*International Journal of Engineering & Technology*, 7 (4) (2018) 5266-5271

International Journal of Engineering & Technology

*Website: www.sciencepubco.com/index.php/IJET*
*doi: 10.14419/ijet.v7i4.28037*
***Research paper***

# Automated slicing scheme for test case prioritization in regression testing

**Manju Kaushal [1] *, Satheesh Abimannan [2]**

*[1] P.G Student, VIT, Vellore, Tamil Nadu, India*
*[2] School of Computer Science and Engineering, VIT, Vellore, Tamil Nadu, India*
**Corresponding author E-mail: satheesh.abimannan@vit.ac.in*

**Abstract**

Motivation: The testing approach which ensures that the software does not have any adverse affects due to the changes made in the existing features or addition of some new features is called regression testing. For testing the changes made in the previous versions of software, this type of testing is performed. To ensure that the numbers of test cases available in the software is not too large, it is important to select the regression tests and to do so, several techniques have been designed. To detect the individual functions from the software, the existing work applied the slicing technique. The parameters which are used in this approach are calculated manually to analyze importance of individual functions. The number of times the function is encountered and the number of functions relevant to the specific function are the two different parameters calculated here. A list of changes in the source code and the execution traces generated from the test cases which are run on previous versions are used to combine the modification, minimization and prioritization-based selection which thus generates a hybrid technique.
Problem Statement: In the existing system, the manual slicing technique is applied to perform test case prioritization. In manual slicing, the total number of times a function is triggered and the total numbers of functions attached are calculated manually to generate final function importance. This approach is very time consuming and inaccurate.
Method: In this paper, we studied that to prioritize the test cases based on the changes, a type of regression testing is used which is test case prioritization. The test cases of the functions which have higher priority are executed first and so on. Based on the changes made, the test cases are prioritized. For identifying maximum number of faults from the modified software, manual slicing and automated slicing are applied in this work. The proposed method will be the enhancement of manual slicing technique. The automated slicing technique will automatically calculate the functional importance based on number of attached functions and number of times function triggered. The proposed method has low execution time and detects more number of defects from the software. The dataset of ten different projects is used to test the performances of proposed and existing algorithms in MATLAB. Each project has seven functions and four numbers of changes are defined for the regression testing.
Results: The simulation results achieved at the end show that in comparison to manual methods, the implementation of automated test case prioritization has provided improvement in the fault detection rate and reduction in the execution time.

***Keywords***: *Automated Slicing; Regression Testing; Test Case Prioritization*.

## 1. Introduction

The mechanism, through which the various technical and non-technical regions within software can be tested such that various kinds of problems can be avoided at the time of its execution, is known as software testing [1]. The software is assessed for determining its quality using this mechanism. Within software engineering, testing is considered to be important since almost half of the development efforts and efforts of the systems that need reliability are included in testing [2].
During the changes of program under test or the external scenario in which it is being executed, it is important to maintain the activities of software which is done through regression testing mechanism [3]. It can be ensured that any kinds of changes made in a program do not impact the overall performance of the software negatively by executing a regression test suite when any new changes are made in the software [4]. Any kinds of tests that cover the test requirements redundantly are discarded by the regression test suite reduction techniques which are implied with the objective of controlling the size and execution time of a test suite. The test cases are reordered on the basis of an established priority metric by the test suite prioritization for enhancing the effectiveness of testing [5]. For example, to ensure that the test requirements are covered at higher speed in comparison to the original ordering, the tests can be rearranged by the prioritizer.
The challenge of running a test suite within a constrained scenario, a test prioritization technique can be provided as an alternative [6]. The test cases which are more likely to ensure that it is possible for a modified program to operate correctly are run such that the cost of testing can be minimized within test suite selection techniques [7]. To make sure that maximum advantage is given to the tested even when the testing is paused at any time duration, the best ordering of test cases is recognized by test case prioritization for testing [8]. To maximize some objective function, the test cases are prioritized and scheduled through test case prioritization techniques. For instance, achieving the code coverage at the highest speed, exercising the subsystems as per the order of their reflection of historical failure propensity or use the expected frequency of use to exercise features are some of the criteria which

can be used by the software test engineers for scheduling the test cases [9]. The test case prioritization approach might not be cost effective in case when the time needed to execute all test cases within a test suite is short [10]. Thus, test cases can be scheduled in any order here. The significance of merits provided through test case prioritization techniques is increased when there is sufficient time provided for running all the test cases [11]. For detecting the faults in shorter time, the tests cases with highest priority are scheduled and run earlier through the test case prioritization techniques [12]. The techniques through which the test cases are prioritized on the basis of costs are known as the cost-based techniques. Several research studies have been presented related to these techniques. Different kinds of costs for each kind of testing are evaluated [13]. The test cases include certain validation costs, the test selection includes the analysis cost and regression testing includes different costs which need to be studied in these techniques. Relative effectiveness can be calculated by comparing the test cases through these approaches [14]. Any regression test selection approaches in which all the test cases present within the existing test suite that identify the faults, are chosen by using these methods [15]. However, due to the discarding of tests, the costs of overlooking the faults are not included here. The test case prioritization methods that are based on the test execution history are called the chronographic history-based techniques. The statistical quality control and statistical forecasting are the two bases used to develop such approaches. A set of test cases is prioritized by applying the requirements-based test case prioritization techniques [16].

The current test case prioritization techniques were used as base to develop these techniques and the test cases were ranked here using certain factors. Further, to perform measurements, values are assigned for each factor by the authors [17]. The importance of testing a requirement earlier is measured using the weight prioritization [18]. The coverage based and fault-based techniques are combined by the maximize coverage for Early Fault Detection (MCEFD) technique. A metric which is used frequently as prioritization criterion is called structural coverage. The probability of quick maximization of fault detection is increased by quick maximization of structural coverage [19]. Maximizing the early coverage is the major objective of prioritization technique even though the test case prioritization aims to achieve a higher fault detection rate. The chance of revealing faults at high speed by maximizing the coverage in testing process is the major objective of this technique.

In this existing method, the function importance is calculated on the basis of two parameters which number is of times function encounter and number of functions attached with particular function [20]. The value of the functional importance depends on defined factors but function encounter and number of functions attached are calculated manually which increase execution time and also detect less number of faults from the software.

The automated slicing technique is proposed in this research work which is enhancement of manual slicing. In the proposed method, the function encounter and number of functions attached are calculated automatically which define functional importance. The proposed approach leads to reduce execution time for test case prioritization and also detect more number of defects.

In this paper, the test case prioritization is proposed for the software defect detection. In the section number 1, introduction is given about the regression testing and test case prioritization. The related work is described in the section 2. The proposed methodology is highlighted in the section 3. The implementation, case study and results are described in detail in section 4, 5, and 6 respectively.

## 2. Related work

Dipesh Pradhan, et.al (2018) proposed a novel technique called REMAP, which applied rule mining and multi-objective search for designing a black-box dynamic TP. The static priortizer, rule miner and dynamic executor and prioritizer are the three key components used in REMAP [21]. From the historical execution data, the relations are mined through rule miner. The test cases are prioritized statically by applying multi-objective search by the static prioritizer. The statically prioritized test cases are executed and the runtime test case execution results are used to update the test case order dynamically through dynamic executor and prioritizer. The evaluations are performed and results are achieved which show that an average of 18% higher Average Percentage of Faults Detected (APFD) was achieved by applying this new proposed technique.

Paruchuri Ramya, et.al (2018) studied that the updated product can be retested and any additional faults entering the existing software can be checked through regression testing [22]. The programming quality is checked and kept in better state through this approach. This study also highlighted that the faults and errors were identified easily by incorporating the requirements in testing phase. The effectiveness is not sufficient without using the source code information even though there are several prioritization techniques. The requirements information when used in test case prioritization, around 80% of increment is achieved in the productivity.

Yijie Ren, et.al (2018) proposed a two-layer model through which test case prioritization is assisted on the basis of GUI software features [23]. Here, the function call graph (FCG) is known to be the inner layer and the event handler tree (EHT) is called the outer layer. On the basis of two-layer model for prioritization, higher level of source code information is utilized here in comparison to the traditional techniques. The importance of modified functions for particular TCP version is highlighted using the centrality measure which is a complex network viewpoint. The proposed model is evaluated and it is seen through the results that the effectiveness of this proposed approach is better.

TomášPospíšil, et.al (2018) proposed a similarity function which was used for TCP techniques. The test cases that are created by different approaches can work with this designed function in universal manner [24]. The Hint-based Adaptive Random prioritization technique is used in this proposed technique. A similarity good effect of hint guidance is achieved through the comparative analysis of results. It shows that HARP can apply the function successfully. The accuracy of proposed approach is also known to be better in comparison to the traditional HARP technique.

Qi Luo, et.al (2018) presented a comparative study in which the TCP techniques are applied to mutation faults and real-world faults. The eight common TCP approaches, 357 real-world faults and around 35k mutation faults were studied here by including the Defects4J dataset collected [25]. As per the attributes of subject programs, the performances on real faults are not correlated with the TCP techniques on mutants as shown in the results. However, when the technique that provides best results on a set of mutants is applied of real faults, the results achieved might not be efficient enough. Thus, the mutation operators generated for particular program domains are generated here.

Maral Azizi, et.al (2018) proposed a novel graph-based prioritization technique through which the effectiveness is improved and two objectives are achieved collectively [26]. Four different open-source applications are utilized along with three widely used techniques to perform evaluations. Thus, the effectiveness and efficiency of prioritization were seen to be improved as per the achieved results. During the presence of limited time budget, the performance of proposed approach was known to be the best.

## 3. Regression testing with automated test case prioritization

The testing approach which checks that the software does not face any adverse affects due to some changes or additions made in an existing version is called regression testing. To prioritize the test cases based on the modifications made in developed software, the test case prioritization technique of regression testing is applied. Both, automated and manual test case prioritization techniques are applied in this work. To detect the faults from project, only manu-

al test case prioritization was applied in the existing technique. Number of functions related to a specific function and number of times function is encountered are the two parameters used in manual test case prioritization. The calculation of importance of each function is done based on these two parameters which used the function traverse value (FTV) value to perform calculation. Based on the modifications defined in the designed software, the calculation of FTV value is done. This work implements the automated test case prioritization for increasing the fault detection rate. Based on the total number of times a function is encountered and the total functions relevant to a specific function, the population values are given as input in the initial step. The population values are traversed and error is calculated at each iteration in the second step of the algorithm. The best mutation value of a function is calculated from the iteration at which the error is the highest for the mutation value. The function importance at which the test cases are prioritized based on the defined changes is called the function mutation value. Based on the defined changes, the function importance values are accessed in the final step. The total percentage of faults identified in a project after any change is given by calculating the best fitness value.

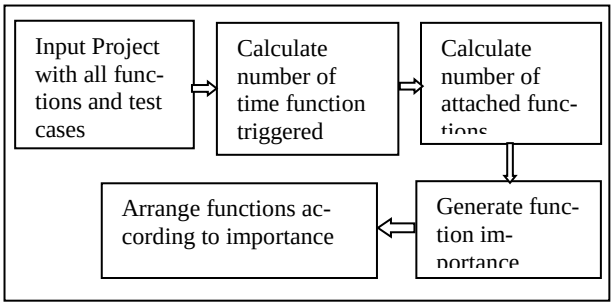


**Fig. 1:** Proposed Architecture.

As shown in figure 1, the architecture of the proposed model is shown. In the proposed architecture the dataset is taken as input. The functional importance is calculated with the equation number 1. The test cases are arranged according to functional importance means functions which have high importance is prioritized first and then so on

$$\text{Function importance}=\frac{\text{Number of time function encountered}}{\text{Number of attached functions}} \quad (1)$$

The proposed automated test case prioritization algorithm follows certain steps which are mentioned below:

Step 1: Depending upon the number of relevant functions, the important for function is calculated within the enhanced multi-objective algorithm. The most important function is considered to be the one which has the maximum association.

Step 2: The automated slicing technique is applied for calculating the number of functions relevant which results in traversing the DFD and creating the final results.

Step 3: An iterative approach is followed for automated slicing and to detect the maximum number of errors from the project, the best value of test case is searched.

Algorithm 1: Automated Test Case Prioritization

Input: Set of Test case ={P1, P2,…..,Pn},Set of clicks of each function={F1,F2,…..Fn}
Output: Prioritized Test Cases
Begin
1. I←Consider value of F(i) for the each test case
2. Test case F(i) value ← i
3. while ( fault value of each test case is calculated )
4. a=F(i)
5. calculate number of links L(i)=F(i)'/F(i)
6. if(L(i)>L(i+1) then
b=L(i)
else
b=L(i+1)
7. end
8. Calculate fault value Fault (i+1) =fault(i)/L(i)
if Fault(i) > Fault(i+1)
best_so_far<-Fault(i) then
i<- generate an individual randomly
End

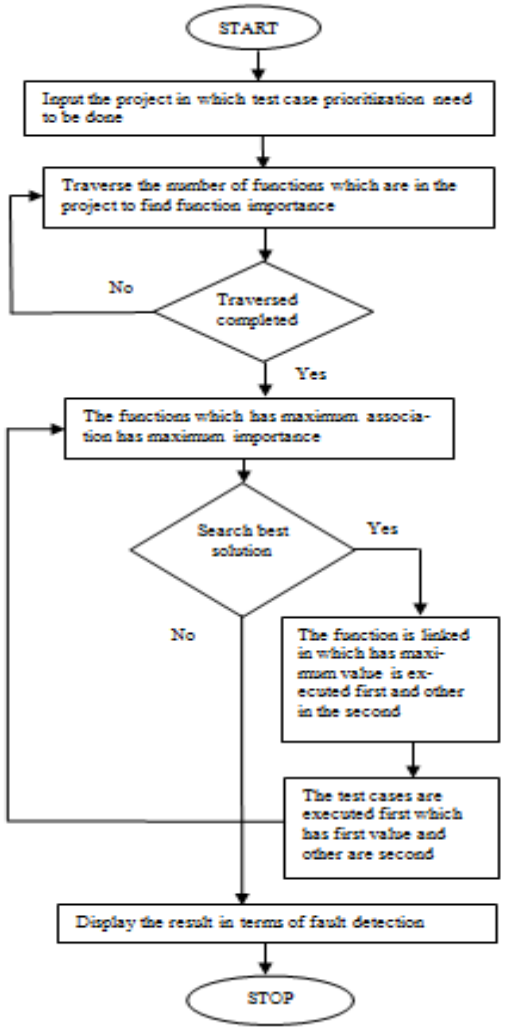


**Fig. 2:** Proposed Methodology.

## 4. Implementation

In this paper, MATLAB tool is used for the implementation of proposed methodology. In the MATLAB c is used as programming language. The guide tool box is used to drive interfaces for the execution. The experiments are performed on the online shopping website. The online shopping website is considered which have seven functions. The function names are show products, show category, check availability, request order, shipping, payment accept and cancel order. Every function of online shopping has its function execution value, number of attached function value. The function importance is calculated on the basis of defined two factors. The test cases of online shopping website are considered which will be prioritized.

## 5. Result and discussion

The results of the proposed model are tested in terms of accuracy, fault detection and execution time. The Table 1, describe the online shopping project details. In the project, the four changes are considered correspond to test case prioritization. The seven functions are considered and of every function execution value, attached functions, function importance. In the last column the fault detected with the automated approach is described.

**Table 1:** Fault Detected by Automated Slicing

| Functions | FE | AF | FI | FV | FDA |
|---|---|---|---|---|---|
| Show Prod-ucts | 3 | 6 | 0.5 | Acc to Change 1: 3.309524 | Acc to Change 1: 5.913 |
| Show Category | 8 | 7 | 1.1429 | Acc to Change 2: 3.166667 | Acc to Change 2: 5.4046 |
| Check Availability | 1 | 6 | 0.16667 | Acc to Change 3: 3.292857 | Acc to Change 3: 6.0006 |
| Request Order | 6 | 4 | 1.5 | Acc to Change 4: 8.459524 | Acc to Change 4: 17.8968 |
| Shipping | 9 | 3 | 3 | | |
| Payment Accept | 2 | 5 | 0.4 | | |
| Cancel Order | 7 | 4 | 1.75 | | |

FE – Function Execution Value; AF – Attached Functions;
FI – Function Importance; FV – Fitness Value;
FDA – Fault Detected by Automated slicing approach

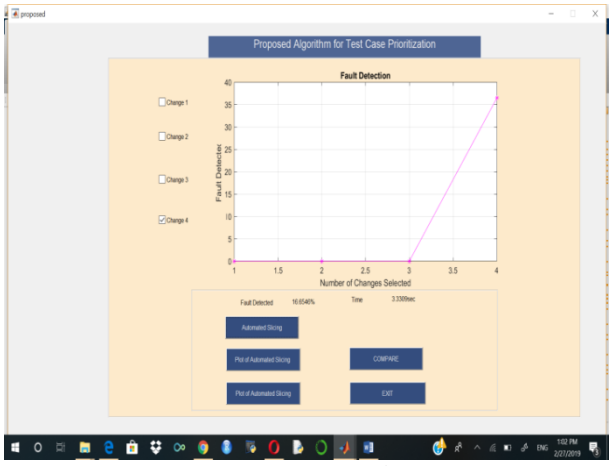


**Fig. 3:** Designed Interface.

In this figure 3, to implement the proposed technique interface is designed in MATLAB with four types of changes. In the interface, the fault detection rate is shown correspond to particular change.

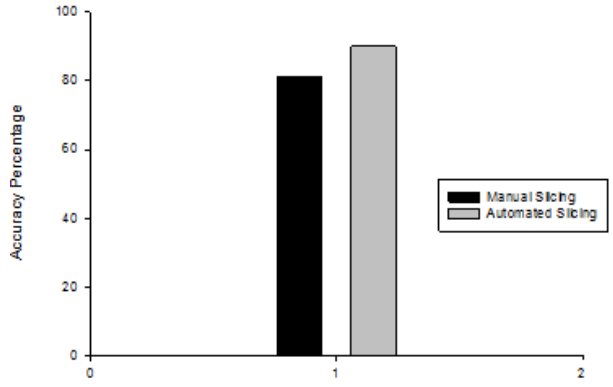


**Fig. 4:** Accuracy Comparison.

As shown in figure 4, the accuracy of the manual slicing and automated slicing is compared for the performance analysis. The accuracy of the manual slicing is less as compared to automated slicing due to do not use of optimization algorithm.

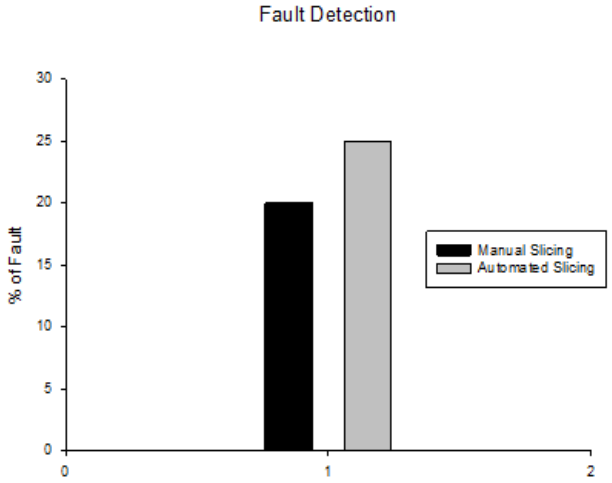


**Fig. 5:** Fault Detection Rate.

As shown in figure 5, the fault detection of the manual slicing and automated slicing is compared for the performance analysis. The fault detection of the manual slicing is less as compared to automated slicing due to do not use of optimization algorithm

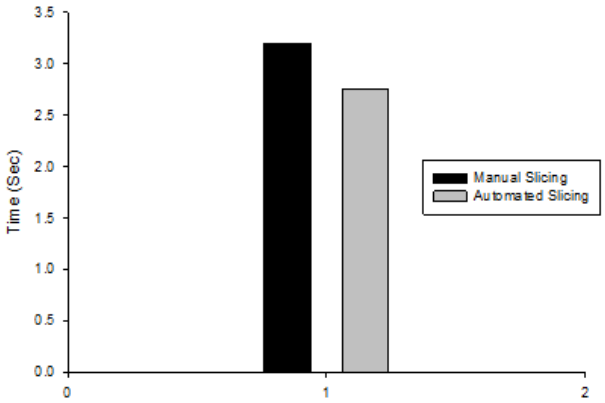


**Fig. 6:** Execution Time.

As shown in figure 6, the execution time of the manual slicing and automated slicing is compared for the performance analysis. The execution time of the manual slicing is high as compared to automate slicing due to do not use of optimization algorithm.

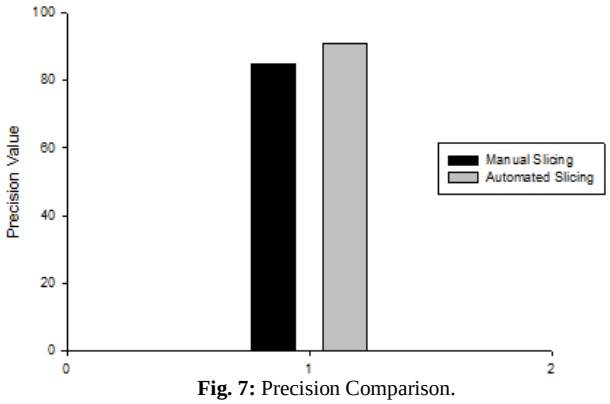


**Fig. 7:** Precision Comparison.

As shown in figure 7, comparative analysis of proposed and existing approaches is shown with respect of precision. The precision value for automated slicing is higher as per this analysis.

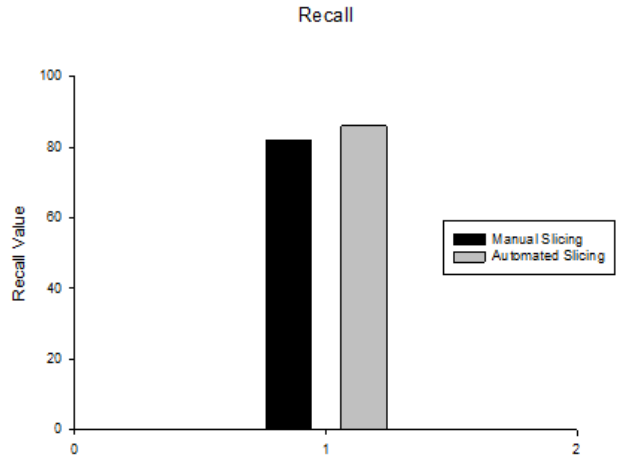


**Fig. 8:** Recall Comparison.

As shown in figure 8, the performance of both proposed and existing techniques are evaluated in terms of recall. Since the optimization algorithm is applied in automated slicing, the recall value achieved here is higher.

## 6. Conclusion and future work

The testing approach which ensures that the software does not have any adverse affects due to the changes made in the existing features or addition of some new features is called regression testing. For testing the changes made in the previous versions of software, this type of testing is performed. In the existing system, the manual slicing technique is applied to perform test case prioritization. In manual slicing, the total number of times a function is triggered and the total numbers of functions attached are calculated manually to generate final function importance. This approach is very time consuming and inaccurate. Thus, to perform test case prioritization automatically, the multi-objective algorithm is applied which includes three steps. Based on the total number of times a function is encountered and the total functions relevant to a specific function, the population values are given as input in the initial step. The population values are traversed and error is calculated at each iteration in the second step of the algorithm. The best mutation value of a function is calculated from the iteration at which the error is the highest for the mutation value. The function importance at which the test cases are prioritized based on the defined changes is called the function mutation value. Based on the defined changes, the function importance values are accessed in the final step. The total percentage of faults identified in a project after any change is given by calculating the best fitness value. Ten different projects that include four changes are used here to analyze the performances of proposed and existing algorithms in MATLAB. The simulation results achieved at the end show that in comparison to manual methods, the implementation of automated test case prioritization has provided improvement in the fault detection rate and reduction in the execution time. For automated test case prioritization, the greedy technique which is based on multi-objective algorithm is applied in the proposed algorithm. Several other greedy algorithms can be applied in regression testing for improving the performance of proposed algorithm. Comparative analysis of existing test case prioritization techniques and proposed algorithm can be done to test the reliability of proposed algorithm.

## References

[1] Zheng Li, Mark Harman, and Robert M. Hierons, "Search algorithm for Regression Test Case Prioritization," IEEE Transactions on Software Engineering, Vol. 33, No.4, April 2007. https://doi.org/10.1109/TSE.2007.38.

[2] Dennis Jeffrey and Neelam Gupta, "Improving Fault Detection Capability by Selectively Retaining Test Cases during Test Suite Reduction," IEEE Transactions on software Engineering, VOL. 33 NO.2, February 2007.

[3] Jennifer Black, Emanuel Melachrinoudis and David Kaeli, "Bi Criteria Models for All uses Test Suite-Reduction," 26th International Conference on Software Engineering (ICSE'04).

[4] Wes Masri, Andy Podgurski and David Leon, "An EmpricalStudey of Test Case Filtering Techniques Based on Exercising Information Flows," IEEE Transactions on software Engineering, VOL. 33, NO.7, February 2007.

[5] Scott McMaster, Atif M. Memon, "Call Stack Coverage for GUI Test Suite Rdduction," IEEE Transactions on software Engineering, VOL. 34 NO.1, January/February 2008.

[6] Maruan Khoury, "Cost Effective Regression Testing," October 5, 2006.

[7] Alexey G. Malishevsy, Gregg Rothermel, Sebastian Elbaum,"Modeling the Cost-Benefits Tradeoffs for Regression Testing Techniques," Proceedings of the International Conference on Software Maintenance ICSM'02), 2002 IEEE.

[8] Sebastian Elbaum, Alexey G. Malishevsky and Gregg Rothermel, "Test Case Prioritization: A Family of Emprical Studies," IEEE Transactions on software Engineering, VOL. 28, NO.2, February 2002.

[9] Gregg Rothermel, Roland H. Untch, Chentun Chu and Mary Jean Harrold, "Prioritizing Test Cases for Regression Testing," IEEE Transactions on software Engineering, VOL. 27 NO.10, October 2001.

[10] Hema Srikanth, Laurie Williams and Jason Osborne, "System Test Case Prioritization of New and Regression Test Cases", In Proceedings of the 4th International Symposium on Empirical Software Engineering (ISESE), pages 62–71. IEEE Computer Society, 2005. https://doi.org/10.1109/ISESE.2005.1541815.

[11] Hyunsook Do and Gregg Rothermel, "On the Use of Mutation Faults in Empirical Assessments of Test Case Prioritization Techniques", IEEE Transactions on Software Engineering, V. 32, No. 9, pages 733- 752, 2006. https://doi.org/10.1109/TSE.2006.92.

[12] K. Onoma, W.-T. Tsai, M. Poonawala, and H. Suganuma, "Regression testing in an industrial environment", Comm. Of the ACM, 41(5):81–86, 1988.

[13] K. R. Walcott, M. L. Soffa, G. M. Kapfhammer and R. S. Roos, "Time-Aware Test Suite Prioritization", In Proceedings of the International Symposium on Software testing and Analysis, pages 1-12, 2006

[14] Londesbrough, I., "Test Process for all Lifecycles", IEEE International Conference on Software Testing Verification and Validation Workshop, ICSTW'08, 2008. https://doi.org/10.1109/ICSTW.2008.4.

[15] Lu Luo, "Software Testing Techniques: Technology Maturation and Research Strategies", Carnegie Mellon University, USA, 1999.

[16] Sreedevi Sampath, Sara Sprenkle, Emily Gibson and Lori Pollock, "Web Application Testing with Customized Test Requirements – An Experimental Comparison Study", 17th International Symposium on Software Reliability Engineering (ISSRE'06), 2006.

[17] Xiaofang Zhang, Baowen Xu, ChanghaiNie and Liang Shi, "An Approach for Optimizing Test Suite Based on Testing Requirement Reduction", Journal of Software (in Chinese), 18(4): 821-831, 2007. https://doi.org/10.1360/jos180821.

[18] Jennifer Black, Emanuel Melachrinoudis and David Kaeli, "Bi-Criteria Models for All-Uses Test Suite Reduction", Proceedings of the 26th International Conference on Software Engineering (ICSE'04), 2004. https://doi.org/10.1109/ICSE.2004.1317433.

[19] Jung-Min Kim, Adam Porter and Gregg Rothermel, "An Empirical Study of Regression Test Application Frequency", ICSE2000, 2000.

[20] Jung-Min Kim and Adam Porter, "A History-Based Test Prioritization Technique for Regression Testing in Resource Constrained Environments", In Proceedings of the International Conference on Software Engineering (ICSE), pages 119–129. ACM Press, 2002.

[21] Dipesh Pradhan, Shuai Wang, Shaukat Ali, Tao Yue, Marius Liaaen, "REMAP: Using Rule Mining and Multi-Objective Search for Dynamic Test Case Prioritization", 2018 IEEE 11th International Conference on Software Testing, Verification and Validation.

[22] Paruchuri Ramya, VemuriSindhura, Dr. P. Vidya Sagar, "CLUSTERING BASED PRIORITIZATION OF TEST CASES", Proceedings of the 2nd International Conference on Inventive Communication and Computational Technologies (ICICCT 2018).

[23] Yijie Ren, Bei-Bei Yin, Bin Wang, "Test Case Prioritization for GUI Regression Testing based on Centrality Measures", 2018 42nd IEEE International Conference on Computer Software & Applications. https://doi.org/10.1109/COMPSAC.2018.10275.

[24] TomášPospíšil, JiíNovák, “New Similarity Function for Test Case Prioritization in Model-Based Context”, 2018 16th Biennial Baltic Electronics Conference (BEC).
[25] Qi Luo, Kevin Moran, Denys Poshyvanyk, Massimiliano Di Penta, “Assessing Test Case Prioritization on Real Faults and Mutants”, 2018 IEEE International Conference on Software Maintenance and Evolution. https://doi.org/10.1109/ICSME.2018.00033.
[26] Maral Azizi, Hyunsook Do, “Graphite: A Greedy Graph-Based Technique for Regression Test Case Prioritization”, 2018 IEEE International Symposium on Software Reliability Engineering Workshops. https://doi.org/10.1109/ISSREW.2018.00014.