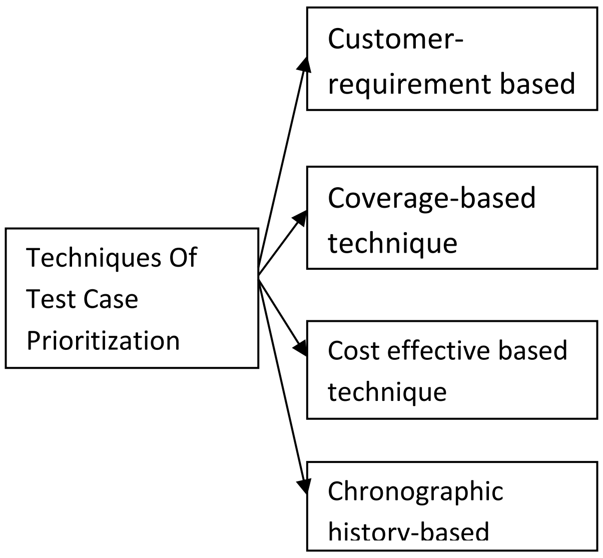

Customer-requirement based
Coverage-based technique
Techniques Of Test Case Prioritization
Cost effective based technique
Chronographic historv-based

Fig:4 Techniques of Test Case Prioritization

2. RELATED WORK

Leung et al. [1989] identified that regression testing is of two types namely progressive regression testing and corrective regression testing which depends upon whether the specification is changed or not [1].

Khan et al. [2006] described a test case reduction technique which is known as Test filter which reduces the size of test suites by simply eliminating unnecessary test cases and also decreases the test case storage capacity, management as well as execution cost [2].

Zheng et al. [2007] identified the most effective algorithm in solving the test case prioritization problem for regression testing and also discuss certain factors that could affect the efficiency of the algorithm [3].

Rothermel et al. [2009] described several techniques for prioritizing test cases and measure the effectiveness of these techniques for improving the fault detection rate [4].

Daengdej et al. [2010] identified two efficient prioritization methods which aim to resolve the problem of many test cases that are having same weight values and the other method is being developed to effectively prioritize multiple test suites [5].

Jatain et al. [2013] illustrated various types of techniques for regression testing and test case prioritization and also describes various types of search algorithms used in the process of test case prioritization [7].

Konsaard et al. [2015] described an algorithm to prioritize test cases based on total coverage which includes five steps namely graph generation, test case generation, test suite generation, fitness calculation and genetic algorithm [8].

Wang et al. [2015] illustrated that most of current regression test case priortization researches neglect to use internal structure of the software which is a significant factor that must influence the prioritization of test cases [9].

Hyunsook et al. [2010] illustrated the effect of time constraint on test case prioritization. Therefore time constraints become a reason to improve prioritization techniques and to improve certain maintenance and testing processes so time constraints are also strongly affect the behaviour of prioritization techniques [24].

3. REGRESSION TESTING

Regression testing is basically used to retest the component of a system that verifies that after modifications defects are being removed or not from the affected part of the software. So the process of verifying the modified software in the maintenance phase is most commonly known as Regression testing. Regression testing is generally being performed by running some or all of the test cases that are created to test modifications of the software. Many techniques have been introduced which that tells that how regression tests are being selected so that the number of test cases does not grow in large amount.

4. ALGORITHMS FOR TEST CASE PRIORITIZATION

Greedy Algorithm: It is based upon the principle that the element with the highest weight is taken into account first, followed by the element with second-highest weight and this process continues until a complete solution has been obtained. It is quite a simple algorithm but in some situations where the results are of high quality it is also prove to be attractive one because it is quite inexpensive both in terms of implementation and execution time.

Additional Greedy Algorithm: The Additional Greedy Algorithm is one of the type of Greedy Algorithm, but it follows quite different process. It combines feedback from previous selection and randomly selects the maximum weighted element of the problem from that part that is not being already consumed by the previously selected elements.

Genetic Algorithm: The population is a set of random individuals. In which each individual is represented by the sequence of genes commonly known as the chromosome. In this selection procedure depends upon the fitness value which decides that which individuals are to be selected as the "parents" for producing the next generation. Crossover is a genetic operator which combines two individuals in order to produce a new individual known as offspring. The mutation operator will alter one or more gene values in the individual depending on the probability of mutation.

Hill Climbing Algorithm: Hill climbing algorithm is a mathematical optimization Approach which belongs to the family of local search. Therefore because of this reason it is also known as local search approach. It is an iterative algorithm that starts its search from an arbitrary solution of the problem and then it will find a better solution by simply changing a single element of the solution. If the alteration will produce a better solution than a change is becomes a new solution, repeating this process until no further improvements can be found.

5. STUDY PLANNING

We focused on ten online projects. Since in our proposed method, first of all we have create database in the MATLAB in which we have define different functions of the project than we our calculating the functional importance of each function by using automated slicing technique on the basis of number of functions executed and number of attached functions. After finding out the importance of each function, the Fitness value (FTV) of each function is calculated corresponding to each change and then this fitness value is being used in the Hill climbing algorithm in order to increase the fault detection rate corresponding to each change. Based upon this fault detected value test cases are prioritize in the descending order and are executed one by one corresponding to each change. After the fault is being detected, Newman Discrete Technique is being applied in order to remove the fault from the test cases. At last the results of the proposed technique is being compared with existing technique, Weight based prioritization technique and with the test case prioritization using Prime's algorithm and it is found that the results of our technique is more efficient than other techniques. Figure 5 represents activity diagram for one of the considered projects.

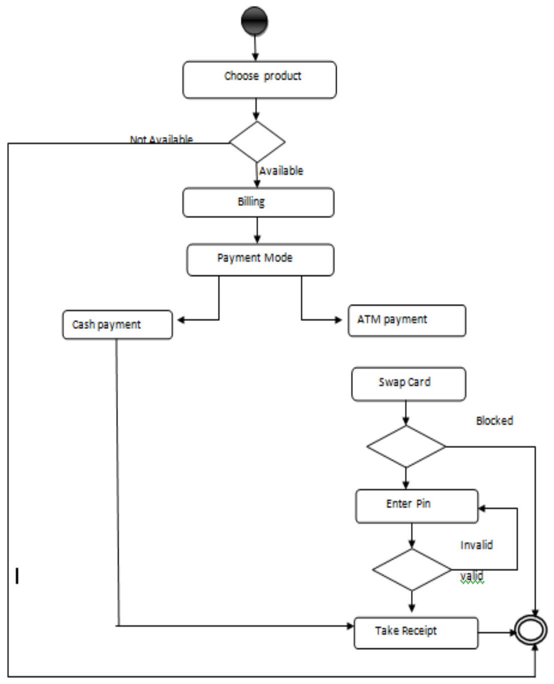


Fig:5 Activity Diagram of Shopping Website

6. RESEARCH WORK

Our research work is used for the Prioritization of Test Cases based upon the Functional Dependency technique by using Hill Climbing algorithm in order to increase the fault detection rate and performance can be analyzed by APFD (Average Percentage Fault Detection) metric. In this work the formula for hill climbing algorithm is as follows:

Probability = (Number of Times Function Executed * Number of Associated Functions) + (Fitness/Sum)

Algorithm for Hill climbing Approach is as follows:

N= number of test cases

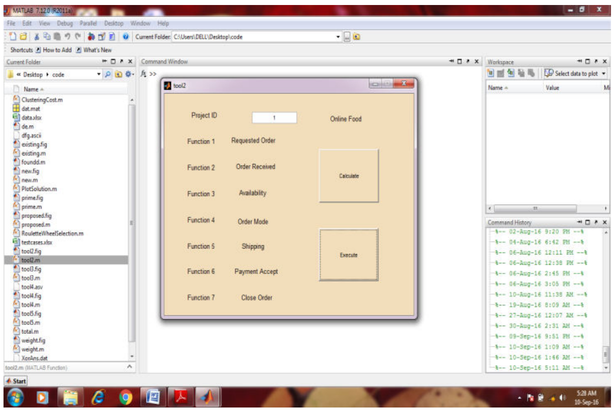
P= Population of member

```
1. Let memory available memory be M
2. If memory (M==full)
   {
Remove unused data from memory M;

   Else
   {
   Load testcases into main memory M;
   }
3. Apply hill clining algorithm to calculate fitness
   on testcases ()
a. For (i=0;i=N;i++)
b. Sum=fitness function of each test case
c. End
d. For (j=0;j<p;j++)
e. Probability =sum of probabilities +(fitness/sum)
f. Sum of probabilities=probability
g. End
h. While (population==full)
i. Number =random between 0 and 1
j. For (j=0;j<p;j++)
k. If(p(i)<p(i+1)
l. Selected=p(i+1)
m. End
4. Define L;
         {
     L=L+1; % for traversing the test cases
     }
     5. If (In L==fault)
     {
     Count =count+1;
     Else
     {
     Count=count;
     }
     6. Display count ;
```

7. RESULTS

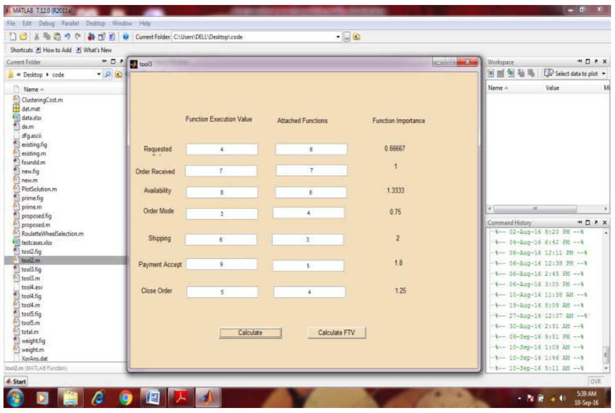
In this figure we have consider a project of online food and seven functions of the project are taken into account in order to calculate importance of each function.



Project ID	1	Online Food
Function 1	Requested Order	
Function 2	Order Received	
Function 3	Availability	
Function 4	Order Mode	
Function 5	Shipping	
Function 6	Payment Accept	
Function 7	Close Order	

Fig:6 Functions of Project

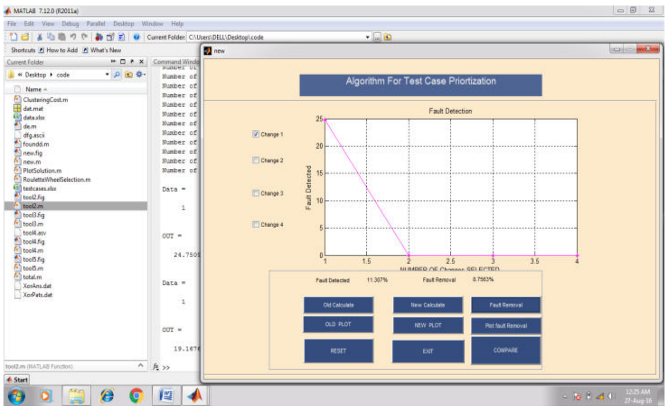
In this figure we have calculate the importance of each function which is based upon the number of times each function is executed and no of associated functions.



	Function Execution Value	Attached Functions	Function Importance
Requested	4	6	0.66667
Order Received	7	7	1
Availability	8	6	1.3333
Order Mode	3	4	0.75
Shipping	6	3	2
Payment Accept	9	5	1.8
Close Order	5	4	1.25

Fig:7 Function Importance

In this work, two algorithms are implemented in MATLAB. The proposed algorithm will be based on functional dependency and it is been analyzed that efficiency of proposed algorithm is high in terms of test case prioritization.



Fault Detected 11.307% Fault Removal 8.7563%

Fig:8 Tool develop

As shown in figure 9, existing and enhanced algorithms are compared and it is been analyzed that proposed algorithm is more efficient in terms of fault detection in order to execute test cases.

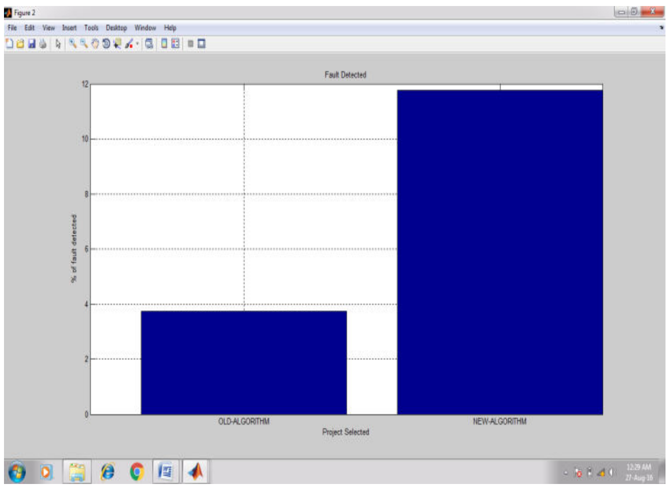


Fig:9 Comparison

8. METHODOLOGY

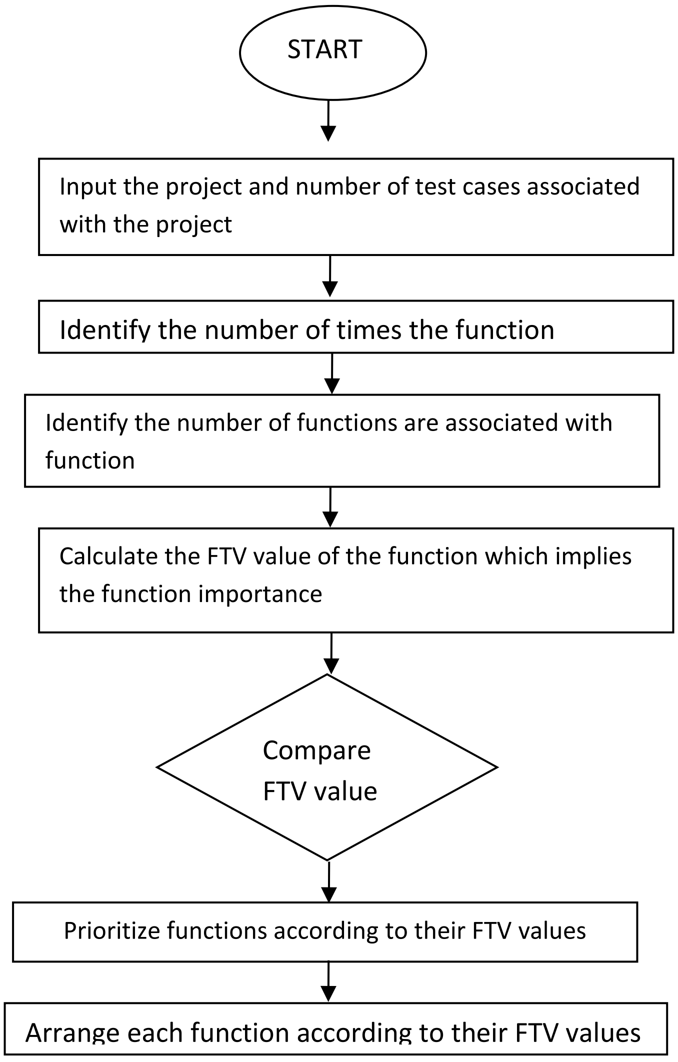

START
Input the project and number of test cases associated with the project
Identify the number of times the function
Identify the number of functions are associated with function
Calculate the FTV value of the function which implies the function importance
Compare FTV value
Prioritize functions according to their FTV values
Arrange each function according to their FTV values

Fig:10 Methodology

9. CONCLUSION AND FUTURE WORK

Test case prioritization is continuously proven to be beneficial technique for Regression testing. In this paper, we have proposed a dependency analysis based upon test case prioritization technique for regression testing and it has been concluded that Hill Climbing Approach is most efficient algorithm to prioritize the test cases by using functional dependency technique. This technique prioritizes test cases based on the dependency structure of the test suites. We have also validated our results with the standard APFD (Average Percentage Fault Detection) metric. Future work included the formalize dependence analysis and study of more efficient algorithms. Also, more cases must be taken into account to verify this work.

ACKNOWLEDGMENT

I wish to express my appreciation and gratitude to the number of people who have helped me in gathering useful information for my work.

REFERENCES

1. H. Leung and L. White, "Insight into regression testing", In Proc. 27th IEEE International Conference Software Engineering, vol. 20, **(1989),** pp. 60-69.
2. S. Khan and A. Awais, "TestFilter: A Statement-Coverage based test case reduction technique", In Proc. 12th IEEE International Conference Engineering, Vol.11, **(2006)**, pp. 5–12.
3. Z. Harman and R. Hierons, "Search algorithms for regression test case prioritization", In Proc. 12th IEEE International Journal Software Engineering, Vol.33, **(2007)** ,pp. 225–237.
4. Korel, G. Koutsogiannakis and L. Tahat, "Application of System Models in Regression Test Suite Prioritization", In International Conference on Software Maintenance, **(2008)**, pp. 247-256.
5. S. S. Hou, L. Zhang, T. Xie and J. S. Sun, "Quota-Constrained Test-Case Prioritization for Regression Testing of Service-Centric Systems", In the 24th International Conference on Software Maintenance, **(2008)**, pp. 257-266.
6. B. Jiang, Z. Zhang, W. K. Chan and T. H. Tse, "Adaptive Random Test-Case

Prioritization", In the 24th International Conference on Automated Software Engineering, **(2009)**, pp. 257-266.

7. P. Tonella and P. Avesani P, "Using the Case-Based Ranking Methodology for Test Case Prioritization" , In 22nd IEEE International Conference on Software Maintenance, **(2009)**, pp.157-162.
8. S. Khan and A. Nadeem, "TestFilter: A Statement-Coverage Based Test Case Reduction Technique" , IEEE Conference on multitopic, (INMC' 06), **(2009)**, pp.275-280.
9. L. Zhang, S.-S. Hou, C. Guo, T. Xie and H. Mei, "Time- Aware Test-Case Prioritization Using Integer Linear Programming", In International Symposium on Software Testing and Analysis, **(2009)**, pp. 213-224.
10. Z. Harman and R. Hierons, "Test case prioritization: An empirical study", in Proc. IFIP International Conference Testing Software System, Vol.35, **(2009)**, pp. 85–90.
11. M. Nada and A. Salami, "Evolutionary Algorithm Defintion" , American Journal of Engineering and Applied Sciences, Vol.2, **(2009)**, pp. 789-795.
12. Askarunisa, L. Shanmugapriya and N. Ramaraj, "Cost and Coverage Metrics for Measuring the Effectiveness of Test Case Prioritization Techniques" , INFOCOMP Journal of Computer Science, **(2009)**, pp. 1-10.
13. J. Daengdej, "Test case prioritization techniques," in Proc. IEEE International Journal Software Engineering Knowledge Engineering, Vol. 22, **(2010)**, pp. 161–183.
14. S. Mirarab, "The effect of time constraint on test case prioritization" , IEEE Transaction on Software Engineering ,VOL.36, No.7, **(2010)**, pp.85-91.
15. R.Malhotra R., A. Kaur A. and Y. Singh Y., "A Regression Test Selection and Prioritization Technique" , Journal of Information Processing Systems, Vol.6, No.2, **(2010)**, pp.167-171.
16. E.Engstrom and P. Runeson, "A Qualitative Survey of Regression Testing Practices" , Springer-Verlag Berlin heidelbreg, LNCS 6156**, (2010)**, pp. 3-16.
17. Hyunsook andS. Mirarab, "The effect of time constraint on test case prioritization" , IEEE TRANSACTION ON SOFTWARE ENGINEERING ,VOL.36 , **(2010)**, pp. 145-151.
18. H. Srikanthi and J. Williams, "System test case prioritization of new regression test case" , IEEE Transaction on Software Engineering ,VOL.36, No.2, **(2011)**, pp.87-94.
19. Kaur and S. Goyal, "A genetic algorithm for regression test case prioritization using code coverage" , International journal on computer science and engineering 3.5 , **(2011)**, pp. 1839-1847.
20. S. Yoo and M.Harman, "Regression testing minimization, selection and prioritization: a survey" , International journal on computer science and engineering, **(2012)**, pp. 67-120.
21. J. Hwang, "Selection of regression system tests for security policy evolution" , Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering. ACM, **(2012)**, pp.69-74.
22. X. Zhang and G. Uma, **"**Factors Oriented Test Case Prioritization Technique in Regression Testing using Genetic Algorithm", European Journal of Scientific Research, Vol. 74, **(2012)**, pp. 34-37.
23. H. Mei, D. Hao, L. Zhang, L. Zhang, J. Zhou and G. Rothermel, "A Static Approach to Prioritizing Junit Test Cases", IEEE Transactions on Software Engineering, vol. 38, no. 6, **(2012)**, pp. 1258-1275.
24. L. Zhang, D. Marinov, L. Zhang and S. Khurshid, "Regression Mutation Testing", In International Symposium on Software Testing and Analysis, **(2012)**, pp. 331-341.
25. S. Yoo and M. Harman, "Regression Testing Minimisation, Selection and Prioritisation: A Survey", Software Testing, Verification and Reliability, vol. 22, no. 2, **(2012)**, pp. 67-120.
26. Jatain and G. Sharma, "A systematic review of techniques for Test case prioritization" , International Journal of Computer Applications, Vol. 68, **(2013)**, pp. 132-135.
27. M. Athar and L. Ahmad, "Maximize the Code Coverage for Test Suit by Genetic Algorithm" , International Journal of Computer Science and Information Technologies, Vol.5, **(2014)**, pp. 431-435,.
28. H. Wang , J. Xing and Q. Yang Q, "Modification Impact Analysis based Test Case Prioritization for Regression Testing of Service-Oriented Workflow Applications" , in Proc. 39th IEEE International Journal Software Engineering Knowledge Engineering, Vol. 30, No. 8, **(2015)**, pp.67-72.