A Regression Test Based on the Test Case Prioritization Techniques by using the Nature of Bees

Manaswini B, Rama Mohan Reddy A

Abstract*: *The software testing is a crucial role in the software development lifecycle. Testing can be done by the user who have full knowledge on the software, who has no knowledge and who has knowledge little bit on the software and the software testing can be tested in various level with various tests like unit, integrate, user, regression, acceptance etc.. The major and most important test is the regression testing. To perform the regression testing it takes more time. The current paper was focused on the regression testing to reduce the execution time and to improve the faults detection rate by considering the parameter of change coverage. The proposed system uses the test case prioritization techniques by using the nature of the bees. After performing the experimental study on an open source applications, the proposed system shows the efficiency and effectiveness in terms of the runtime and detecting faults when compared with the traditional system.

Index Terms*: *Software Testing, Regression Test, Change Coverage, Honey bee, Runtime

I. INTRODUCTION

Software testing is a process of evaluating and validating the mistakes of various applications. It is used to identify the missing requirements, gaps, incompleteness etc. of the user requirements [1]. Software testing can be done manually or automatically. Testing can be done in three different ways as shown in the figure 1.By collecting the various requirements from the user the testing will be performed in various ways. Some tests will be for the functional test and some other tests will be for the non-functional tests. The testing which was done with the full knowledge of the software is called the white box testing, the testing done with the limited knowledge is called as a grey box testing and the testing with no knowledge about the software is called as a black box testing. Various tests based upon the requirements are shown in the Figure 2.

One of the methods that used to improve the effectiveness of the regression test was a test case prioritization. The major goal for the testing is to the fault detection rate [2]. Many researchers has been worked in the area of test case prioritization, [3][4][5][6][7] to efficiently use the resource by identifying more faults. But not accomplished full-fledged.

Regression testing is a process/methodology to effectively and efficiently validates the errors after doing some changes to the existing software [9][10[11]. It is also called as validation test that which leads to assure the software quality. Generally regression test has many test cases which required more time to test all the test cases. So there is a necessity for optimizing those test cases [15].

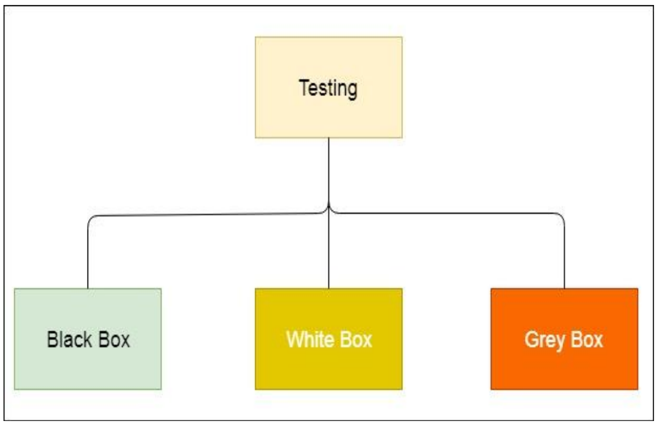

Figure 1: Different types of testing done by the knowledge acquired by the user.

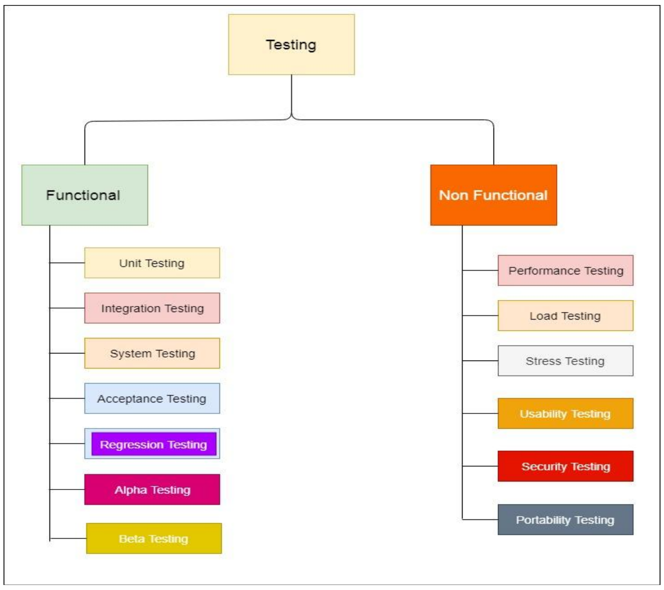

Figure 2: Various types of tests done based on the user requirements

Revised Manuscript Received on July 22, 2019.

Ms. B. Manaswini, Research Scholar, Department of CSE, Sri Venkateswara University, Tirupati, Andhra Pradesh

Dr. A. Rama Mohan Reddy, Professor, Dept of Computer Science and Engineering, SV University College of Engineering, Tirupati, India.

.

To optimize the various test cases the researches has proposed various techniques/methods based on the ratio of fault detection which will not cover more defects. Generally regression test can be done in different ways like test case prioritization, test case selection, reset all, etc.. But the proposed technique was done test case prioritization based on the maximum coverage of the functions of the user.

In regression testing the test case can be prioritize based on the method, branch, statement coverage which can be done statistically or dynamically [16][17][24]. The various Regression techniques are shown in figure 3 [25]. Software testing helps to the users to complete the software depending upon the requirements with error free[33].The regression testing cost is depend upon the product size, executed test cases and the resources to perform the test[34].the test case prioritization depending upon the entities like type of an application and requirements of the user. Based upon the execution time user can know the time for executing the each test case[35]. Many applications the actual solutions will be optimized to achieve and get the optimal solution [36]. Regression testing is a cost based testing and it is a continuous process of identifying the failure and gives the fast feedback for the developers [37]. The prioritization technique will provide the ranks for all the test cases in a test suite for the purpose of minimizing the test cases with high detection rate of false [38].

II. RELATED WORK

In paper [1], authors proposed a clustering based approach to perform the regression testing based on the test case prioritization, by considering the code scope as a metric and results prove that the productivity percentage was increased by 80%.

In paper [2] authors introduced evaluated by using the open source applications and results have been drawn to perform effective regression testing based on the test case prioritization.

In paper [8] author proposed a novel approach which optimize the testing process after doing the modifications for the existing software. This process has some automatic nature of identifying and ordering the test cases for detection of the false with time bounds. Results had shown some effectiveness in the process of regression test.

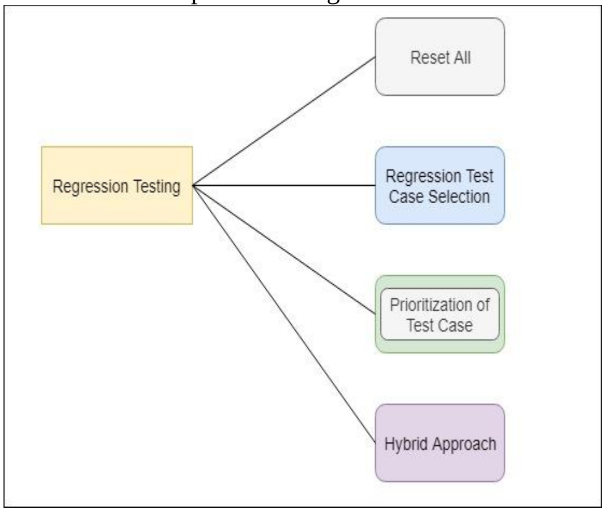

Figure 3: different techniques to perform the Regression Test

In paper [12] authors has proposed an algorithm to improve the test case prioritization for the regression testing to achieve objective of the text called hyper volume based genetic algorithm. Performed the empirical research and evaluated the effectiveness of cost to achieve the efficiency of the proposed algorithm.

In paper[13] author proposed various techniques for prioritizing the test cases on the various open source web application like backbone, lodash, bootstrap that collected from the website bower.io .The various techniques are like random ordering, exact matching, similarity matching, failure matching and combination of above techniques.

n Paper [15] author has improved their previous work by adding the real time practical entities, and concentrated more on the business side to identify the various defects.

In paper[16] author proposed a method and performed on empirical study by considering three independent variables prioritization techniques, test granularity, coverage criteria, prioritization and done the experiment study on 15 java projects that collected from Git Hub.

In paper [34] author proposed an empirical approach for evaluation of the software applications related to the databases. In paper [35] proposed an approach to measure the execution time for various manual test cases. Here two algorithms were designed for estimation and predicting the test cases.

In paper[37] proposed an Bloom filtering algorithm to filter the least period test cases during the selection phase of the testing and experimentally the proposed approach was demonstrated using the dataset of the goggle.

In paper [38] author has reviewed the ant colony based test case prioritization methods.

In paper [39] proposed an approach to identify the failures in the software based on the history this consist of two phases. In one phase weights will be assigned to each test case and second phase reordering of the test cases will be done based upon the failures of test cases.

In this paper [40] proposed and implemented a method by using the correlation between the changes of the court and failures of the test case. The result has proved that test cases will be more prioritized by taking the measurements like precision, recall and integration.

Here [41] proposed a prioritizing the test case method based on the co-relations of the requirements. This is particularly focused on the faults that occurred after performing the regression test. Finally, concluded that by prioritizing the test cases in the test suite the regression test will be optimized.

In this paper [42] proposed an approach to perform the regression test based on the reduction and prioritization techniques by considering the parameter path aware notation. The experimental results proved the efficiency of the proposed technique by comparing with the previous work.

In this paper [43] proposed an incremental based approach for the execution of the software. Here the execution will be done currently on the old and new software. The old is the existing software and new is the modified existing software with the concepts of inter-threads and inter-functions/inter-procedures the execution will perform parallel to identify the faults. The experimental results was generated by using the huge number of multi-thread c-programs, after comparing the results with existing and proposed solution by using the KLEE tools the proposed shows the effectiveness in terms of the code coverage.

III. PROPOSED SYSTEM

Many researches has proposed various regression testing methods to optimize the test cases by using ant colony optimization [27][28][29]and genetic algorithm[27][30][31][32] .Here algorithm was proposed for regression test case prioritization by using the nature of the bees. This was particularly focused on the metric, change coverage among the various versions, in the open source application jtopas which was obtained from the repository (SIR) [33]. The jtopas application consist of 5.4k lines of code in the form of a java library the various classes, test classes, test method was tabulated in the table 1[33].

Table 1: Details about the open source application Jtopas

Applications	Classes	Test Classes	Test Methods	Faul ts
Jtopas-v1	19	10	126	9
Jtopas-v2	21	11	128	9
Jtopas-v3	50	18	209	17

A. Nature of Bees:

Generally Bees will be three types, Employee Bees, Onlooker's Bees and Scouts Bees. Each employee bees has one source of food. It says that the count of an employee bees will be equal to the count of an food sources in the particular location/hive. The employee bees will check the food source and come back to the hive and will dance based on amount of the food they found. The onlookers' bees will notice the dance and choose any one of the food source based upon the way danced by the employee bees. Among the employee bees those who found the abundant food source will become the scout bees and search for the new source food.

B. Working of Proposed Algorithm:

First Initializes the all the existing test cases Ti={t1,t2,t3,t4,-------,tn} and then calculate the fitness value for all the initialized test cases FiTi={F_1T_1,F_2T_2,------F_nT_n} by calculating F_iT_i=w_1+W_2+W_3 , Where W_1,W_2,W_3 are functions defined by users after modifying the existing software. Compare all the fitness value and find the highest fitness value H [F_iT_i] and that test case will be given the higher priority and repeat the process till all the test case complete. By this process all the test cases will be arranged in the prior order for the execution. The architecture of the proposed system was shown in Figure 4.

Algorithm 1: /*Input =Test cases without prior order, Output=Test cases with prior order, Where n is a total number of test cases, FiTi = Maximum number of Requirements (Fitness value), H[FiTi] is Highest Fitness value among test cases.*/

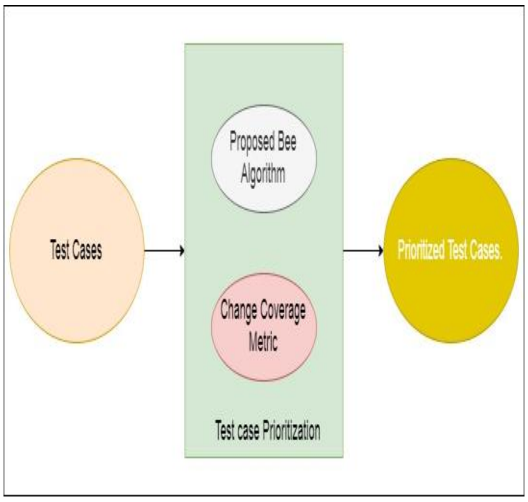

Figure 4: Architecture for the proposed system

```
Algorithm: Bees based Regression Testing Process
Input: T_1 [t_1, t_2, t_3,----------, t_n]
Ouput:T_H [t_2,t_3,t_4,-----------,t_n]
1: Start Bees based Regression testing process
2: repeat
3: for each test cases T_i
4: F_iT_i <-------- select test case T_i
5: H[F_iT_i] <-------- select test cast T_i
6: end for
7: until all the test cases finished.
8: T_H <--------- select test T_i
9: end Bee based regression testing process.
```

Table 2: Test cases with the change coverage's

S. No.	Test case	W1	W2	W4	W5	W6	W7	W8
		Functional requirements of User						
1	T_1	✔	✔	✔	✔	✔		
2	T_2		✔	✔	✔	✔		
3	T_3	✔	✔		✔			
4	T_4		✔	✔	✔		✔	
5	T_5	✔				✔		
6	T_6							✔
7	T_7	✔	✔	✔	✔			
8	T_8							
9	T_9	✔	✔	✔	✔			
10	T_{10}							

IV. RESULTS & DISCUSSION

After performing the testing on the open source database the results was shown in this section. The figure 5 shows the line graph. The graph represents the runtime of every test case for the proposed system with respect to the Test cases in terms of the minutes. It shows that each test case has different time to run the test case, because the coverage will be different for every test case. In the Figure 6 the graph represents the change coverage of the every test case. Here the metric was taken was the change coverage. The change coverage says that modifying the software. After modifying the software, the change converges with respect to the user requirements were shown in this plot. In the Figure 7 shows the fault detection rate of the proposed system and the existing system. It shows that the proposed system has high rate compared to the existing due to the metric, change coverage with the integration of the test case prioritization.

Table 3: Various test cases with the execution time

Test Case Number	Test Case	Execution Time
T_1	Login	0.12
T_2	Average Values	0.32
T_3	Min and Max Values	1.03
T_4	Array Size	1.20
T_5	Searching Array	0.67
T_6	Storing a Value	0.53
T_7	Getting the value	1.34
T_8	Actions	1.21
T_9	Responses	0.59
T_{10}	Filtering	1.21
T_{11}	Iterations	0.80
T_{12}	Initializations	0.23
T_{13}	Clock time	0.56
T_{14}	Run time	1.08

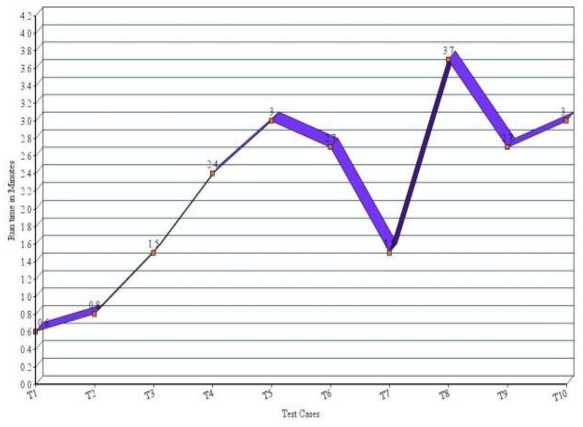

Figure 5: Test cases with respect to the Runtime of the test cases for the proposed system

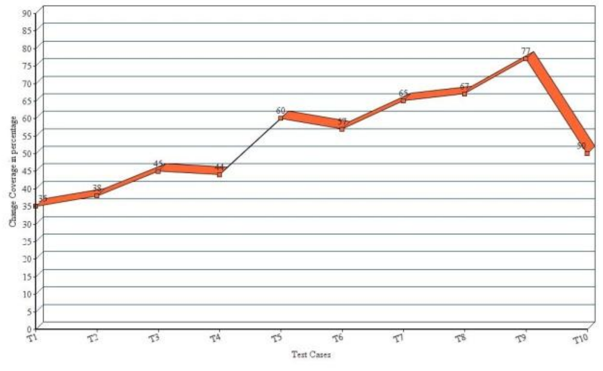

Figure 6: Test cases with respect to the change coverage

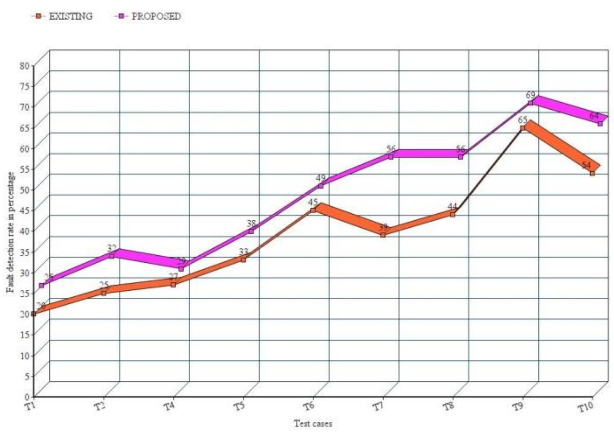

Figure 7: Fault detection rate for the proposed and exist systems.

V. CONCLUSION

This paper proposed an algorithm based on the nature of bees to perform the effective regression testing based on the text prioritization method this was implemented on the open source application jtopos, which is collected from the “subject infrastructure repository which has c and the java programming languages. Particularly the proposed algorithm shows the effective results in terms of the change coverage, runtime and fault detect rate when compared with traditional algorithm.

REFERENCES

1. Ramya, Paruchuri, Vemuri Sindhura, and P. Vidya Sagar. "Clustering Based Prioritization of Test Cases." 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT). IEEE, 2018.
2. Azizi, Maral, and Hyunsook Do. "Graphite: A greedy graph-based technique for regression test case prioritization." 2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW). IEEE, 2018.
3. de Souza, Jerffeson Teixeira, et al. "The human competitiveness of search based software engineering." 2nd International Symposium on Search Based Software Engineering. IEEE, 2010.
4. Harman, Mark. "Making the case for MORTO: Multi objective regression test optimization." 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops. IEEE, 2011.
5. Gu, Qing, Bao Tang, and DaoXu Chen. "Optimal regression testing based on selective coverage of test requirements." International Symposium on Parallel and Distributed Processing with Applications. IEEE, 2010.
6. Li, Zheng, et al. "A fine-grained parallel multi-objective test case prioritization on gpu." International Symposium on Search Based Software Engineering. Springer, Berlin, Heidelberg, 2013.
7. Yoo, Shin, and Mark Harman. "Pareto efficient multi-objective test case selection." Proceedings of the 2007 international symposium on Software testing and analysis. ACM, 2007.
8. Ulewicz, Sebastian, and Birgit Vogel-Heuser. "Industrially applicable system regression test prioritization in production automation." IEEE Transactions on Automation Science and Engineering 99 (2018): 1-13.
9. Luo, Qi, et al. "How do static and dynamic test case prioritization techniques perform on modern software systems? an extensive study on github projects." IEEE Transactions on Software Engineering (2018).
10. Lu, Yafeng, et al. "How does regression test prioritization perform in real-world software evolution?." 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE). IEEE, 2016.
11. Zhang, Lingming, et al. "Prioritizing JUnit test cases in absence of coverage information." 2009 IEEE

International Conference on Software Maintenance. IEEE, 2009.

12. Di Nucci, Dario, et al. "A Test Case Prioritization Genetic Algorithm guided by the Hypervolume Indicator." IEEE Transactions on Software Engineering (2018).
13. Kwon, Jung-Hyun, In-Young Ko, and Gregg Rothermel. "Prioritizing browser environments for web application test execution." 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE). IEEE, 2018.
14. Bower 2017 bower . https://bower.io/search/2017
15. Mahmood, Md Hasan, and Md Shazzad Hosain. "Improving test case prioritization based on practical priority factors." 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS). IEEE, 2017.
16. Zhou, Jianyi, and Dan Hao. "Impact of Static and Dynamic Coverage on Test-Case Prioritization: An Empirical Study." 2017 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW). IEEE, 2017.
17. Kim, Jung-Min, and Adam Porter. "A history-based test prioritization technique for regression testing in resource constrained environments." Proceedings of the 24th international conference on software engineering. ACM, 2002.
18. Jones, James A., and Mary Jean Harrold. "Test-suite reduction and prioritization for modified condition/decision coverage." IEEE Transactions on software Engineering 29.3 (2003): 195-209.
19. Walcott, Kristen R., et al. "Timeaware test suite prioritization." Proceedings of the 2006 international symposium on Software testing and analysis. ACM, 2006.
20. Zhang, Lingming, et al. "Bridging the gap between the total and additional test-case prioritization strategies." Proceedings of the 2013 International Conference on Software Engineering. IEEE Press, 2013.
21. Hao, Dan, et al. "A unified test case prioritization approach." ACM Transactions on Software Engineering and Methodology (TOSEM) 24.2 (2014): 10.
22. Hao, Dan, Lu Zhang, and Hong Mei. "Test-case prioritization: achievements and challenges." Frontiers of Computer Science10.5 (2016): 769-777.
23. Mei, Hong, et al. "A static approach to prioritizing junit test cases." IEEE Transactions on Software Engineering 38.6 (2012): 1258-1275.
24. Thomas, Stephen W., et al. "Static test case prioritization using topic models." Empirical Software Engineering 19.1 (2014): 182-212.
25. Ansari, Ahlam, et al. "TEST CASE PRIORITIZATION FOR OPTIMIZING A REGRESSIO
26. Ma, Zengkai, and Jianjun Zhao. "Test case prioritization based on analysis of program structure." 2008 15th Asia-Pacific Software Engineering Conference. IEEE, 2008.
27. Kulkarni, Nandakishore J., et al. "Test case optimization using artificial bee colony algorithm." International Conference on Advances in Computing and Communications. Springer, Berlin, Heidelberg, 2011..
28. Wagner, Israel A., Michael Lindenbaum, and Alfred M. Bruckstein. "Ants: agents on networks, trees, and subgraphs." Future Generation Computer Systems 16.8 (2000): 915-926.
29. Alaya, Ines, Christine Solnon, and Khaled Ghedira. "Ant colony optimization for multi-objective optimization problems." 19th IEEE international conference on tools with artificial intelligence (ICTAI 2007). Vol. 1. IEEE, 2007.
30. Michael, Christoph C., et al. "Genetic algorithms for dynamic test data generation." Proceedings 12th IEEE International Conference Automated Software Engineering. IEEE, 1997.
31. Bern, D., et al. "Breeding software test cases with genetic algorithm." Proc. 36th Hawaii Int. Conf. on System Sciences. 2003.
32. Michael, Christoph C., Gary McGraw, and Michael A. Schatz. "Generating software test data by evolution." IEEE transactions on software engineering 27.12 (2001): 1085-1110.
33. Rosero, Raúl H., Omar S. Gómez, and Glen Rodríguez. "Regression Testing of Database Applications Under an Incremental Software Development Setting." *IEEE Access* 5 (2017): 18419-18428.
34. Indumathi, C. P., and S. Madhumathi. "Cost aware test suite reduction algorithm for regression testing." *2017 International Conference on Trends in Electronics and Informatics (ICEI)*. IEEE, 2017.
35. Tahvili, Sahar, et al. "Towards Execution Time Prediction for Manual Test Cases from Test Specification." *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 2017.
36. Bian, Yi, et al. "Epistasis based aco for regression test case prioritization." *IEEE Transactions on Emerging Topics in Computational Intelligence* 1.3 (2017): 213-223.
37. Kwon, Jung-Hyun, and In-Young Ko. "Cost-Effective Regression Testing Using Bloom Filters in Continuous Integration Development Environments." *2017 24th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2017.
38. Sharma, Sonia, and Ajmer Singh. "Model-based test case prioritization using ACO: A review." *2016 Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC)*. IEEE, 2016.
39. Cho, Younghwan, Jeongho Kim, and Eunseok Lee. "History-based test case prioritization for failure information." *2016 23rd Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2016.
40. Knauss, Eric, et al. "Supporting continuous integration by code-churn based test selection." *Proceedings of the Second International Workshop on Rapid Continuous Software Engineering*. IEEE Press, 2015.
41. Ma, Tingting, Hongwei Zeng, and Xiaolin Wang. "Test case prioritization based on requirement correlations." *2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*. IEEE, 2016.
42. Sun, Chang-ai, et al. "dμreg: A path-aware mutation analysis guided approach to regression testing." *Proceedings of the 12th International Workshop on Automation of Software Testing*. IEEE Press, 2017.
43. Guo, Shengjian, Markus Kusano, and Chao Wang. "Conc-iSE: Incremental symbolic execution of concurrent software." 2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2016.

AUTHORS PROFILE

Ms. B. Manaswini has completed her B.Tech degree at Sri Padmavathi Mahila University, Tirupati and M.Tech at Andhra University. She published 10+ Seminars, Conferences and Journals at National and International wide. She attend more than 20 Workshops and Training Programmes.

Dr. A. Rama Mohan Reddy He **r**eceived his B.Tech Degree from JNT University Anantapur in 1986, Masters in Computer Science and Engineering from NIT, Warangal in 1991 and Ph.D. in Computer Science and Engineering from Sri Venkateswara University, Tirupati in 2007. He is currently working as a Professor of Computer Science and Engineering, SV University College of Engineering, Tirupati, India. His research interests are Software Engineering, Software Architecture, Cloud Computing, Operating System and Data Mining. He is life member of ISTE, IETE, ISC and CSI. He has more than 30 years of experience in teaching, 7 scholars completed Phd.'s under his guidance. He has 100+ publications and presented 78+ papers in National and International conferences.