International Journal of Technical Innovation in Modern Engineering & Science (IJTIMES)
Impact Factor: 3.45 (SJIF-2015), e-ISSN: 2455-2585
Volume 4, Issue 5, May-2018

FINDINGS AND IMPLICATIONS OF TEST CASE PRIORITIZATION TECHNIQUES FOR REGRESSION TESTING

Sarika Chaudhary
Computer Science & Engineering & Amity University,sarikacse23@gmail.com

Abstract*— Software makes headway with time which results in an expanded number of test cases. Software testing and maintenance consumes 50% cost of the overall software development process. This is imperceptible to re-execute the complete test suite as and necessary to evolve the software keeping in mind the end goal to minimize the cost and efforts. Regression (moving in reverse) is an unavoidable truth in programming frameworks. Despite the fact that something worked previously, there is no certification that it will work after the most recent "minor" change. Regression testing is carried out during maintenance phase to confirm whether faults are eliminated or not after the software experiences alterations or modifications and ingest 80% cost. It is the technique to select, minimize and prioritize test cases with the specific end goal to permit testers to detect faults as early as possible during the maintenance phase. So, optimizing regression testing is the prime objective of any system testers with a specific end goal to minimize the overall cost. Test case prioritization is a technique to schedule test cases so that test case with higher priority executes first thus, enhancing the performance objective of early fault detection. In this work, we present findings and implications of different code based and customer based test case prioritization techniques and also suggest soft clustering technique for effective cluster formation based on dependency between test case functions and faults experienced.

***Keywords*— code coverage, customer requirements, prioritization, regression testing, test case**

I. INTRODUCTION

Software testing is carried out to verify and validate the requirements. The performance goal of testing is to ensure that the software is behaving as expected and also to extricate the unseen errors. If testing is not done in a formalize way it will not be able to detect error and also result into an increased cost and effort and thus resulting a delay in software delivery.

In today's changing business and commercial environment quality of a product is a major concern and for developing customer requirement based software's, testing plays an important role in evaluating the functional and timing correctness. Effective testing results in improved quality of software thus automatically gaining customer's confidence. Software maintenance is defined as the set of activities that are performed when software is released for use. Software undergoes continuous changes during its life cycle. These progressions are accountable of different reasons like change in requirements, enhancement in existing requirement or functionality, fixing faults or adapting to a new environment.

Regression testing is a testing technique that re-executes software with the intent of finding additional error or faults that are introduced during the process of modification or fixing some already existing bugs. Regression test suites contain already developed test cases with a combination of test case generated after modification. Therefore it is infeasible to re-execute all the test case thoroughly because it results in substantial increase in time and effort thus incorporating the additional cost and also reduce the efficiency of testing process. By virtue of these reasons researches designed various methods for minimizing the cost of regression testing. These include minimization of test cases, selection of test cases and then prioritization of test cases. Regression testing can be classified in following categories:

1) *Adaptive regression testing:* It is a process of re executing the test cases when software is upgraded to a new execution environment.

2) Continuous regression testing: It is a process of re execution of test cases to find out the faults that arise when new requirements or functionalities are added or enhanced in the existing software.

3) Corrective regression testing: It is defined as the testing that uncovers the errors and bugs when slight modifications to the existing code are done.

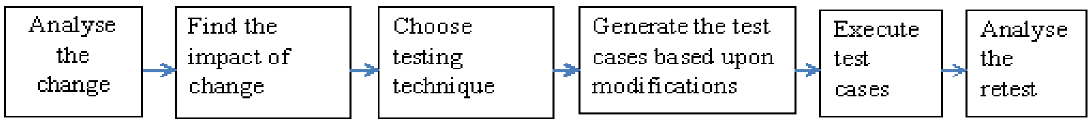

Fig. 1 Regression testing phases

A. Test Case

A test case is defined as a set of rules that can be utilized to verify and validate the requirements specified by the customer. A test case includes input value, expected behaviour and the observed behaviour. In order to check whether the system is working correctly or not the expected behaviour and observed behaviour are evaluated ad if the result comes out to be the same then it is concluded that system is working correctly otherwise not. While designing a test case it must be keep in mind that it should react to both positive and negative conditions for 100% successful testing.

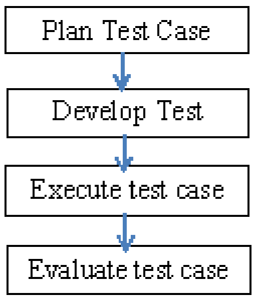

Fig. 2 Test case process

B. Regression testing techniques

1) Regression test selection: In this technique it is appreciable to select test cases related to the part of software that have undergone modifications instead of re executing the complete test suite. Regression test selection can be carried out where the requirements have changed and also where they have not changed. In the former case it is mandatory to figure out the test cases that are obsolete to the changed requirements before performing test case selection. There are various techniques for test selection like minimization technique, dataflow technique, safe technique, ad hoc technique/random technique and retest all.

2) Test case prioritization: It is defined as a process of scheduling and prioritizing the test cases in an order so that the performance objective of an expanded rate of early fault detection can be achieved. Also effective test case prioritization can reduce the time and cost associated with the testing process during maintenance phase of SDLC.

C. Advantages of test case prioritization are:

1) Improved rate of prior fault detection.

2) Improved rate of code coverage in the framework under test at a quicker rate

3) Better confidence in reliability of system.

4) Improved rate of customer satisfaction.

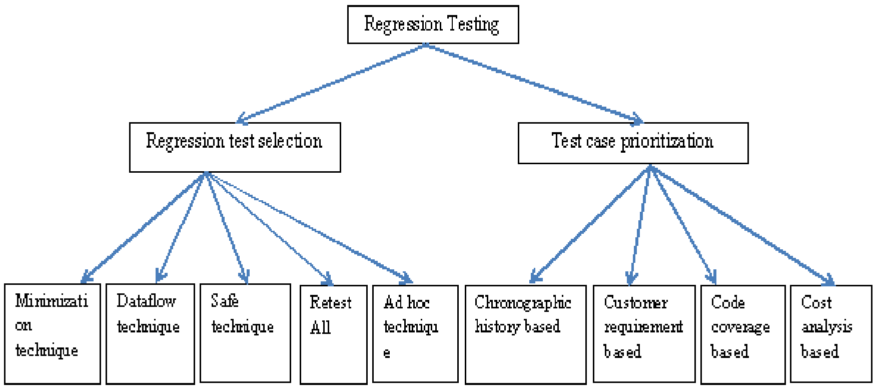

Fig. 3 Regression testing process

D. Techniques of test case prioritization

1) Customer requirement based technique: It is a method of prioritizing test cases on the basis of requirements specified by the customers. In this different requirement factors based on customers are taken into account and some weights are assigned to each factor and then analyzed continuously. Based on these values, weighted prioritization value is computed and test cases are prioritized.

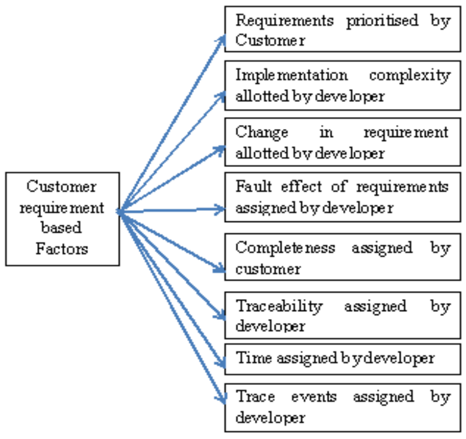
Customer requirement based Factors
Requirements prioritised by Customer
Implementation complexity allotted by developer
Change in requirement allotted by developer
Fault effect of requirements assigned by developer
Completeness assigned by customer
Traceability assigned by developer
Time assigned by developer
Trace events assigned by developer

Fig. 4 Customer requirements based factors

2) Code-coverage based Techniques: This technique is type of white box testing as it involves prioritizing test cases based on the analysis of how much lines of code is covered by each test case. The higher the combined number of statement covered, number of loops executed, number of branches executed, number of conditions executed, higher is the priority of test case. Therefore higher priority test case required to be executed first to achieve the performance objective.

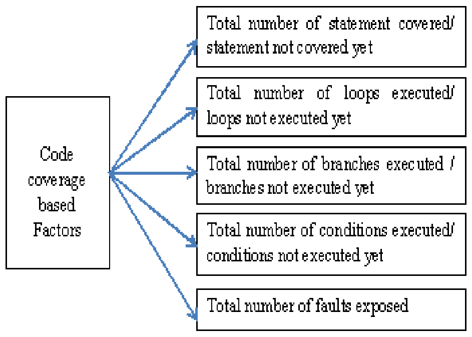
Code coverage based Factors
Total number of statement covered/ statement not covered yet
Total number of loops executed/ loops not executed yet
Total number of branches executed / branches not executed yet
Total number of conditions executed/ conditions not executed yet
Total number of faults exposed

Fig. 5 Code coverage based factors

3) Cost analysis based prioritization techniques: This technique provide a method for prioritizing test case based on cost factors associated with each test case. There are various direct and indirect cost involved such as execution cost, result analysis cost, test selection cost, fault severity cost, tool cost and overhead cost. In this actual time required to execute the test case manually or automatically and cost consumed in test case execution is evaluated and test cases with higher cost are prioritized first.

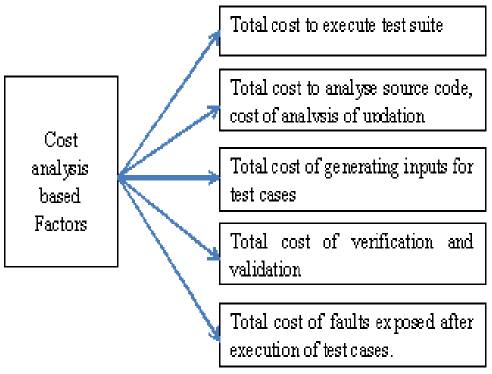
Cost analysis based Factors
Total cost to execute test suite
Total cost to analyse source code, cost of analysis of updation
Total cost of generating inputs for test cases
Total cost of verification and validation
Total cost of faults exposed after execution of test cases.

Fig. 6 Cost based factors

4) Chronographic history based prioritization techniques: This technique defines a method to prioritize test cases on the basis of past execution history of test cases in the current testing environment. It is also known as black box testing.

II. Literature Review

Sebastian Elbaum et. al[1] examined potential of version specific test case prioritization, in which test cases are organized and effectiveness of rate of fault detection is estimated with respect to change in particular version of a

software. In this paper three groups namely control, statement and function level with a set of 14 test case prioritization factors are examined. In order to quantify the objective of increased rate of fault detection average percentage of fault detection (APFD) metric is used. 8 'C' programs with faulty version and test cases are utilized as source. The results show that prioritization technique can enhance the rate of fault discovery in contrast to the one without prioritization technique.

Siripong and Jirapun [2] first introduced the 4C's prioritization types and they are customer requirement based, code coverage based, cost analysis based and chronographic based. Many existing approaches of test case prioritization failed in prioritizing multiple test suite and test cases with same priority values thus making them inefficient for large commercial systems. This paper proposed two novel efficient methods to solve the issues discussed above. The metric used to validate the results are HPRE (high priority reserved effectiveness), size of acceptable test case and TPT (total prioritization time).

Shin and Mark [3] has introduced a hybrid interleaved cluster based prioritization technique and proven that pair wise comparison between the test cases can be reduced to a significant amount by utilizing this. Analytic Hierarchy process (AHP) algorithm is a decision making tool that help in prioritizing tasks. In this paper AHP based technique is compared with coverage based prioritization and the results show that AHP is robust and can outperform later one.

Thillaikaras and Seetharaman [4] stated that the most important performance objective in regression testing is at what rate the fault is detected. Many researchers have analyzed the coverage based prioritization techniques but customer requirement based techniques has not been evaluated in a cost effective manner yet. This paper introduced a model that prioritized the test cases based on six factors to increase the rate of fault detection. More practical weights are assigned to each test case and APFD metric is used to validate the algorithm.

Raju and Uma [5] proposed a new clustering based prioritization technique accounting the dynamic run time behavior of test cases. In this prioritization of test cases is done with the help of 4 requirement factors: rate of fault detection, requirement volatility, and fault impact and implementation complexity. The results demonstrate that this technique will reduce the re-execution time of project as number of pair wise comparisons are minimized and the rate of detection of severe fault is enhanced.

Kavitha and Suresh kumar [6] suggested that a prioritizes test suite is more effective than a random one. Total of 8 factors based on system requirement test case are considered but the focus is on trace event. It is defined as the maximum number of times each test case is executed. Based upon the requirement factor value, test case weights are calculated and the test case with more weight is executed first. The proposed technique significantly decreases the cost of computation and time.

Deepak Garg et. al [7] suggested that with every cycle of regression testing there is an increase in test cases in a test suite. Due to modifications many test cases become obsolete and therefore execution of these obsolete test cases results in reduced test coverage and increased test execution time. In this paper a new bipartite graph approach is used to map between the modified source code and test case, thus eliminating the subsets of obsolete test cases. This approach help in improving the execution time and also minimizing the cost of execution as redundant test cases are not in picture any more.

Indumathi and Selvamani [8] suggested that prioritizing test cases aids in meeting two crucial factors of software success i.e time and budget. This demands us finding the fault earlier. Existing techniques namely Random order, greedy technique, genetic algorithm, cluster based as well as fine grained and coarse grained have not taken into account the dependency between test cases. Existing techniques manually find out dependencies. In this paper set of algorithms are proposed for automatic dependency detection among different functions with count. Highly dependent functions are assigned highest priority using graph coverage values and executed first, thus the rate of fault detection will be improved at earlier stage and thus helping in early removal of fault.

Imrul Kayes[9] proposed a new metric Average Percentage of Fault Dependency Detected (APFDD) to measure the effectiveness of test case prioritization technique. The results show that the using this new metric, test cases are more effective in detecting dependency among faults. But in practical world its effectiveness is minimized as this paper considers fault severity and execution time to be uniform.

Prem and Ravi [10] proposed a technique of test case prioritization by utilizing genetic algorithm. In this code coverage factors that covers the most number of lines of source code is taken into consideration and the test case that covers maximum code coverage given highest priority an executed first. To find out the maximum coverage a fitness function is calculated. This technique will provide near an optimal solution most of the time.

Muhammad M. [11] proposed a test suite prioritization scheme, t-SANT, that works by utilizing the application usage from tester as well as user point of view and extract the sequences that occur frequently using a sequence mining algorithm. In this paper a prioritization algorithm is derived that prioritize the test cases without human intervention based on the longest sequences of interactions. After that a fault seeding approach is used to measure the effectiveness of the technique.

Sarabjit and Saloni [12] suggested an enhanced hill climbing algorithm based on functional dependency for test case prioritization. The fitness value of functions are calculated and based upon these values prioritization is done. The proposed method yield in better accuracy of fault detected and reduction in execution time. For experimental analysis 10 online projects are considered. The metric used to validate the result is APFD. The result analysis shows that the proposed hill climbing algorithm is the most efficient in comparison to other existing techniques.

III. COMPARATIVE ANALYSIS

On the basis of the literature review for test case prioritization techniques published in [1][2][3][4][5][6][7][8][9][10][11][12] , systematic comparisons are made on the basis of key ideas, metrics used and outcomes as shown in fig. 7. The comparative analysis utilizes the predefined criteria of prioritization type. There are several other techniques proposed in the past, but, the most recent are discussed in this paper. Every technique inhibits pros and cons. Tester can pick any one depending upon the scenario of the project and its objective.

TABLE I

Sr.No.	TECHNIQUE PROPOSED BY	TYPE OF PRIORITIZATION	KEY IDEAS	METRIC USED	OUTCOMES
1	Sebastian Elbaum, Alexey G. Malishevsky and Gregg Rothermel	Code coverage based	1) Version specific test case prioritization. 2) 14 factors categorised into 3 groups: contol, statement and function level groups. 3) Control technique: Random ordering. Optimal ordering. 4) Statement level techniques: Total statement coverage. Additional statement coverage. 5)Total fault exposing potential(FEP) Prioritization	APFD	1) Improved the rate of fault detection. 2) If the cost of delays in detecting fault is sufficiently high then statement level techniques can produce better results. 3) Not considered real commercial projects into account.
2	Siripong Roongruangsuwan and Jirapun Daengdej	Customer requirement based	1) Introduces the 4 C's type of test case prioritization. 2)4 factors i.e cost , time ,defect, and complexity with practical weights are considered.	HPRE, SAT, TPT.	1) Problem of multiple test cases with same weights is resolved. 2) Many test suite prioritization can achieved. 3) Real commercial data is not taken into account. 4) Scope in improving the capability of automatically finds redundant test cases with same values.
3	Shin Yoo and Mark Harman	Code coverage based	1) Incorporate expert knowledge via AHP to reduce the cost of human interactive prioritization. 2) Introduced a hybrid interleaved cluster based prioritization technique	APFD	1) AHP based prioritization can outperform coverage based techniques. 2) Number of pair-wise comparison is reduced because of clustering. 3) Improved rate of fault detection.
4	Thillaikarasi Muthusamy and Dr. Seetharaman.K	Customer requirement based	6 factors are considered: customer alloted priority,developer obseverd code implemntation complexity, change in requirement,fault impact of requirement,completeness,traceability	APFD	1) Improved rate of fault detection as practical set of weight factors are used. 2) Risk factors are not utilized.

5	S.Raju and G.V.Uma	Customer requirement based	1) 4 factors are considered: Rate of fault detection (RF), Requirement volatility (RV), and Fault impact (FI), Implementation complexity (IC). 2) Clustering and PORT (prioritization of requirements for test) based prioritization.	APFD and PTR	1) Beneficial for small applications. 2) Time aware prioritization. 3) Reduction in execution cost.
6	Kavitha rajarathinam and Sureshkumar Natarajan	Customer requirement based	1) 8 factors are considered: customer assigned priority; developer perceived code implementation complexity, changes in requirements, fault impact of requirements, completeness, traceability, time and trace events. 2) trace event based prioritization	APFD	1) Improved rate of severe fault detection. 2) Reduced number of test cases. 3) Cost of execution time is minimized.
7	Deepak garg, Amitava Datta and Tim French	code coverage based	1) User defined components in source code are considered. 2) These components are mapped with test cases by utilising bipartite graph. 3) Test cases required with respect to current versions are selected.	APFD	1) General approach that can be applied to any software application. 2) Independent of test tool and applicable to any test case format. 3) The modified source code that has not been tested earlier has more tendencies to introduce faults.
8	Indumathi CP and Selvamani K	Code coverage based	1) Dependency structure among test cases is extracted. 2) Test cases are grouped into coarse grained test to handle test cases dependency. 3) Level ordering is done for computing the exact number of dependents for each test case. 4) Highly dependent test cases are given more priority and executed first.	APFD	1) Extracts the dependency structure automatically among the test cases. 2) Prioritize the test cases within a very short period time. 3) Increased rate of fault deduction at earlier stage.
9	Md. Imrul Kayes	Code coverage based	1) Proposed a new metric APFDD (average percentage of fault dependency detection). 2) It considers fault severity and test case execution time to be uniform. 3) Test cases are prioritized based upon dependencies in fault exercised.	APFDD	1) Prioritized test cases are more effective. 2) Faster feedback is provided on the basis of fault dependency that helps developers to resolve severe faults that may lead to other faults. 3) Not suitable for practical world as execution time and fault severity cannot be uniform.
10	T.Prem Jacob and T.Ravi	Code coverage based	1) Test cases are selected based upon the modified lines covered by the test case. 2) Genetic algorithm is used to prioritize the test cases.	Fitness Function	1) Improved rate of fault detection. 2) Reduced number of test cases. 3) Cost of execution time is minimized.
11	Muhammad Muzammal	Customer requirement based	1) A new technique t-SANT (test suite prioritization by application navigation tree mining) is proposed. 2) Prioritize test cases based on user and tester perspective of using the application. 3) Test cases with frequent longest sequences are given priority.	Fault seeding approach	1) Can work even without human intervention. 2) No real data is taken into account. 3) Mapping process have scope to be more formalize thus enabling t-SANT to work efficiently with large applications.
12	Sarabjit Kaur and Saloni Ghai	code coverage based	1) Utilises hill climbing using dependency analysis. 2) Automated slicing technique is used to calculate the functional importance of each function. 3) Newman Discrete technique to remove faults.	APFD	1) Real commercial test data is taken into account. 2) Efficient than weight based and prioritization using Prime's algorithm techniques. 3) Improved rate of early fault detection.

IV. CONCLUSIONS

Test case prioritization is the most beneficial technique of regression testing. Test case need to be prioritized before execution thus enabling early fault detection and minimizing the cost of execution. Various techniques of test case prioritization are discussed here and it is concluded that mostly code coverage prioritization techniques are used as shown in fig.7. A very few approaches used customer requirement based and cost analysis based prioritization techniques with limited number of parameters. Real commercial data is not used in past except the hill climbing approach, all researches are carried out by taking simple projects. So, more real test cases must be taken into consideration to validate the fault free operation of real commercial projects. Although the concept of functional dependency between test cases utilizing the code coverage factors is introduced but it need to be more formalize using customer requirement based factors and code coverage based factors, as well defined requirements contributes in high quality of software thus gaining customer confidence in software. Existing researches use the concept of hard clustering i.e they assumes every test case is divided into different clusters , where each test case can strictly belong to a single cluster,. But this lacks effectiveness as it is desirable that a test case can potentially belong to multiple clusters when functional dependency between different functions of test case and different faults is found out. Future work includes the use of more efficient clustering techniques like fuzzy clustering and study of more optimized prioritization algorithm for complex industrial applications.

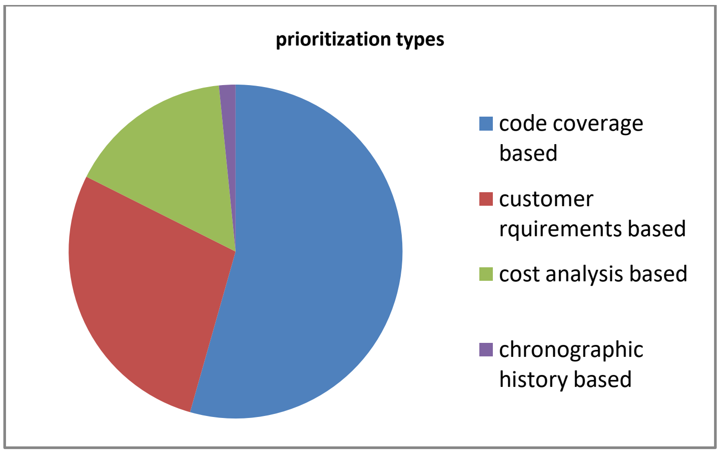

Fig. 7 Ratio of prioritization types exploited so far

REFERENCES

[1] S. Elbaum, G. Malishevsky, G. Rothermel,"Prioritization Test Cases for regression Testing" , *Proc. The 2000 ACM SIGSOFT International symposium of Software testing and Analysis, Portland*, Oregon, U.S.A., pp. 102-112, 2000.

[2] S. Roongruangsuwan and J. Daengde,"Test Case Prioritization Techniques", *Journal of Theoretical and Applied Information Technology*,Vol-18,2010.

[3] S.Yoo and M.Harman,"Clustering Test Cases to achieve effective & scalable prioritization incorporating expert knowledge" ,*International Symposium on Software Testing and Analysis Chicago*, IL, USA- 2009,ACMNew York, NY, USA.

[4] T. Muthusamy and K. Seetharaman, "Measuring The Effectiveness of Test Case Prioritization Techniques Based on Weight Factors", *Computer Science & Information Technology (CS & IT)*, Vol-4, pp. 41–51, 2014.

[5] S. Raju, G.V. Uma, "An efficient method to achieve test case prioritization in regression testing using prioritization factors", *Asian Journal of information Technology*, Vol-11, issue 5, pp.169-180, 2012.

[6] K. Rajarathinam and S. Natarajan, " Test suite prioritization using trace events technique", *The institute of engineering and Technology Software*, vol.7,issue 2,pp 85-92, 2013.

[7] D. Garg, A. Dutta, and T. French, " A novel bipartite graph approach for selection and prioritization of test cases", *ACM SIGSOFT Software Engineering Notes*, Vol 38, pp.1-6, 2013.

[8] C. P. Indumathi and K. Selvamani,“ Test Cases Prioritization using Open Dependency Structure Algorithm”, *ELSEVIER, Procedia Computer Science* , Vol-48 ,pp-250 – 255, 2015.

[9] I. Kayes, “Test case prioritization for regression testing based on fault dependency”, IEEE 2011.

[10] P. Jacob and T. Ravi, “ Optimization of test cases by prioritization”, *Journal of Computer Science*, Vol.9, Issue 8,pp. 972-980, SCI Publication ,2013.

[11] M. Muzammal, “Test suite prioritization by application navigation tree mining”, International *Conference on Frontiers of Information Technology*, pp.205-210, IEEE, 2016.

[12] S. Kaur and S.Ghai ,“ Performance enhancement in Hill climbing approach for test case prioritization using functional dependency technique”, *International Journal of Software engineering and its Applications*, vol.10, No. 11, pp. 25-38, 2016.