Test Case Prioritization Using Regression Testing

[1]Supriya S. Lichade, [2]Parveen Thakur,
[1]Research Scholar [2]Assistant Proffesor,
[1]Computer Science and Engineering,School of Engineering and Emerging Sciences
[1]Baddi University of Emerging Sciences and Technologies, Baddi, India

***Abstract* - Regression testing is very important process in software testing but it is an expensive process. Regression means the re-execution of already existing thing. While performing Regression Testing there may be insufficient resources to perform re-execution of all test cases. Test case prioritization helps to improve the effectiveness of Regression Testing. However, running all the test cases in the test suite is prohibitive in most cases. Optimization of test case execution time to maximize the early fault detection rate of the original test suite will help to minimize the cost of regression testing. Because of resource and time constraint, it becomes necessary to develop some techniques which help to minimize existing test suites by removing redundant test cases and prioritizing them. Recently, clustering has been recognized as a primary data mining method for knowledge discovery in spatial database. The database can be clustered in many ways depending on the clustering algorithm employed, parameter settings used, and other factors. Multiple clustering can be combined so that the final partitioning of data provides better clustering.**

***Index Terms* - Regression Testing, Test case Prioritization, Clustering, Test Suites.**

I. INTRODUCTION

What is Regression Testing?
Regression means retesting of the unchanged parts of the application. Test cases are re-executed in order to check whether previous functionality of application is working fine and new changes have not introduced any new bugs. This test can be performed on a new build when there is significant change in original functionality or even a single bug fix. This is the method of verification. Verifying that the bugs are fixed and the newly added features have not created in problem in previous working version of software. Testers perform functional testing when new build is available for verification. The intend of this test is to verify the changes made in the existing functionality and newly added functionality. When this test is done tester should verify if the existing functionality is working as expected and new changes have not introduced any defect in functionality that was working before this change. Regression test should be the part of release cycle and must be considered in test estimation. Regression testing is usually performed after verification of changes or new functionality. But this is not the case always. For the release taking months to complete, regression tests must be incorporated in the daily test cycle. For weekly releases regression tests can be performed when functional testing is over for the changes. Testing is an important activity to ensure software quality. Big organizations can have several development teams with their products being tested by overloaded test teams. In such situations, test team managers must be able to properly plan their schedules and resources.
Regression testing concentrates on finding defects after a major code change has occurred. Specifically, it exposes software regressions or old bugs that have reappeared. It is an expensive testing process that has been estimated to account for almost half of the cost of software maintenance. Regression testing is an expensive, but important process in software testing. Unfortunately, there may be insufficient resources to allow for the re-execution of all test cases during regression testing. In this situation, test case prioritization techniques aim to improve the effectiveness of regression testing by ordering the test cases so that the most beneficial are executed first. Regression testing is primarily a maintenance activity that is performed frequently to ensure the validity of the modified software. In such cases, due to time and cost constraints, the entire test suite cannot be run. Thus, it becomes essential to select or prioritize the tests in order to cover maximum faults in minimum time.

Why Regression Test? How Much Regression Testing?
This depends on the scope of newly added feature. If the scope of the fix or feature is large then the application area getting affected is quite large and testing should be performed thoroughly including all the application test cases. But this can be effectively decided when tester gets input from developer about the scope, nature and amount of change.
Types of Regression tests: As these are repetitive tests, test cases can be automated so that set of test cases can be easily executed on new build. Regression test cases need to be selected very carefully so that in minimum set of test cases maximum functionality is covered. These set of test cases need continuous improvements for newly added functionality. It becomes very difficult when the application scope is very huge and there are continuous increments or patches to the system. In such cases selective tests needs to be executed in order to save testing cost and time. These selective test cases are picked based on the enhancements done to the system and parts where it can affect the most.

What We Do in Regression Test?
- Rerunning the previously conducted tests
- Comparing current results with previously executed test results

This is a continuous process performed at various stages throughout the software testing life cycle. A best practice is to conduct regression test after the sanity or smoke testing and at the end of functional testing for a short release.
To conduct effective testing, regression test plan should to be created. This plan should outline the regression testing strategy and exit criteria. Performance testing is also the part of this test to make sure system performance is not affected due to the changes made in the system components.
Regression testing is a type of software testing that verifies software that was previously developed and tested still performs correctly after it was changed or interfaced with other software. Changes may include software enhancements, patches configuration changes, etc. During regression testing new software bugs or regressions may be uncovered. Sometimes a software change impact analysis is performed to determine what areas could be affected by the proposed changes. These areas may include functional and non-functional areas of the system. One of the main reasons for regression testing is to determine whether a change in one part of the software affects other parts of the software. Common methods of regression testing include rerunning previously completed tests and checking whether program behavior has changed and whether previously fixed faults have re-emerged. Regression testing can be performed to test a system efficiently by systematically selecting the appropriate minimum set of tests needed to adequately cover a particular change. In Contrast with non-regression testing (usually validation-test for a new issue), which aims to verify whether, after introducing or updating a given software application, the change has the intended effect.

USES
Regression testing can be used not only for testing the correctness of a program, but often also for tracking the quality of its output. For instance, in the design of a compiler, regression testing could track the code size, and the time it takes to compile and execute the test suite cases.
"Also as a consequence of the introduction of new bugs, program maintenance requires far more system testing per statement written than any other programming. Theoretically, after each fix one must run the entire batch of test cases previously run against the system, to ensure that it has not been damaged in an obscure way. In practice, such regression testing must indeed approximate this theoretical idea, and it is very costly. Regression tests can be broadly categorized as functional tests or unit tests. Functional tests exercise the complete program with various inputs. Unit tests exercise individual functions, subroutines, or object methods. Both functional testing tools and unit testing tools tend to be third-party products that are not part of the compiler suite, and both tend to be automated. A functional test may be a scripted series of program inputs, possibly even involving an automated mechanism for controlling mouse movements and clicks. A unit test may be a set of separate functions within the code itself, or a driver layer that links to the code without altering the code being tested. Regression testing is the process of testing changes to computer programs to make sure that the older programming still works with the new changes. Regression testing is a normal part of the program development process and, in larger companies, is done by code testing specialists. Test department coders develop code test scenarios and exercises that will test new units of code after they have been written. These test cases form what becomes the test bucket. Before a new version of a software product is released, the old test cases are run against the new version to make sure that all the old capabilities still work. The reason they might not work is because changing or adding new code to a program can easily introduce errors into code that is not intended to be changed.

II. LITERATURE REVIVIEW

In this paper **R.Krishnamoorthi and S.A.Sahaaya Arul Mary[1]** proposed a new test case prioritization technique using Genetic Algorithm (GA).This experiment analyzes the genetic algorithm with regard to effectiveness and time overhead by utilizing structurally-based criterion to prioritize test cases. An Average Percentage of Faults Detected (APFD) metric is used to determine the effectiveness of the new test case orderings.Experimental analysis demonstrates that their approach can have This paper identifies and evaluates the challenges associated with time-aware prioritization.
Thillaikarasin Muthusamy and Dr. Seetharaman[2] proposed a algorithm which is based on analysis of the percentage of test cases performed to find the faults and on APFD metric's results. Outcomes demonstrate that their algorithms can also achieve better execution. For
instance, in the first project if only 75% test cases could be melt down due to resource constraint, random strategy could find more or less 66% faults; while our proposed algorithm detects about 88% faults. They had also validated their results with the aid of standard APFD metric.

Xiang Chen, Jian Xia and Pengfei[3] developed the algorithm to reduce the cost of regression testing, they can optimize test case execution schedule to maximize the early fault detection rate of the original test suite. Different from previous research, they use classification algorithms to guide the schedule process based on code change information and running result analysis.

Suresh Nageswaran[4] presents a new approach to the estimation of software testing efforts based on Use Case Points [UCP] as a fundamental project estimation measure. The acceptance test plan is then prepared with the use cases from the requirement documents as input.

In this paper, **Ryan Carlson, Hyunsook Do and Anne Denton[5]** consider a clustering approach to help improve test case prioritization. We implemented new prioritization techniques that incorporate a clustering approach and utilize code coverage, code complexity, and history data on real faults.

Due to the resource and time constraints, it becomes necessary to develop techniques to minimize existing test suites by eliminating redundant test cases and prioritizing them so
Manika Tyagi and Sona Malhotra[6] proposes a 3-phase approach to solve test case prioritization. In the first phase, we are removing redundant test cases by simple matrix operation. In the second phase, test cases are selected from the test suite such that selected test cases represent the minimal set which covers all faults and also at the minimum execution time. For this phase, we are using multi objective particle swarm optimization (MOPSO) which optimizes fault coverage and execution time. In the third phase, we allocate priority to test cases obtained from the second phase. Priority is obtained by calculating the ratio of fault coverage to the execution time of test cases.
In this research paper **Chintala Abhishek, vwginati Pavan Kumar, Harish Vitta and Praveen Rajan Srivastava[7]** focuses on finding a method which gives a measure of the effort to be spent on the testing phase. This paper provides effort estimates during pre-coding and post-coding phases using neural network to predict more accurately. The proposed effort estimation models for pre coding phase based on use case point and soft computing technique- neural network has been applied to improve upon the accuracy. The method that has been followed and the metric proposed have an advantage that it produces accurate results. For the post coding effort estimation the proposed model estimated the effort based on and used neural network to improve upon accuracy and the results have been found to show that the proposed estimation is in synchronization with the traditional effort estimation models.
K.Mumtaz and Dr. K. Duraiswamy[8] proposed a novel density based k-means clustering algorithm to overcome the drawbacks of DBSCAN and kmeans clustering algorithms. The result will be an improved version of kmeans clustering algorithm. This algorithm will perform better than DBSCAN while handling clusters of circularly distributed data points and slightly overlapped clusters. Dbkmeans clustering algorithm in different application areas such as medical image segmentation and medical data mining.

Eduardo Aranha and Paulo Borba[9] presents an estimation model for test execution complexity based on the size and execution complexity measured from test specifications written in a controlled natural language. Existing estimation models in the literature are based on system specifications and they estimate the effort required to perform more activities than test execution, such as defining and implementing tests.

Afnizanfaizal Abdullah, Safaai Deris and Mohd Saberi Mohamad[10] developed a new hybrid optimization method called Hybrid Evolutionary Firefly Algorithm (HEFA) is proposed. The method combines the standard Firefly Algorithm (FA) with the evolutionary operations of Differential Evolution (DE) method to improve the searching accuracy and information sharing among the fireflies. The HEFA method is used to estimate the parameters in a complex and nonlinear biological model to address its effectiveness in high dimensional and nonlinear problem. This is important to ensure that the speed performance will not be affected by the problem complexity. Secondly, the direction of the fireflies can be added to the method so that the firefly movements can be improved substantially. Lastly, the proposed method should be tested to estimate the parameters in more complex problems such as noise and identifiability.

Dr. Arvinder Kaur and Divya Bhatt[11] proposed the HPSO is a combination of Particle Swarm Optimization (PSO) technique and Genetic Algorithms (GA), to widen the search space for the solution. The Genetic Algorithm (GA) operators provides optimized way to perform prioritization in regression testing and on blending it with Particle Swarm Optimization (PSO) technique makes it effective and provides fast solution. In this paper they presented, the hybrid approach to solve prioritization in regression testing. The execution of algorithm has shown the effectiveness of the technique proposed. The automation of algorithm has provided solid base for its effectiveness.

Bharti Suri and Shweta Singhal[12] presents the analysis of the regression test prioritization technique to reorder test suites in time constraint environment along with the sample runs on various programs. Our analysis concluded that the ACO finds better orderings at higher values of the time constraint (TC). The correctness of the technique has also been recorded to be near optimal at an average.

Dianiel Di Nardo, Yvan Labiche[13] presents **t**his paper as an industrial case study of
coverage-based prioritisation techniques on a real world system with real regression faults. The study evaluates four common and different test case prioritisation techniques and examines the effects of using various coverage criteria on the fault detection rates of the prioritised test suites. The results show that prioritisation techniques that are based on additional coverage with finer grained coverage criteria perform significantly better in fault detection rates. The study also reveals that using modification information does not significantly enhance fault detection rates.

III. CONCLUSION

This paper reviews various works done in the field of test case prioritization. The paper analyses the various techniques that are used for test case prioritization. The researchers have developed various algorithm such as Genetic Algorithm, PSO, MOPSO, ACO, DBSCAN, etc. The above information describes these algorithm in short which helps to understand the techniques of test case prioritization. There are lot of work has been done in this area. In future the DBK-means algorithm will be introduce which is a combination of DBSCAN and K-means for prioritizing clusters of circular shape test case.

REFERENCES

[1] R.Krishnamoorthi and S.A.Sahaaya Arul Mary,” Regression Test Suite Prioritization using Genetic Algorithms” Department of Computer Science and Engineering, Bharathidasan Institute of Technology, Anna University, Trichy-24, India
[2] Thillaikarasi Muthusamy and Dr. Seetharaman.K,” Effectiveness of test case prioritization technique based on Regression Testing” Department of computer science,Annamalai University, Annamalai Nagar, Tamilnadu,India
[3] Xiang Chen, Zhaofei Tan, Jian Xia, Pengfei He“Optimizing Test Case Execution Schedule using Classifiers” School of Computer Science and Technology, Nantong University, Nantong 226019, China
[4] Suresh Nageswaran“Test Effort Estimation Using Use Case Points”Cognizant Technology Solutions, National Games Road,Yerwada, Pune – 411006.Maharashtra, India
[5] Ryan Carlson, Hyunsook Do, Anne Denton” A Clustering Approach to Improving Test Case
Prioritization: An Industrial Case Study”Department of Computer Science North Dakota State University Fargo, ND
[6] Manika Tyagi and Sona Malhotra“ Test Case Prioritization using Multi Objective Particle Swarm Optimizer” Department of CSE U.I.E.T., Kurukshetra University Kurukshetra, Haryana, India
[7] Chintala Abhishek, Veginati Pavan Kumar, Harish Vitta, Praveen Ranjan Srivastava
“Test Effort Estimation Using Neural Network” Department of Computer Science and Information System, Birla Institute of Technology and Science, Pilani, India.
[8] K. Mumtaz1 and Dr. K. Duraiswamy 2,“A Novel Density based improved k-means Clustering Algorithm – Dbkmeans”1 Vivekanandha Institute of Information and Management Studies, Tiruchengode, India 2 KS Rangasamy College of Technology, Tiruchengode, India
[9] Eduardo Aranha1 and Paulo Borba2“An Estimation Model for Test Execution Effort”
1Informatics Center Federal University of Pernambuco PO Box 7851, Recife, PE, Brazil 2Mobile Devices R&D Motorola Industrial Ltda Rod SP 340 - Km 128,7A - 13820 000 Jaguariuna/SP – Brazil
[10] Afnizanfaizal Abdullah, Safaai Deris, Mohd Saberi Mohamad,and Siti Zaiton Mohd Hashim “A New Hybrid Firefly Algorithm for Complex and Nonlinear Problem”
[11] Dr. Arvinder Kaur and Divya Bhatt “Hybrid Particle Swarm Optimization for Regression Testing” University School of Information Technology, GGSIPU, Delhi, India
[12] Bharti Suri1 and Shweta Singhal2 “Analyzing Test Case Selection & Prioritization using ACO” 1 Assistant Professor (Computer Science Department) University School of Information Technology Guru Gobind Singh Indraprastha University, Dwarka, Delhi-110075, India 2 Assistant Professor (Information Technology) Jagan Institute of Management Studies 3, Institutional Area, Sector-5, Rohini, Delhi-110085, India
[13] Daniel Di Nardo, Nadia Alshahwan, Lionel Briand 1 Yvan Labiche“Coverage-Based Test Case Prioritisation: An Industrial Case Study” 1 Interdisciplinary Centre for Security, Reliability and Trust University of Luxembourg, Luxembourg 2 Software Quality Engineering Laboratory Systems and Computer Engineering, Carleton University Ottawa, Ontario, Canada .