# Analytical Review on Test Cases Prioritization Techniques: An Empirical Study

Zainab Sultan
Department of Software Engineering
Bahria University Islamabad, Pakistan

Shahid Nazir Bhatti
Department of Software Engineering
Bahria University Islamabad, Pakistan

Rabiya Abbas
Department of Software Engineering
Bahria University Islamabad, Pakistan

S. Asim Ali Shah
Department of Electrical Engineering
Bahria University Islamabad, Pakistan

***Abstract*—For conclusively predicting the quality of any software system, software testing plays an important but a vital role. For finding faults early and to observe failures (anomalies) before implementation stage, software testing is done and if bugs (defects) are detected then software is passed through maintenance phase. The success and failure of a software project is often attributed to the development methodology used. It is also observed that in many scenarios, the software engineering methods are not implemented in their true spirit. Moreover, many of the development methodologies don't cater the change very well, because they follow a predefined development path which allows very less deviation. In software testing, regression testing is the important type of software testing. When any change made on the software then regression testing is done to check that it doesn't influence other parts of software. In regression testing, test cases are prioritized in order to reuse new test cases and existing test cases. Test case prioritization is done by using different techniques. This paper presents a review of different test case prioritization techniques.**

***Keywords*—Agile Software Engineering (ASE); Testing; Regression Testing; Test Suit Reduction; Test Case Generation; Test minimization; Test Case Prioritization Technique**

## I. Introduction

With quality defined as "meeting requirements, testing defines the quality as "fulfillment of the requirement specification", thus testing gives a good idea of the quality level. This leads to main objective of testing i.e. "Testing reduces the level of uncertainty about the quality of an software system. Software testing is the most significant step of software development life cycle. The testing involves the programs or an application's implementation having aim to discover software bugs and faults. There are different types of testing that software tester adopt according to their requirements such as Mutation Testing, Regression Testing, stress testing, security testing, load testing, black box testing, white Box Testing. According to testing type, the tester creates the number of test cases is called Test Suite. In testing process duration, tester finalizes test cases, implement on software according to developed test cases and then verify and check the results come by those executions. Regression testing is a testing technique which is applied on the altered application using pre-defined sets of Test cases.

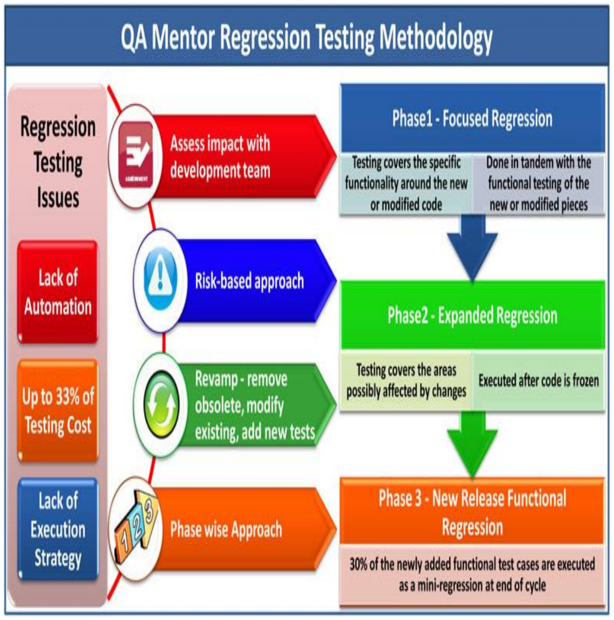
QA Mentor Regression Testing Methodology
Regression Testing Issues
Lack of Automation
Up to 33% of Testing Cost
Lack of Execution Strategy
Assess impact with development team
Risk-based approach
Revamp - remove obsolete, modify existing, add new tests
Phase wise Approach
Phase1 - Focused Regression
Testing covers the specific functionality around the new or modified code
Done in tandem with the functional testing of the new or modified pieces
Phase2 - Expanded Regression
Testing covers the areas possibly affected by changes
Executed after code is frozen
Phase 3 - New Release Functional Regression
30% of the newly added functional test cases are executed as a mini-regression at end of cycle

Fig. 1. Regression testing methodology

When an application is first time tested, test suite is constructed to enhance its functionality. Tester preserve test suite for further use. As changes are made in system, then these pre-defined test suites are applied by testers so that it can be ensured that no new bugs are introduced in the code that have been tested. If changes occur in system, then re executing every test for each module after change is really inapplicable and illogical. Moreover, it is much costly approach to execute all test cases once changes made. So to decrease the regression testing cost and to mold it in more profitable form, "test case prioritization" concept was introduced by the researchers. In test case prioritization, all test cases are arranged in an order to magnify some equitable behavior. To establish the priorities of test cases certain factors depending upon the requirement are analyzed and selected and then preference is allocated to test cases. Test case prioritization delivers a path to lineup and executes test cases, which has maximum priority in order to detect earlier faults. Different test case techniques of prioritization are reviewed in this review paper. For researchers, it can provide help to find which technique is best suitable for which scenario.

## II. Regression Testing Approaches

Here are some approaches for regression testing [1].

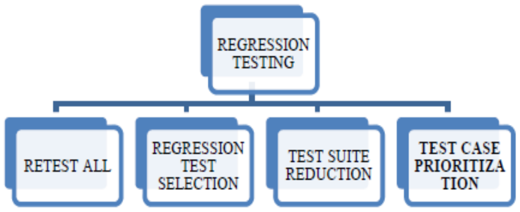

Fig. 2. Approaches of Regression Testing

### 1) Retest All

For regression testing, it is the most genuine Technique. During this approach, simply in test suite all test cases execute.

### 2) Regression Test Selection

It deals with the problem arising, from test suite for adopting a subset of test case. Then chosen test cases are executed to verify and test modifications occur in program.

### 3) Test Suite Reduction

This process containing two parts: First is, identification of relevant or superfluous test cases, second one is, exclude those test cases.

### 4) Test Case Prioritization

It includes with the realization of idealized sorting of test cases. That ordering must increase the fascinating properties like early detection of faults and no of fault detection and minimizes the cost factor.

## III. Test Case Life Cycle

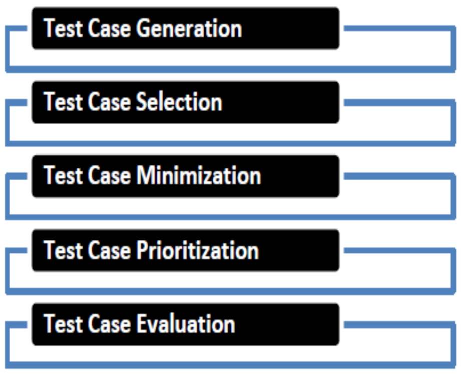

Fig. 3. TCLC test case life cycle

Figure 3 presents Life cycle of test case (TCLC). Testing of software is important, significant and profitable process in SDLC (software development life cycle). In testing process, on a set of different test cases tester executes program, and compare actual outcome with expected outcome. From different software artifacts like requirements specifications and designs test cases are mostly extracted. Testers study requirement document first to understand the requirements and specifications, when they understood requirements they start preparing test cases. Using tools they are automatically generated. Different techniques are used for test cases generation. Various phases of test case life cycle are, “Test case generation, test case selection, test case minimization, test case prioritization and evaluation”. Test case generation is the process of generating test suites for a particular system. Some methods of test case generation relays on application, like test case generation for object oriented application, web application, UML applications, applications based on evolutionary and genetic algorithms, structured based systems, and many others. Test cases are categorized into five classes as “reusable, retest able, obsolete, structural, new specification and new structural test cases”. In test case selection from a test suite choosing test cases in software testing process for reduction of time, Effort and cost. It is just like to test case minimization technique. The Test suite minimization approach, relay on metrics such as from a single version measurement of coverage of the program under test. Diversity among these two approaches builds upon the modifications occurred in SUT. According to the changes made among preceding and ongoing version of the SUT, the test cases are chosen. In Test case prioritization from multiple test suites of software test cases are conscripted, ranked and arranged. To rank and organize the test cases there are a lot of techniques. Some priority is accredited to each test case; however when multiple test cases are assigned the same priority or weights problem originates sometimes. In test case evaluation, to test the software, test cases are appraised to determine the suitable test cases. To evaluate the test cases, perform: (1) Prepare experiment data, (2) Run the test suites prioritization method, (3) Evaluate results.

## IV. Test Case Prioritization Vs Random Test Cases

In a research paper [3], researchers took a bank application project and compared prioritized and random test cases. They found, more faults can be identified if test cases will be prioritized. Results are displayed in figure 4.

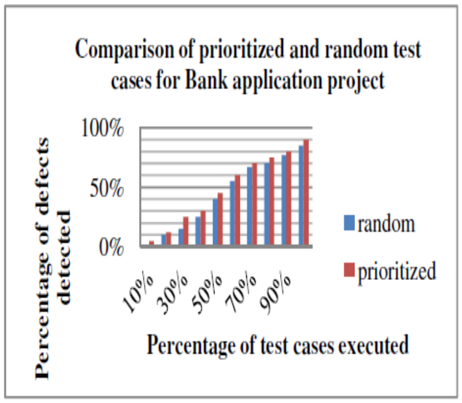

Fig. 4. Prioritized and Random comparison

Some attributes are taken and evaluated them and compared either proposed techniques by different authors are better or random ordering.

Attributes are: Comparison based on 1) size of test cases 2) Time taken by test cases 3) Effort taken by test cases 4) Cost taken by test cases 5) Efficiency 6) More defects found by test cases 7) Can be used in other projects.

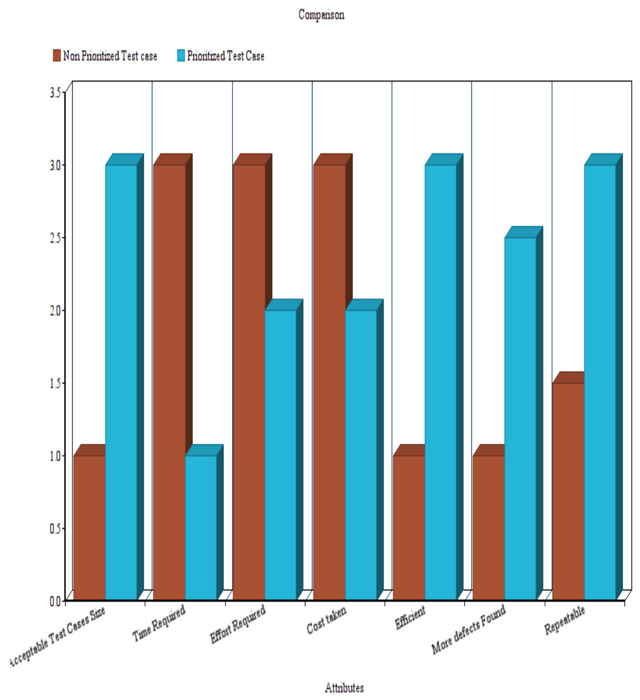

Fig. 5. Comparison b/w Random and prioritized test cases on different Attributes

Hence the test cases which are prioritized give much exceptional fault discovery than the test cases which are not prioritized. Further using techniques of test case prioritization, diminish the project's time and budget by prioritizing the most significant test cases.

## V. Factors for Classification of Test Case Prioritization

Prioritization of test cases relays on following certain factors. In this section, the assorted factors are explained, prioritization depends. [2]

1) *Customer Requirements*
2) *Coverage Based*
3) *Cost effective*
4) *History Based*

### 1) *Customer Requirements*

In "customer requirements based prioritization techniques", test cases are arranged having focus on customer's requirements which are documented all the while requirements specification collection phase. Key points to prioritize the test cases for this approach are "Assigned property of customer (CP), Complexity of requirement (RC), and volatility of requirement (RV)". Values are accredited to these factors and high value factor illustrates a need for prioritization of test cases.

### 2) *Coverage Based*

Prioritization technique which depends on coverage, during testing the test case prioritization are on validating and calculating a program's source code that has been executed. [2] Word "coverage" refers during the process of testing the form of code that has coated, it can be "coverage of requirement, coverage of total requirement, and coverage of additional requirement", hence test case has capacity in this approach to test the main code's parts and prioritize them.

### 3) *Cost effective*

In this approach, test cases are prioritized depending and analyzing cost factor. Cost conceivable as, "requirement gathering cost, regression testing cost, execution and validating cost of test cases, analyses cost and prioritization cost of test cases. Therefore, test cases demanding the diminish cost have maximal value.

### 4) *History Based*

In "History based", Test cases are scheduled depending upon history of test cases that refers priority relays on test case's prior achievements. The past performances of test cases boost or decline the possibility in order that it will be valuable in testing ongoing session.

## VI. Literature Review

This section mentions and summarizes some test case prioritization research papers.

Hema Srikanth and Laurie Williams in their paper [4] introduced a technique in which they target three aspects: on requirements Customer assigned priority (CP), complexity of requirements (RC), Volatility of Requirement (RV). Customers allocate the CP value. Developers assign RC value. They allocated values from 1 – 10 to every factor for analysis. They declared the factors having high values identify the demand of prioritization of test cases. Weight prioritization (WP) indicates significance of verifying requirement at early stage, and it is represented by below equation:

$$WP = \sum_{PF=1}^{n}(PF\ value\ *\ PF\ weight)$$

Here, n denotes absolute no of test cases, WP illustrates weight prioritization, and PF value is allocated to each factor i.e. CP, RC and RV. PF weight is assigned to every factor such as CP, RC and RV. Test cases are organized in corresponding way so that, having higher WP value is performed before others.

Siripong [5] described 4C classification in his research paper. He categorized techniques into 4 bases: Based on "Requirements of customer, Coverage, Cost, and Chronographic history". The Author also defined two methods for test case prioritization. In MTSSP Method: MTSSP was developed to figure out issues originated in multiple test suites allocated to the similar preference. In test suite, on the basis of defect factor test cases are assigned preference. If issue is still not resolved then the test suite is arranged on the basis of timeline aspect. Again if the problem is not fixed then the test suite is prioritized according to budget factor. If this experiment does not succeed then align them with complex factor. If the obstacle remains same then random method is used for test suite prioritization. MTSPM intention is to prioritization of multiple test suites effectively. Test suites are only prioritized by time and cost factors. If the issue does not resolved and test suites are having the similar priority then they are randomly organized. In test suites to prioritize the test cases MTSSP method is used. After MTSSP if the priority remains same, then MTSPM method is applied to prioritize them.

Srivastata [6] recommended test case prioritization according to criteria of increased APFD (Average percentage of fault detected) rate. Author invented a new approach in which test case determine the average no of faults found per minute. Finding that value arranges the test cases in descending order, APFD is determined by equation

$$APFD = 1 - \left(\frac{TF_1 + TF_2 + \ldots\ldots\ldots TF_m}{nm}\right) + \left(\frac{1}{2n}\right)$$

Here, under assessment T is test suites, m represents number of faults found in software or module of software under evaluation, n denotes absolute number of test cases. TFi represents number of test cases that are possessed in test suite T, that reveals faults i. If the previous knowledge of faults is known then APFD can be applied.

Korel et al. [7] implemented an experiment in order to validate and verify efficiency and effectiveness of both simple code based and model based test case prioritization. The Aim of this experiment was to verify these approaches to evaluate the performance of earlier fault detection in the system that has modified. So the result shown that as the execution of model based test cases are very quick as compared to code based test case prioritization so it can bring improvement in earlier detection of fault. That’s why for the whole test suites the model based test prioritization in comparison of code based test case prioritization is cheap.

Korel et al. [8] in their paper prioritized the test cases by applying certain model based test prioritization heuristics. It has some problems that selective model based prioritization focus only the number of identified transitions which does not have valuable impact on the improvement of earlier detection of faults. Value based regression test case prioritization [9] is used for the earlier fault detection. For prioritization of test cases, they proposed an algorithm on the basis of 6 factors. Modifications in requirement, Priority of customer, complexity in Implementation, traceability in requirement, time of execution and Impact of fault. The PSO (Particle swarm Optimization) is applied for allocating utility to factors and comparison of factor’s values. Maximum value is the highest number, and minimum value is the lowest number. For all the test cases, sum up the values of factors. If the factor values of two test cases match, then it shows by comparing requirement utilities of those test cases the judgement is made. This implies, in terms of time and cost PSO algorithm is much efficient, powerful and useful than greedy algorithm.

Kumar et al. [10] proposed that the prioritization is set on the basis of harshness of faults. The total severity of faults detection (TSFD) is addition of severity frequency of all defects that are exposed in a product is given in below equation. Here n shows total number of faults occur in product.

$$TSFD = \sum_{i=1}^{i=n} SM$$

R.Beena et al. [11] has given an effective and efficient way of choosing and organizing of test cases on the basis of coverage of code. This technique is much significant for decreasing time and budget for regression testing. This approach consists of 3 techniques. “Minimization, selection and prioritization of test cases”. TCS algorithm considers for selection of test cases and TCP algorithm considers for prioritization. Test cases are assembled in three categories, Outdated, Required and Surplus. TCCij is matrix that represents the test cases and the statements covered. SDELi is vector that represents the statements deleted in P. SMODi is vector that represents the statements modified in P. These all 3 are the input and output will be the modified matrix TCCij, cluster of test cases, out datedi, surplusi, requiredi. Any statement that cover any test case or many test cases, is considered as out dated. The statements that are modified and are not covered by any statement will be added into surplus cluster and will be removed from TCCij. Remaining test cases are added into required cluster. So the original TCCij will be greater than the required TCCij. .For prioritization, that test case selection output is considered as input of the prioritization algorithm TCPi and output will be TCPi which is vector and consist of test cases to get 100% code coverage. In this algorithm, the statements that are camouflaged by test cases, from new or needed TCCij they are summed up. Choose test cases having maximum value and include it into the TCPi Vector. Remove TCPi Vector from TCCij. Repeat all these steps till all the statements are deleted.

Parakash et al [12] invented a new method for test case prioritization known as “potentially weighted method”. This approach prioritizes test cases based on “potential coverage” like coverage of code, function, branch, fault, and path and for criteria weights are assigned. Test cases are given preference on the basis of value of weight, and the weight is from high to low. Every statement in code must be executed at least once; this is the basic purpose of code coverage testing. The criteria value Ticd is determined as

$$T_i^{cd} = \left(\frac{N_{cd}}{M_{cd}}\right) * 10$$

Line of codes that are coated by test case Ti, denoted by Ncd. Any test case Ti that covered Maximal number of codes represented by Mcd. The function coverage testing is very useful as code must be executed at least one time. Function coverage is defined as

$$T_i^{fn} = \left(\frac{N_{fn}}{M_{fn}}\right) * 10$$

Here no of functions are denoted by Nfn that are measured by test case Ti. Test case Ti covered maximum number of functions denoted by Mfn. To increase the capability and performance and for reduction of budget and time this approach is very useful.

Praveen [13] proposed the paper in which average faults per minute are determined; test cases are prioritized on the basis of fault detection rate. For test case prioritization, author invented a new algorithm. Average fault per minute is calculated in this algorithm.

$$AF/m = \frac{F}{Tcost}$$

In the algorithm input is T which is Test suite, identification of number of faults by test case f, and to run each test case cost required is Tcost and and output will come as prioritized test suite. After calculating the fault identified per minute, on the basis of each test case value arrange T in descending order. With the help of APFD analysis has done for prioritized and non-prioritized cases. Author proved with the help of graphs that in the experiment and analysis that test cases which are prioritized are more efficient and useful.

Kavitha et al [14] invented a technique on the basis of rate of fault impact and fault detection to prioritize the test cases. For identification of dangerous faults at earlier unique algorithm is invented. The invented algorithm determines the test case weight age.

$$Tcw = RFT + Fl$$

Tcw denotes test case weightage age.

$$T_{cw} = RFT_i + Fl_i$$

RFTi denotes fault detection rate, average no of faults per minute by test case, is knows as fault detection.

$$RTFi = \left[\left(numberof\ \frac{faults}{time}\right) * 10\right]$$

Here, Fli denotes fault impact.

$$FLi = \left(\frac{Si}{\max(s)}\right) * 10$$

Where, Si Denotes Test case value.

$$Si = \sum_{j=1}^{t} SV$$

Max(s) denotes high level of severity. The algorithm prioritizes the test cases on the basis on test case weightage. Results have proved, the proposed algorithm is efficient and useful.

Using genetic algorithm [15] invented testing that includes determination of the test cases, which can able to detect bugs in system. The process is tough and time taking. Author proposed a new technique in this paper, for test case prioritization according to their capabilities of discovering bugs. Higher priority is assigned to more similar errors. Low priority is assigned to less similar errors. Through genetic algorithm this order will be achieved. For finding the fittest chromosome like selection, crossover, mutation is applied on chromosomes. Genetic algorithm Steps includes: Set population, find fitness of population, for individual apply selection, apply crossover and mutation, figure out and recreate chromosome. Approximately an optimized solution for large number of time will be provided.

Amitabh et al [16] invented a prioritization technique on the basis of binary code. They delivered a system called ECHELON. On the basis of modifications that are being done in the program it prioritizes the test cases. ECHELON is a unified part of Microsoft development process. It uses simple and quick algorithm. It give results within few seconds by saving time and resources.

H.Do et al [17] present a controlled experiment. Software developers used Junit framework for generating test cases that are being executed in java. Junit provides helps to testers to create test cases and to re execute these test cases when modifications occur in program. There experiment is for finding the efficiency and effectiveness of test case prioritization under this JUnit Framework. They developed 6 blocks and method level techniques which are as follows. 1) Total block coverage 2) Additional block coverage 3) Total method coverage 4) Additional method coverage 5) Total DIFF method 6) Additional DIFF method.

Bryce et al [18] presents the prioritization of test cases for interaction coverage. Their focus was for event driven software. On the basis of five criteria's they prioritize the test cases. 1) Unique event coverage > Prioritize test cases like as soon as possible they measurer all different and uncommon events. 2) Event interaction coverage > Covers 2 way interaction and 3 way interaction. 3) Random test ordering > randomly ordering of test cases without any rule. 4) Shortest to longest with length of test cases. 5) Longest to shortest with length of test cases. The results concluded that for quick detection of faults, test suite must have dominant 2-way and 3-way interaction's percentage.

Do et al [19] invented a technique in which they wanted to figure out what are the impact on specific prioritization technique of variations in time constraint and also the impact on cost of regression testing. They presented four techniques, in which two belong to total and additional coverage and two are related to Bayesian network. The equation used in this technique is:

$$COST=PS * \sum_{i=2}^{n} (CS\ (i) + CO_i\ (i) + CO\ (i) + b(i) * CV_i(i) + C(i) * CF(i))$$

Additional techniques are considered to be better than total. Results have shown that time constraints perform a remarkable role in test case prioritization techniques.

Jieng et al [20] invented test case prioritization technique ART (Adaptive random). They presented nine new coverage based ART techniques. They categorized them into three groups "maxmin, maxavg, and maxmin". Their coverage is at statement level, branch level, and function level. "1) ART-st-maximum 2) ART-st-maxavg 3) ART-st-maxmax 4) ART-fn-maximum 5) ART-fn-avg 6) ART-fn-maxmax 7) ART-br-maxmin 8)ART-br-maxavg 9)ART-br-maxmax." A comparison was done between these techniques and randomly ordering and the results was these are 40-50% more efficacious than randomly ordering of test cases. ART-br-maxmax is perfect among all groups. In order of exposing defects and failures they are more effectivethan traditional coverage techniques.

Maia et al [21] invented a metaheuristic algorithm that is known as GRASP (greedy randomized adaptive search procedure). They have done automatic prioritization of test cases with the help of GRASP. A metaheuristic algorithm found best and optimistic solutions. They compared the GRASP technique with some search algorithms like simulated annealing, greedy, genetic. Their comparison was on the basis of performances and coverage. Their coverage criteria were block, decision and statement. The results has shown that additional greedy is best algorithm but GRASP is not worse than that among all these five algorithms. Among all these algorithms, GRASP surpassed the simulated annealing, genetic and greedy algorithm.

Dennis et al [22] invented prioritization technique using relevant slice. A program includes a lot of statements. Some statements have no impact on output generated by test cases but some statements have potential to effect the output generated the test. All these statements create a group and this group goes to relevant slice. In this technique following factors are considered. 1) No of statements of the output in relevant slice 2) No of statements of the output that are not in the relevant slice are implemented by test cases. Equation that is used for checking test case weight is,

$$TW = Reqslice + ReqExercise$$

Reqslice presents the no of requirements in the relevant slice of output. ReqExercise presents the no of requirements that are exercised by the test case.

Leung et al [23] invent a cost model that compares the certain regression techniques. They divide the cost into 2 groups. Direct cost and Indirect cost. Direct cost involves 1) System analysis cost Ca 2) Test selection Cost Cs 3) Test execution cost Ce 4) Result analysis cost CT. Indirect cost includes 1) Overhead cost 2) Tool Development cost. One Big disadvantage of this technique was that this technique ignores the cost of undetected faults.

For regression testing Alexay et al [24] invented cost model of cost benefits tradeoffs. They performed experiments for selection, reduction and prioritization of test cases. They used cost factors like

- Ca (T) analysis cost
- Ce (T) execution cost
- Cc (T) result checking cost
- Cs (T) selection cost
- Cm (T) maintenance of the test suite's cost.

In experiment for test case prioritization, they focus 2 factors which are cost required for analysis Ca (T) and cost of prioritization Cp (T). When they were performing experiments they divided testing process in two phase, one is preliminary and second is critical phase. These two phases are having different costs. The results have shown, optimal ordering, additional function coverage and total function coverage have maximum savings.

## VII. Factor based Comparison

On the basis of review of different prioritization techniques, in this a comprehensive table is developed. That is, in Table I, different papers are compared on the basis of factors.

TABLE I. Factor based Comparison of Test Case Prioritization

| **S #** | **Methods/ Technique** | **Based on Factor** | **Key Point** | **Formula/Equation/Algorithm/Tool/Technique Used** |
|---|---|---|---|---|
| 1. | Hema Srikanth and Laurie Williams | Based on Customer Requirement | Test cases are being ordered according to WP values. WP is weightage Prioritization. Test cases having the higher values are executed first. | *WP* =€( *PF value* ∗ *PF weight*) |
| 2. | R.kavitha et al. | Based on Customer Requirement | they consider 4 factors<br>1) Priority of requirements assigned by customer<br>2) Code implementation complexity assigned by developer<br>3) Changes in requirements<br>4) Fault impact<br>Test cases are ordered according to values of TCW. TCW represents test case weight. | $RFVi = \sum_{j=1}^{3} Factor\, value\; j / 3$<br><br>$TCW = \left[ \frac{\sum_{x=1}^{i} RFVx}{\sum_{y=1}^{n} RFVy} \right] * i / n$ |
| 3. | Ashraf et al. | Based on Customer Requirement | they consider 6 factors<br>1) Modification in requirement<br>2) Priority of customer<br>3) Complexity in Implementation<br>4) Traceability in requirement<br>5) Time of execution<br>6) Impact of fault | They present a value based prioritization algorithm. To get the net values calculations are being done on the values get from the above 2 levels. These values are further used for ordering of test cases. |
| 4. | Wong et al. | Based on Code Coverage | Propose a technique in which their criterion of test case prioritization is of increasing cost per additional coverage. | They use the tool called ATAC an automatic testing tool for analysis in c. |
| 5. | Rothermal et al. | Based on Code Coverage | Propose 4 coverage based techniques they are total coverage, additional, branch and | they used the<br>Aristotle a program analysis tool. APFD is used |

| | | | | |
|---|---|---|---|---|
| | | | statement coverage. | for measuring the results. |
| 6. | Erlbaum et al. | Based on Code Coverage | Propose the version specific prioritization technique. They present 8 function level techniques they are<br>1) Total function<br>2) Additional function<br>3) Total FEP function<br>4) Additional FEP function<br>5) Total FI function<br>6) Additional FI function<br>7) Total FEP FI functional<br>8) Additional FEO FI functional | APFD metric is used for fault detection. ANOVA and Bonferroni analyses were performed on all techniques. |
| 7. | Amitabh Srivastava and Jay thigarajan | Based on Code Coverage | Proposed a prioritization technique based on binary code. | They gave a system called ECHELON.<br>□ ECHELON prioritize the test cases based on the modification are being done in the program. |
| 8. | Belli et al. | Based on Code Coverage | Proposed techniques in this ordering of relevant events are being done. The events have many features. Events are prioritizing according to the importance of their features. | Graph modal based approach is used for prioritization. Fuzzy c-Mean clustering algorithm is used for erection of events. |
| 9. | Do et al. | Based on Code Coverage | Proposed a technique for regression tetsing in which they want to find out what are the effect on a specific prioritization technique of variation in time constraint and also the effect on cost profit. | $COST = PS * \sum_{i=2}^{n} \left( CS(i) + CO_i(i) + CO_r(i) + b(i) * CV_i(i) + C(i) * CF(i) \right)$ |
| 10. | Leung and white | Based on Cost | Propose a cost modal that compare the various regression strategies. They divide the total cost into<br>two parts<br>• Direct cost<br>• Indirect cost | Direct cost includes<br>1) System analysis cost $Ca$<br>2) Test selection cost $Cs$<br>3) Test execution cost $Ce$<br>4) Result analysis cost $Cr$<br>Indirect cost includes<br>1) Overhead cost<br>2) Tool development cost |
| 11. | Alexey Malishevsky et al. | Based on Cost | Proposed cost modal of cost benefit tradeoffs. They did experiments for selection, reduction and prioritization and presents cost modals for them. | In experiment for test case prioritization they consider two factors cost required for analysis and prioritization $Cp\ (T)$. They divide the testing process in two phase. These two phases have different costs. |
| 12. | Jung-Min-Kim and Adam Porter | Based on chronographic history | It is for regression testing. Their main motive behind this is to show that historical information can be useful for decreasing the cost and it may be beneficial in increasing the efficiency of testing process. | They did comparison of some prioritization methods like<br>LRU, random, safe random. Weakness of their cost modal is that they only take the consequence of last execution of the test case |
| 13. | Fazlalizadeh et al. | Based on chronographic history | They make some changes in the technique of Kim and porter. If resource and time constraint environment is considered they motive is to give faster fault detection. | A comparison was being done with the random ordering.<br>The box plots shows that it has faster fault detection and stability. |
| 14. | Park et al. | Based on chronographic history | Propose an approach for cost-cognizant test case prioritization. that uses the historical information. | A comparison is being done between their technique and functional coverage technique. Results show that in terms of<br>APFD it better than functional level technique. |

## VIII. Comparison of Different Techniques

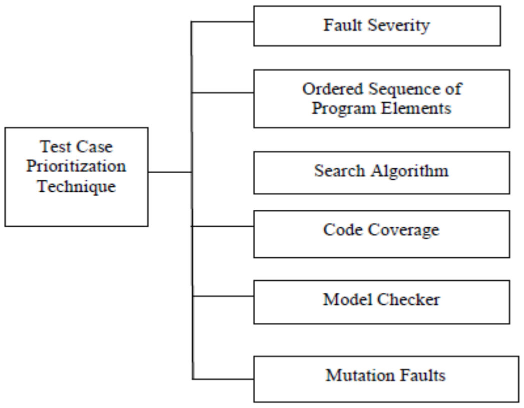

Fig. 6. Techniques of Test Case Prioritization

Figure 6 is showing different techniques of test case prioritization. As, in software development life cycle regression testing is very expensive process. But it makes ensure that the project will satisfy all requirements of stakeholders. In testing phase, about 50% of the total software cost is consumed [25]. Engineers perform assigning test cases preferences through regression testing and execute those test cases which have more significance. The main target of test case prioritization is fault detection. In software testing now a days there are so many techniques which are invented by different researchers to prioritize the test cases. Different techniques are compared which are widely used by the researchers these days. Such techniques are mutation faults, model checker, ordered sequence of program elements, fault localization, fault severity etc. Each technique has own pros and cons. In this section advantages, disadvantages and main idea of techniques are discussed, and on basis of that graph based results is generated.

TABLE II. Comparison of Test Case Prioritization Techniques

| S.No | Technique | Key Idea | Advantage | Disadvantage |
|---|---|---|---|---|
| 1. | Fault Severity | Base on Requirement specification | 1. It enhances the software quality.<br>2. The faults are discovered quickly with high severity.<br>3. It can enhance the fault detection rate.<br>4. Requirements volatility is most important factor. | 1. It does not remove the induced factor of requirements volatility.<br>2. Project scope is limited |
| 2. | Fault Localization | Based on execution information of fault localization. | 1. A postmortem analysis approach.<br>2. Faster failure exposes | 1. It is not much effective.<br>2. The subsequent fault localization may suffer |
| 3. | Mutation faults | Based on changes in program code | 1. Fault detection rate is Improved | 1. Cost reduction is still not significant. |
| 4. | Ordered Sequence of Program Elements | Based on execution frequencies of the program element | 1. Bugs are detected quickly in loops.<br>2. Cost effective approach | 1. Still it's not much effective approach. |
| 5. | APFD | Based on average faults found per minute | 1. Rate of faults detection is easy at system level | 1. This technique not much more efficient in fault detection. |
| 6. | Model Checker | Based on functional model of program test | Prioritization is efficiently applied on the time of creation of test cases | Many factors are still not included such as:<br>1. Actual test case execution costs.<br>2. The costs of potential Faults |
| 7. | Search Algorithm | Based on code coverage | 1. Efficient<br>2. Flexible.<br>3. Program's size does not have impact on prioritizing the test cases. | 1. Still cannot solve large number of test case.<br>2. Sometimes it produces different results. |

## IX. Findings and Results

There are many other techniques that are used for test case prioritization such as Empirical study, coverage based, Decision coverage etc., but did not discussed in this paper. Figure 7 is showing the graph result that is the comparison of seven techniques which are commonly used now a day. Each and every technique has its advantages and disadvantages. These techniques are based on different factors. If the tester wants to pick any technique so he/she can choose any technique according to his/her requirements and specifications.

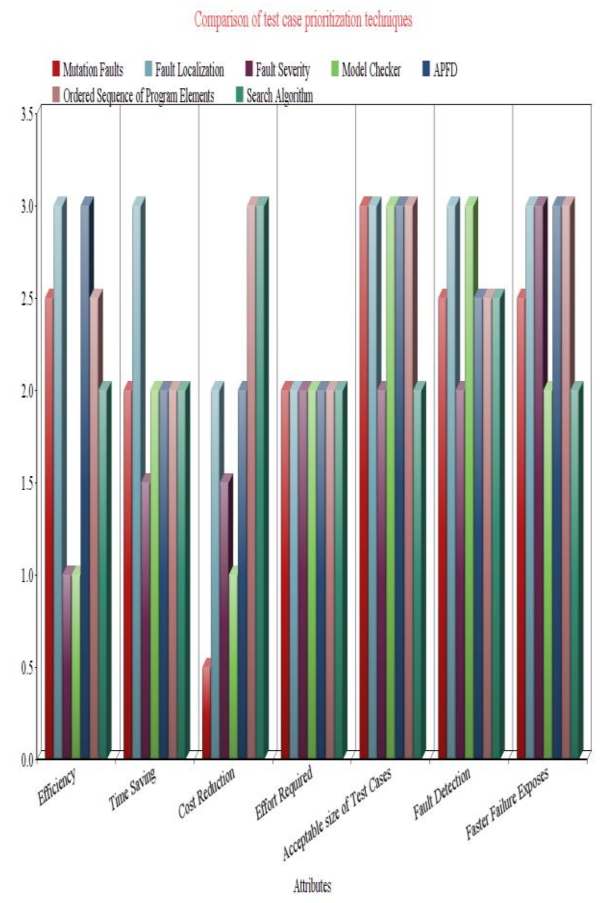

Fig. 7. Comparison of Test case Prioritization Techniques

Testing is a technique for evaluating product quality and also for indirectly improving it, by identifying defects and problems. The testing is conducted in view of a specific purpose (test objective), which is stated more or less explicitly, and with varying degrees of precision. Stating the objective in precise, quantitative terms allows for establishing control over the test process.

## X. Conclusion

Testing is an important and mandatory part of the software development. Software testing is most significant process of software development life cycle. The testing phase involves finding the bugs and removal of defects at earliest if possible. There are different types of testing that software tester adopt according to their requirements such as Mutation, Regression, Stress, Security, Load testing etc. Regression testing is the important type of software testing. When modifications occur in software then there is a need to perform regression testing to check that it doesn't influence the other modules of system. Test case prioritization is done by using different techniques. This paper furnished a comprehensive analysis of different various regression techniques, which primarily focuses on prioritization of test cases. Prioritization means "To schedule or organize" the test cases execution. Few Prioritization techniques are examined in details. Software quality and fault detection in order to more effectiveness and efficiency can be enhanced through regression testing. This paper comprehensively summarizes different research articles (via practitioners) along with their techniques, approaches and methodology they used. Many techniques are investigated and compared that are used for test case prioritization. Each technique has its own advantages and disadvantages. All techniques are tried to explained and concluded so that tester can use any technique according to their requirements and need.

## References

[1] S.Yoo, M.Harman, "Regression testing Minimisation, Selection and Prioritization: A Survey" Wiley InterScience DOI: 10.1002/000,2007

[2] Sahil Gupta, Himanshi Rapria, Eshan Kapur, Harshpreet Singh, And Aseen kumar " A Novel Approach for Test Case Prioritization " at IJCSEA, Vol. 2, No.3, June 2012

[3] Thillaikarasi Muthusamy and Dr. Seetharaman.K "EFFECTIVENESS OF TEST CASE PRIORITIZATION TECHNIQUES BASED ON REGRESSION TESTING" at (IJSEA), Vol.5, No.6, November 2014

[4] Hema Srikanth, Laurie Williams, "Requirements-Based Test Case Prioritization".

[5] Siripong Roongruangsuwan and Jirapun Daengdej. Test Case Prioritization Techniques. JATIT, 2005-2010.

[6] Praveen Ranjan Srivastava. Test Case Prioritization. JATIT, 2008.

[7] B. Korel, G. Koutsogiannakis, "Experimental Comparsion of Code Based and Model model Based Test prioritization," IEEE 2009.

[8] B. Korel, G. Koutsogiannakis, and L.H.Tahat, "Application of System Models in Regression Test Suite Prioritization," in Proceedings of the 24thIEEE International Conference Software Maintenance (ICSM '08) pp.247- 256, 2008.

[9] E.Ashraf, A.Rauf and K.Mahmoat "Value based regression test case prioritization"WCECS 2012, October 24-26 2012.

[10] kumar, Dr Varun,"Sujata and M.kumar,"Test case prioritization using fault severity""IJCST 1, no.1 (2010):67-71.

[11] R.Beena, S.Sarala "code coverage test case selection and prioritization" IJSEA vol.4, no.6, November 2013.

[12] Prakash.N, Rangaswamy "Potentially weighted method for test case prioritization" JCIS 9:18(2013) 7147-715

[13] Praveen Ranjan Srivastava "Test case prioritization", Journal of theoretical and applied information technology, 2005-2008.

[14] R.Kavitha, N.Sureshkumar "Test case prioritization for regression testing based on severity of fault", IJCSE vol.02, no.05 2010, 1462-1466.

[15] Ruchika malhotar, abhishek bharadwaj "Test case prioritization using genetic algorithm" IJCSI: 2231-5292, vol-2, Issue-3, 2012.

[16] A.Srivastava, and J.Thiagarajan, "Effectively prioritizing tests in development environment", Proceedings of the International Symposium on Software Testing and Analysis, pp.97-106, July 2002.

[17] H.Do, G.Rothermel and Kinner, "Empirical studies of test case prioritization in a Junit testing environment", Proceeding of the International Symposium on software Reliability Engineering, pp.113-114, NOV 2004.

[18] R.C. Bryce, A.M. Menon, "Test Suite Prioritization by Interaction coverage", Proceedings of the workshop on domain specific approaches to software test automation (DOSTA), ACM, pp. 1-7, 2007.

[19] H. Do, S. Mirarab, L. Tahvildari, G. Rothermel, "An Empirical Study of the effect of time constraints on the cost benefits of regression testing" Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering,pp71-82,2008.

[20] B.Jiang, Z.Zhang, W.K.Chan, T.H.Tse, "Adaptive Random test case prioritization" In Proceedings of International Conference on Automated Software Engineering, pp:233-243, 2009.

[21] C. L. B. Maia, R. A. F. do Carmo, F. G. De Freitas, G. A. L. de campos and J. T. De Souza, "Automated test case prioritization with reactive GRASP," In Proceedings of Advances in Softwar Engineering, pp.1-18, 2010.

[22] Dennis Jeffrey and Neelam Gupta “Test case prioritization using relevant slices” Department of computer science The University of Arizona TUCSON, AZ85721

[23] Harton K. N. Leung and Lee White “A Cost Modal to compare Regression Test Strategies” CH3047-8|91|0000|0201, IEEE 1991.

[24] Alexey G. Malishevsky, Joseph R. Ruthruff, Gregg Rothermel, Sebastian Elbaum “Cost-cognizant Test Case Prioritization” Technical Report TRUNL-CSE-2006-0004, Department of Computer Science and Engineering, University of Nebraska – Lincoln, 2006.

[25] Srikanth, Hema, Laurie Williams, and Jason Osborne. "System test case prioritization of new and regression test cases." In Empirical Software Engineering, 2005. 2005 International Symposium on, pp. 10-pp. IEEE, 2005.