Effectiveness of Prioritization of Test Cases Based on Faults

Soumen Nayak, Chiranjeev Kumar, Sachin Tripathi
Department of Computer Science and Engineering
Indian School of Mines
Dhanbad, India
soumen.nayak@gmail.com, kumar.c.cse@ismdhanbad.ac.in, var_1285@yahoo.com

***Abstract*—Regression testing (RT) is an expensive activity. It is applied on a modified program to enhance confidence and reliability by ensuring that the changes are accurately true and have not affected the unmodified portions of the SUT. Due to limited resources, it is not practical to re-run each test cases (TC). To improve the regression testing's effectiveness, the TCs should be arranged according to some objective function or criteria. Test case prioritization (TCP) arranges TCs in an order for execution that enhances their effectiveness by satisfying some testing goals. The highest priority assigned to TCs must execute before the TCs with low priority by virtue of some performance goal. Numerous goals are possible to achieve of which one such goal is rate of fault detection (RFT) in which the faults are surfaced as quickly as possible within the testing process. In this paper, a novel technique is suggested to prioritize the TCs that increase its effectiveness in detecting faults. The effectiveness of the proposed method is compared and matched with other prioritization approaches with the help of Average Percentage of Fault Detection (APFD) metric from which charts have been prepared.**

Keywords—Regression Testing, TCP, APFD, Severity of Faults.

I. INTRODUCTION

Software testing is driven largely by the quality goals of the software. It is a continuous process that locates bugs or failures with the objective of having high assured software. The domain of inputs that is possible to the software is very large to test. Also, the rise of attrition rate poses a threat to the software industry. Therefore, effective testing is preferable over exhaustive testing in this era of limited resources like time, cost and effort. Software testing is a stepping stone to 'process improvement'. It must be planned, specified, designed, executed and quantified and it should be in such a fashion that there is enough time for important and critical features of the software. The testing process should be monitored by quantifying the quality goals. RT is one of the most crucial maintenance level activities. It laid emphasis on unearthing new software errors in existing functional and non functional areas of the SUT. These happen after modifications such as new enhancements, or configuration changes like patches etc that have been performed on SUT. It ensures that the changes made to the SUT do not introduce new bugs [1]. It also determines that whether a modification in some part of the software influence other part of it.

RT not only provides the reliable mean to verify the code changes but also tracks the quality of the output by checking the modifications that are having effect on integrity of an application's existing functionality. A software development firm has a regression test suite that contains over 30,000 TCs for one of its primary product. It requires over 1,000 machine hours to execute all the TCs. It requires engineer time of more than hundreds of hours to look after this RT process [2]. Whenever software is changed, many new TCs, in addition to the existing one, might be needed to test the new functionalities. By this, as the software evolves so as the TCs grow tremendously. Therefore, TCs occupy a core theme in software testing. It should be created, studied, purposefully used, controlled, and preserved. The reuse of the test suite is prevalent in the software industry.

Re-executing every TC with resource constraints is impractical in business point of view. This problem can be solved by arranging (prioritize) the TCs in some order that could able to meet some testing goals. This ordering follows some test adequacy criteria. A testing criterion is a set of axioms or rules. It sets requirements on a set of TCs. The area to which a criterion is satisfied in terms of branching or coverage is measured by test engineers; a test set succeeded in achieving 100 percent coverage if it totally cover the criterion. Test requirements for statement coverage is that the requirement is each statement; for mutation, the requirement is each mutant; and in data flow testing, the requirement is each DU pair [3]. Coverage criteria are used as a halting point to decide when a program is sufficiently tested.

The techniques used in TCP [4],[5], purposefully schedule TCs in a fashion that try to maximize some objective criterion such as aiming code coverage as fast as possible or enhancing RFT, location of high-risk faults, and likelihood of surfacing regression bugs or errors. It also have impact in controlling budget overrun, product release delays and also on errors that might slip into released product. The test engineer orders the TCs in a way that the highest priority assigned to TCs will run before the lower ones.

Prioritization techniques never discard any TCs, so it can eliminates the demerits that can happen when TC selection and test suite reduction or minimization get rid of test suites. It can increase the likelihood that the testing time have been spent more beneficially in case RT have been abruptly terminated than if TCs that were not prioritized. Earlier feedback is possible when there is an improved or better RFT on a SUT that in turn begins earlier debugging activities than might otherwise be possible.

978-1-4799-8579-1/16/$31.00 ©2016 IEEE

Many testing criteria are beneficial for locating TCs that exhibit different architectural, functional and non-functional scope in a software. Hence, efficient metric should be determined to improve the effectiveness of the TCs. In this paper, one such effective metric for regression TCs is proposed by virtue of which the arrangements of TCs at system level have been done. Higher the metric value, the TC will be given more priority and will be executed earlier than with lower metric value. This proposed approach can identify more faults at the beginning part of the testing activity. This metric suggested to design our algorithm depends on two factors:

- Fault rate, and
- Severity value of the fault

The rest of the paper is arranged as follows: part 2 discusses the TC prioritization problem, part 3 discusses the literature review, part 4 explains the proposed work, part 5 deals with experimentation and analysis, part 6 presents the comparative study and part 7 discusses about the conclusion and future aspects.

II. TC PRIORITIZATION

The TCP problem [6] can be explained by the definition as given below:

It is given that, a test suite, T, the set of permutation of T, PT, and a function f from PT to the real numbers.

The problem is to find T' Є PT such that for all T'', T'' Є PT and T'' ≠ T' [f(T') ≥ f (T'')]

In the above definition, PT denotes the set of all combinations of sorting arrangements of T and f is the objective function, that when put on to any such order returns an award value for that specific ordering.

For simplicity, from the definition it can be assumed that higher award values are preferable to lower ones. There are several possible number of performance criteria for TC prioritization. The motive of this paper is to build a TC prioritization technique that prioritizes the regression testing TCs which can enhance the early detection of faults.

III. LITERATURE REVIEW

TCP typically sorts the existing TCs so as to obtain the desired performance aim. In recent times many researchers and academicians have searched numerous metrics and methods for arranging regression test suites (RTS). Rothermel et al. [4] proposed various TCP techniques and empirically calculated their quality, importance and the amount of the RFT. The result depicts from their research was that the prioritization technique can efficiently detect faults as quickly as possible and also it reflects trade-offs between numerous prioritization techniques.

Rothermel et al [4] and Elbaum et al. [7] propounded APFD metric for measuring the effectiveness of fault detection by several prioritization techniques. The disadvantage of APFD metric is that they consider defect severities to be same and did not consider varying costs of all TCs. To overcome this, Elbaum et al. [8] and Malishevsky et al [9] propounded aother metric $APFD_c$ which considers the varying TC amounts and fault severities into TCP. Muthusamy and Seetharaman [10] present a new metric based on practical priority factors namely fluctuating requirement priorities, priorities of TCs, execution time and severities of faults. The finding reveals that the TC priority rate per unit time can be improved and enhances testing quality and satisfaction of customers. Kavitha et al. [11] considers two metrics: RFT and FI. The efficacy of testing can be even better by following the TCs which detects maximum severe faults. Severity value is assigned to each fault on the basis of fault's impact on SUT.

B. Jiang et al. [12] proposed an ART-based TCP algorithm. This algorithm groups two functions for calculating the distance, which is based on code coverage data, between two TCs and selecting a TC among the resultant candidate set. Each TC choose one TC among the resultant set till all TCs have been covered.

R. Krishnamoorthi et al. [13] ordered the system TCs using six parameters namely: priority of customer, requirements modification, complexity in implementation, that can be usable elsewhere, flow of application and impact of the faults. The result shows there is a improvement in the rate of severe fault detection.

Various search algorithms or meta-heuristic algorithms may be used to find an answer to the TCP problem. Li et al. [14] have done a empirically examined the usage of some greedy, meta-heuristic, and evolutionary search algorithms to order TCs. The result depicts the type of the RT search space where genetic algorithm performs better. Mala et al. [15] consider similarities between artificial bee colony (ABC) optimization with ant colony optimization (ACO) and concluded that the prioritized TC based on artificial bee colony optimization have several advantage over ant colony optimization. Singh et al. [16] used ACO algorithm to prioritize TCs in a time controlled conditions. The efficiency of the suggested method is matched with various other methods by calculating the APFD value and it was observed that it was equal to the optimal ordering but better than the rest.

IV. PROPOSED WORK

This section presents the proposed algorithm and the factors affecting this algorithm.

Prior work shows that a huge amount of time is invested to execute TCs and to order them [9],[17]. The major concern is the voluminous test suite and/or the execution time needed by the TCs to detect faults. Also, there is a dearth of proper algorithms that could optimize the execution time. However, an efficient prioritization technique could able to solve this problem by prioritizing the TCs that will increase the RFT.

A new method is proposed that could effectively arrange the selected TCs based on weighted priority (WP) values. This new RTS prioritization technique prioritize the TCs with the aim of detecting the amount of faults as quickly as

978-1-4799-8579-1/16/$31.00 ©2016 IEEE

possible in constrained environment such as time, effort and cost. Fig. 1 represents the flow-chart representation of the proposed technique. The complexity of the proposed algorithm is $O(n^2)$. The parameters affecting the proposed algorithm and the proposed method for prioritization are discussed below.

A. Parameters Affecting Proposed Work

We take into consideration two factors for our suggested methodology as discussed below.

1 Fault Rate

The fault rate (FR) is defined as the maximum amount of faults located by a TC per unit time or the total execution time. For TC TC_j, FR_j have been calculated using total number of faults, N_j, located by TC_j and the total execution time, $Time_j$, required by TC_j to uncover those faults. It can be expressed in the form of equation as follows.

$$FR_j = N_j / Time_j \qquad (1)$$

2 Fault Severity

The efficiency of testing can be enhanced by giving emphasis on the TC that has been assigned the higher value of severe faults. The fault severity value (FSV) has been computed based on the effect of the fault on SUT. Severity weighted value [10] as in Table 1, have been given legally based on 32 point scale.

TABLE 1 SEVERITY VALUES

Amount of Severity	Severity Notation	Severity Weighted Value
Very High	VHS	32
High	HS	16
Medium	MS	08
Less	LS	04
Least	VLS	02

3 Weighted Priority

The weighted priority (WP) for every j^{th} TC is calculated as product of fault rate and the severity value for each TC. Formally, it is represented as follows.

$$WP_j = FR_j * FSV_j \qquad (2)$$

4 Proposed Methodology for Prioritization

The technique for assigning the value for prioritization or ordering is represented in the algorithmic form here under:

Algorithm: Test Case Prioritization Algorithm

Input: Test suite, TS and factor values like number of faults uncovered, the total amount of time required for execution and the severity value for each test case
Output: Ordered test suite, TS'

Begin

1. *Assign TS' to be void*
2. *for each test case t_j Є TS do*
3. *Compute fault rate (FR) for each test case using Eq. (1)*
4. *Calculate weighted priority (WP) for each test case using Eq. (2)*
5. *end for*
6. *Order TS in lower order of WP values for each test case*
7. *Set TS' be T*

End

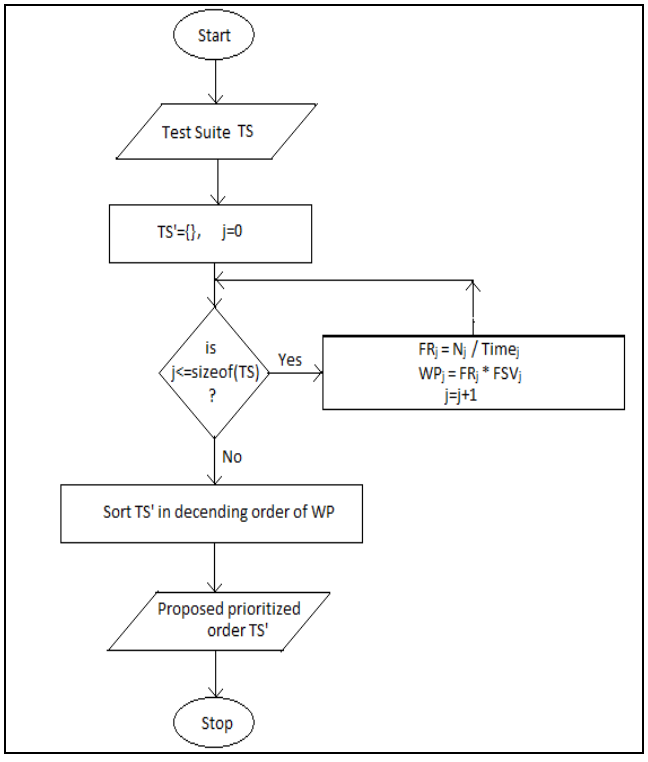

Fig. 1. Flow-chart representation of proposed work

978-1-4799-8579-1/16/$31.00 ©2016 IEEE

V. EXPERIMENT AND ANALYSIS

We have taken into consideration exactly the same test suite for our experimental purpose that is used in Kavitha et al. [11]. Here, they have carried out an experiment on a VB project. They put 10 faults differing in severity level inside the project and finally calculated the required time to detect the faults by every TC. The table 2 depicts the sample data and Table 3 amplifies the values of N_i detected, the execution time and the severity value for each TC.

TABLE 2: TEST CASES (TCs) AS TC1…TC10 AND FAULTS AS FA1…FA10, '+' DEPICTS THE FAULTS UNCOVERED BY THE TC

TCs/ Faults	TC 1	TC 2	TC 3	TC 4	TC 5	TC 6	TC 7	TC 8	TC 9	TC 10
FA1								+	+	
FA2		+	+		+					
FA3				+		+				+
FA4		+	+							
FA5								+		
FA6								+	+	
FA7				+	+		+			
FA8	+					+				
FA9				+		+				+
FA10	+							+		

TABLE 3: N_i LOCATED, THE EXECUTION TIME AND THE SEVERITY VALUE FOR EACH TC

TCs	N_i Located	Time(ms)	Weighted Severity Value
TC1	2	9	6
TC2	2	8	6
TC3	2	14	6
TC4	3	9	10
TC5	2	12	8
TC6	3	14	10
TC7	1	11	4
TC8	4	10	20
TC9	2	10	12
TC10	2	13	6

The value of FR value and the WP value can be calculated using equation (1) and equation (2) respectively. The severity value can be obtained from table 3. Table 4 represents the calculated value of FR and WP for every TC, i.e., TC1…TC10 respectively.

TABLE 4: FR AND WP VALUES FOR TCS T1…T10

Test Case (TC)	FR	WP
TC1	0.222	1.333
TC2	0.25	1.5
TC3	0.143	0.857
TC4	0.333	3.333
TC5	0.167	1.333
TC6	0.214	2.143
TC7	0.091	0.364
TC8	0.4	8
TC9	0.2	2.4
TC10	0.154	0.923

The TCs are ordered and is executed according to the values of WP. The ordering of TCs should be done in a manner that those with higher WP values must be executed before the lower ones. Therefore, the TCP order obtained is TC8, TC4, TC9, TC6, TC2, TC1, TC5, TC10, TC3, TC7. [Note: If two or more values are same, then the TC that arrives earlier in the fault matrix, should be given preference].

VI. COMAPARISON

The test suite effectiveness can be quantified by a metric, APFD, that judges the weighted APFD detected during the execution of a TS. Its value ranges between 0-100, where a higher designated value means faster RFT. If we plot %age of test suite executed on the x-axis and %age of FA located, then the area enclosed within the curve is the APFD value. Also, the APFD can be calculated as given below:

$$APFD = 1- ((TSF1 + TSF2 +\ldots\ldots + TSF_m)/nm) + 1/2n \quad (3)$$

Where, TSF_i is the position of the first test in test suite TS that exposes FA i,
m is the N_i detected, and
n is the total number of TCs in TS.

A. Comparison with Previous Work [11]

In this part, our suggested order sequence is matched with Kavitha et al work[11]. The suggested order of TCs and the prioritized sequence of TCs as given by Kavitha et al. [11] for the same TS is depicted on Table 5.

TABLE 5: TC ORDERING SEQUENCE FOR PROPOSED WORK AND THE EXISTING WORK [11]

Proposed Sequence	Existing Order [11]
TC8	TC8
TC4	TC4
TC9	TC9
TC6	TC6
TC2	TC5
TC1	TC2
TC5	TC1
TC10	TC10
TC3	TC3
TC7	TC7

978-1-4799-8579-1/16/$31.00 ©2016 IEEE

B. Comparative Study

In this part, Table 6 depicts the suggested technique is matched with other prioritization approaches such as no prioritization technique (NPT), reverse prioritization technique (RPT) and random prioritization technique (ROT). In no prioritization, the TCs are ordered in similar fashion as generated. In the reverse prioritization, the TCs are arranged in reverse manner of that generated by no prioritization method. In random prioritization, the TCs are randomly organized.

TABLE 6: SEQUENCE OF TCs FOR DIFFERENT PRIORITIZATION METHODS

NPT Order	RPT Order	ROT Order	Proposed Order
TC1	TC10	TC7	TC8
TC2	TC9	TC1	TC4
TC3	TC8	TC10	TC9
TC4	TC7	TC3	TC6
TC5	TC6	TC8	TC2
TC6	TC5	TC2	TC1
TC7	TC4	TC4	TC5
TC8	TC3	TC5	TC10
TC9	TC2	TC9	TC3
TC10	TC1	TC6	TC7

The APFD %age NPT, RPT, ROT, existing work [11], and proposed sequence ordering is depicted in Figure (2-6) respectively.

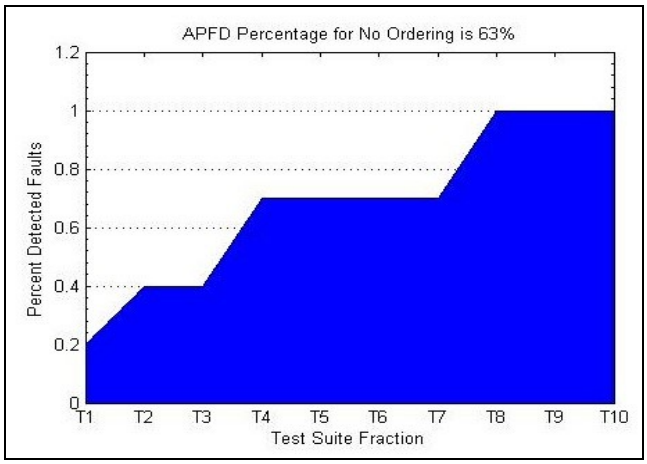

Fig. 2. APFD %age for NPT

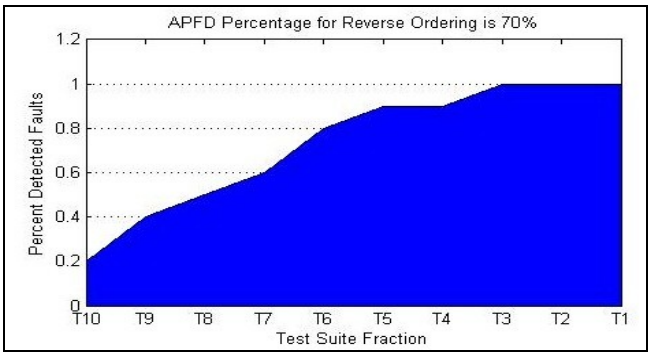

Fig. 3. APFD %age for RPT

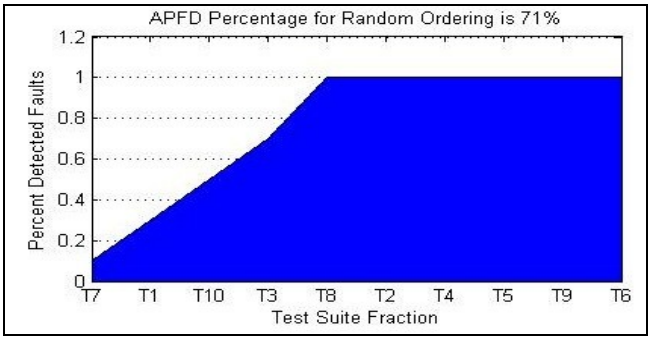

Fig. 4. APFD %age for ROT

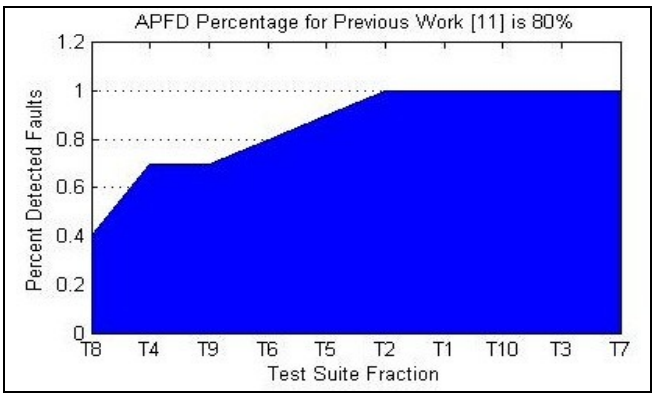

Fig. 5. APFD %age for previous work [11]

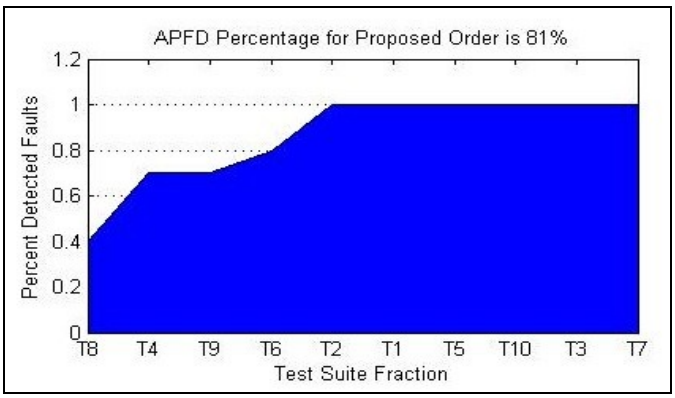

Fig. 6. APFD %age for proposed work

978-1-4799-8579-1/16/$31.00 ©2016 IEEE

C. Discussion

The APFD %age for NPT, RPT, ROP, existing work [11] and suggested sequence ordering are depicted in Table 7.The APFD %age for proposed suggested work is more than other various prioritization techniques that shows its better efficacy. It can be concluded that the suggested approach is better than other prioritization techniques and slightly have higher value than the previous work [11]. The relevance of our proposed approach will be pronounced when the test suite is very large and there is a crunch in resources, when compared with previous work [11].

TABLE 7
APFD PERCENTAGE FOR VARIOUS PRIORITIZATION APPROACHES

Ordering Techniques	APFD Percentage
NPT	63
RPT	70
ROT	71
Existing Work [11]	80
Suggested Order	81

The comparison is done between the non-prioritized and different prioritized TC that can be observed in Fig. 7, which emphasis that the new suggested prioritization technique requires only 50 % of TCs to detect all FAs which is same as random order. But 80 % and 60 % of TCs are required to detect all the FAs in case of no order , which is same as reverse order, and existing work [11] respectively. By comparing the two conditions, APFD value and %age of TCs executed to detect all the FAs, the proposed approach performs better than the rest of the techniques.

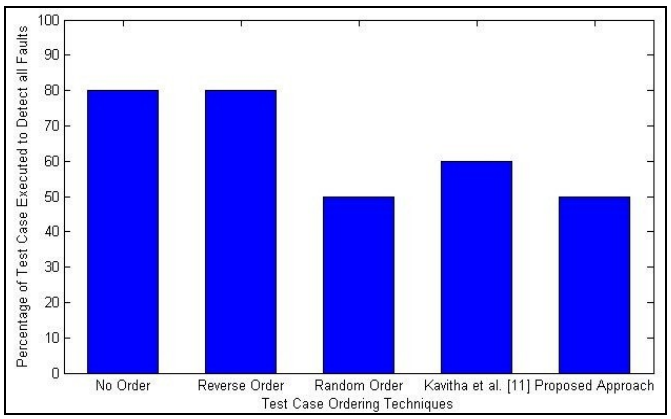

Fig. 7. Percent of TC needed by different ordering techniques to detect all FAs in the TS

VII. CONCLUSION

In this manuscript, a new prioritization algorithm is proposed to increase the RFT by prioritizing the TCs for RT based on a metric, Weighted Priority (WP). The proposed work is compared with various prioritization techniques and also with the existing previous work. Results suggests that the proposed suggested approach shows way to a enhanced RFT in comparison to no order, reverse order, random order, previous work, using APFD metric. It is also shown that the number of TCs required to be executed to find the entire FA is same as a case of random order but lesser than other prioritization techniques. The results prove the efficiency and efficacy of our proposed suggested prioritization method, is better than the others including the existing one. In future, TC prioritization may be done by applying ANOVA for finding more accurate effectiveness.

REFERENCES

[1] G. J. Myers, The Art of Software Testing, John Wiley & Sons, 1979.

[2] H. Do, S. Mirarab, L. Tahvildari, and G. Rothermel, "The effects of time constraints on test case prioritization: A series of controlled experiments," IEEE Transactions on Software Engineering, vol. 36, no. 5, pp. 593-617, Oct. 2010.

[3] N. Chauhan, Software Testing: Principles and Practices, Oxford University Press, 2010.

[4] G. Rothermel, R. H. Untch, C. Chu, and M. J. Harrold, "Test case prioritization: an empirical study," Proc. Int'l Conf. Software Maintenance, pp. 179-188, Aug. 1999.

[5] W. E. Wong, J. R. Horgan, S. London, and H. Agarwal, "A study of effective regression testing in practice," Proc. Eighth Int'l Symp. Software Reliability Eng., pp. 230-238, Nov. 1997.

[6] G. Rothermel, R. H. Untch, C. Chu, and M. J. Harrold, "Prioritizing test cases for regression testing," IEEE Trans. Software Eng., vol. 27, no. 10, pp. 929-948, Oct. 2001.

[7] S. Elbaum, A. Malishevsky, and G. Rothermel, "Prioritizing tes cases for regression testing,"Proc. the 2000 ACM SIGSOFT Int'l Symposium on Software Testing and Analysis, Portland, USA, pp. 102-112, Aug. 2000.

[8] S. Elbaum, A. Malishevsky, and G. Rothermel, "Incorporating varying test costs and fault severities into test case prioritization," 23rd Int'l Conf. on Software Eng., Ontario, Canada, pp. 329-338, May 2001.

[9] A. Malishevsky, J. R. Ruthruff, G. Rothermel, and S. Elbaum, "Cost-cognizant test case prioritization," Technical Report TR-UNL-CSE-2006-004, University of Nebraska-Lincoln, Mar. 2006.

[10] T. Muthusamy and K. Seetharaman, "Efficiency of test case prioritization technique based on practical priority factor,"Int'l J. Soft Computing, vol. 10, no. 2, pp. 183-188, 2015.

[11] R. Kavitha and N. Sureshkumar, "Test case prioritization for regression testing based on severity of fault," Int'l J. Computer Sc. and Eng. (IJCSE), vol. 02, no. 05, pp. 1462-1466, 2010.

[12] B. Jiang, Z. Zhang,W. K. Chan, and T. H. Tse, "Adaptive random TC prioritization," 24th IEEE Int'l Conf. on Automated Software Eng., Auckland, New Zealand,pp. 233-244, Nov. 2009.

[13] R. Krishnamoorthi and S. A. Mary, "Factor oriented requirement coverage based system test case prioritization of new and regrssion TCs," Information and Software Technology, vol. 51, no. 4, pp. 799-808, 2009.

[14] Z. Li, M. Harman, and R. M. Hierons, "Search algorithms for regression test case prioritization," IEEE Trans. Software Eng.,vol. 33, no. 4, pp. 225-237, Apr. 2007.

[15] D. J. Mala, M. Kamalapriya, R. Shobana, and V. Mohan, "A non-pheromone based intelligent swarm optimization technique in software test suite optimization," IEEE Conf. Intelligent Agent and Multi-Agent Systems, pp. 1-5, July 2009.

[16] Y. Singh, A. Kaur, and B. Suri, "Test case prioritization using ant colony optimization," ACM SIGSOFT Software Eng. Notes, vol. 35, no. 4, pp. 1-7, July 2010.

[17] S. Elbaum, A. G. Malishevsky, and G. Rothermel, "Test case prioritization: A family of emperical studies," IEEE Trans. Software Eng., vol. 28, no. 2, pp. 159-182, Feb. 2002.

978-1-4799-8579-1/16/$31.00 ©2016 IEEE