International Journal of Engineering Research and Development
***e*-ISSN: 2278-067X, *p*-ISSN: 2278-800X, www.ijerd.com**
Volume 3, Issue 12 (September 2012), PP. 85-90

Prioritization based on test suites by analysing automated testing strategies to minimize selected test process

R. Deepa[1], D. Rampriya[2], T. Prasath[3]
[1,2,3]*Final year M.Tech (CSE), Christ college of Engineering and Technology, Pondicherry.*

Abstract—Software testing is a process of refining faults and errors in a software system or project. Software testing is done by both manually and systematically. Automated testing of software is mostly followed in today's industry. To achieve this, the concern project is divided into various test suites and then testing is done. These test suites should be minimized, selected and prioritized to make the testing process more efficient. This paper gives a brief description about the techniques for test suite minimization, test suite selection and test suite prioritization. A detailed analysis on various prioritization techniques will help the tester to separate and order the test suites for the testing process.

Keywords— Test Suites, Automated testing, test suite minimization, test suite selection, test suite prioritization.

I. INTRODUCTION

In business management, software plays a significant part. It panels equipment maintenance supervision, logistic, resource provision, business procedure, financial dealings, accounting, communication, human resources etc. Development has to be considered in the case of the rapid improvement in the software usage. Improper calculation, inappropriate data edits and ineffective data edits, incorrect matching and merging of data, unfitting results that yields during data search, data relationship based on incorrect processing, business rules are coded and implemented incorrectly etc .So a cautious management has to be considered. Model and processes to central the production of well sustainable software is the aim of software engineering which is liable. Based on the input selected in a replicated or actual situation a software execution was done which is named as software testing.

Fault detection, confidence establishment in software, evaluation of software properties are some of the aim of software testing. Certain difficulties are considered in the software testing some of them are output correctness determination, internal to external state result comparison, adequate state consideration and determination, actual says about the software based on its determination, characteristics analysis based on measuring performance, and strategies for test comparison. Some small stages of testing are testing related to unit, testing related to feature, testing related to integration and large stage testing's are based on system testing, end-to-end testing, operations readiness testing, beta testing, load testing, stress testing, performance testing, reliability testing, regression testing strategies.

Verification and validation plays an important role in software testing strategies. Reviews and meetings that involves for evaluating documents, requirements, specifications, code and plans are considered to be verified. Verification takes place before actual testing is validation. Until the software becomes error free, verification and validation process was sustained. Component identification, test specification design, test specification review are some deliberations in test case development and design. Code review, test execution and evaluation, performance and simulation are some of the test execution case.

Implementing new technologies and additional features for the report and application improvement are some of the test process analysis involved. Certain testing methodologies are black box testing, white box testing, incremental testing, thread testing etc. WinRunner, LoadRunner, rational robot, QTP, test director are examples of test tools and utilities. Test can be executed efficiently by dividing the software program into various test suites. The test suites will be minimized first for the testing process. Then it will be selected and prioritized for the testing process. To achieve this various techniques of test suite prioritization have been ruled out.

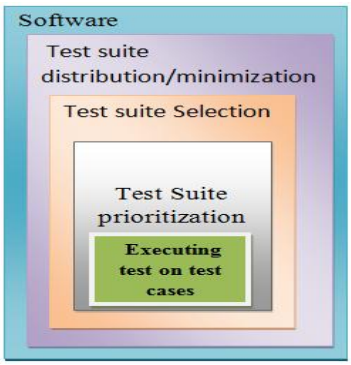

Fig.1 Software Testing Process

A. Techniques for Test Suite Minimization

Superfluous test case is used to locate and detached from the test suite is known as test suite minimization [2]. Test cases become superfluous because (1) changes in the program will affect the input and output relation in the program ,(2) for particular program this test were developed and it has been customized.(3)using software coverage it cannot predict the formation. Every technique performs as a transitory subset of the test suite, in spite of only minimization techniques will eliminate the test cases where minimization or test suite reduction means a constant eradication. Chavatal proposes all necessities which as been enclosed using greedy algorithm for choosing test case. This algorithm gives a elucidation for the minimum set-cover difficulty.

By using greedy approach, a possible weak point in early assortment will lead to rendered surplus test case after selected and there is a bind among numerous test cases, single test case is arbitrarily chosen. More amount of testing requirements will be collected by this algorithm. Offutt, Pan, and Voas propose about the double of the least hitting set problem and issue related to the test suite minimization. For performance predetermined ordering is used instead of a variety of test case orderings. Harrold, Gupta, and Soffa cover the identical set of difficulty by choosing a minimal split of test case. Tallam and Gupta propose a relation among test cases and testing requirements based on the hierarchical clustering, greedy approach urbanized a perception lattice in order to go beyond the limitation.

Jeffrey and Guppta deal with the discriminating redundancy in the minimized test suites and enchanting in explanation numerous set of testing difficulty using conventional single-criterion minimization technique. Black, Melachrinoudis, and Kaeli, proposed that integer linear programming gives best solution using single-criterion conventional test suite minimization. This approach is used to handle the kind of minimization problem and it is used for error finding. Hsu and Orso [HO09] such as Black et al. multi-criteria test minimization solver utilize of Integer Linear Programming (ILP).

B. Techniques for Test Suite Selection

Test suite minimization issue is related to test case selection or regression testing selection issue. While selecting a separation of test case from the test suite both techniques consist of similar crisis. Amendment aware is done in the best part of collection technique where test case selection technique aims to reduce the cardinality of a test suite. Original size of the program focuses on the detection of the modified modules, where assortment is not momentary. Based on Rothermel's et-al identify the alteration traversing test cases in the given test suite focus on regression testing [4]. Special criteria have been used for a variety of techniques they are digit encoding, Program slicing, vibrant slicing, chart-walking, textual variation in resource code and modification finding. Harrold, Gupta and sofa partial a convention change used in program slicing technique to identify definition use-pairs. Without complete data flow analysis often very expensive testing is done on the identification of explanation utilize the pair by slice technique. The data flow change is not correlated to spot change where the test case selection technique divide is based on the data flow analysis. Agarwal, Horgan, Krauser, and London propose a special program slicing technique based on the test case selection technique. Set of statements is executed by the given test case based on the finishing copy using execution slice of a program. A productivity statement gives the collection of statements in the execution slice for the dynamic slice of the program. The program productivity is not affected in a execution slice so dynamic slice is a detachment of a execution slice. Agarwal et al decided to locate the additional slicing criteria: estimated relevant slice and a significant slice. Unusual productivity is produced where all the predicate statement in the program, if analysed in a different way where a significant slice of a program regarding test case is a dynamic slice. The dynamic slice with all the predicate statements in the execution slice which would fabricate different output and it include predicates where an approximated relevant slice is an extra additive approach. Chen, Rosenblum and VO to choose test cases alteration based technique is used, test pipe is called as a testing structure. Cursor management is one of the faults of the test pipe.

C. Techniques for test suite Prioritization

Prioritization technique is test cases with high prospect are executed foremost, and the test cases with a little opportunity of finding defects are executed at the last part based on the distinct criteria it classify the test cases. Test cases with great priority are executed first and the test cases with fewer priorities are executed using regression testing process, this method sort the test cases in order to afford for the testers. In order to get the convention coverage at the best rate feasible, tester might desire to rearrange the test cases.

Rothermel, UNtch, Chu and Harrold propose nine diverse test case prioritization techniques using its advance : organic(no Prioritization), random (randomized ordering), best(optimize rate of error finding is structured), entire-declaration (reporting of statement is prioritize in order), additional - statement (prioritize in order of reporting of statements not yet roofed), entire- branch(reporting of branches is prioritize in order), extra- division (prioritize in order of coverage of branches not yet covered), entire -FEP (order of whole probability to prioritize in revealing faults), FEP-further (property of prior test is adjusted). Do,Rothermel, and Kinneer proposed JUnit to coverage base technique which is the most popular unit testing structure. Discovery of faults is enhanced in JUnit test cases [7], effect show that prioritized execution.

Test case prioritization techniques depend on surrogates. Mirarab and Tahvildari proposed feedback method and a Bayesian Networks (BN) utilize a prioritization crisis [8]. BN is a directed acyclic graph consists of three basics: random variables representing nodes, probabilistic dependencies between those variables representing arcs, and a conditional Probability Distribution Table (CPT) for every variable, which comprise the conditional probabilities of outcomes of its variable known the valued of all its parents. BN finding an error using different numerous sources of information, it is used to predict the probability of test case. Mirarab et al estimate the performance of BN approach.

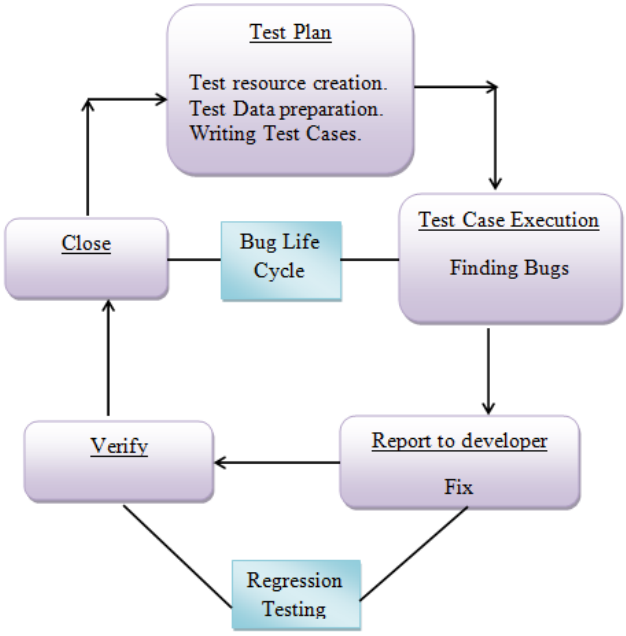

Fig.2 Software testing life Cycle

D. Many Test Suite Prioritization Methods:

Prioritization is basically made for single set of test cases but as per the evolution multiple test suites [1] has to combined or re-arranged for testing. Basically the test suite defines a set or collection of similar test cases for prioritization many test suites the following flow chart can be referred.

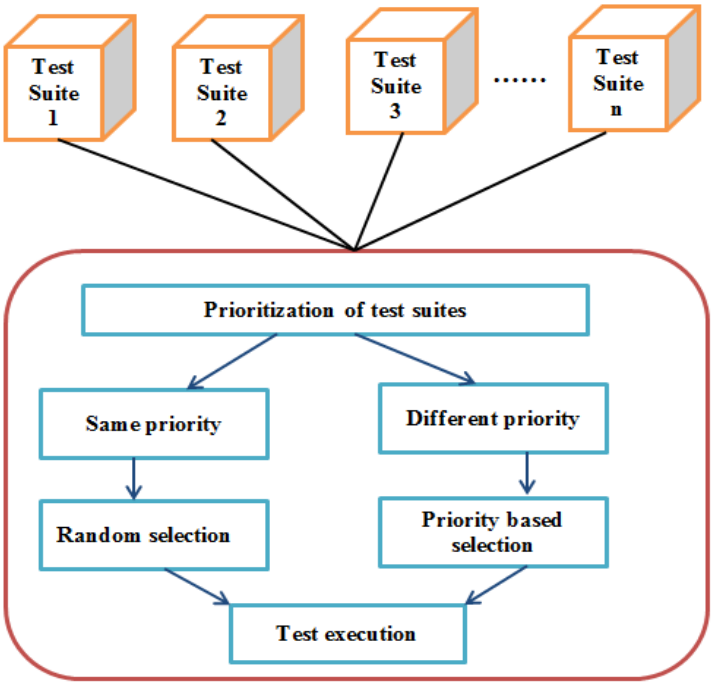

Fig.3 Multiple test suite prioritizations

This flow chart initially prioritizes the test suites depending on time and cost factors then the test suites with same priority should be selected and ordered randomly. If not the test suite with higher priority should be ordered first. Thus prioritization of multiple test suites can be carried out.

E. Coverage-based prioritization techniques:

Code coverage analysis provides a measure for software testing based on the test coverage analysis for the experts. The quantity of source code during the testing of a program was defined by it. A direct code inspection was taking out by this testing which is a form of white box testing. Finding area of a program not exercised by a set of test case, to increase coverage additional test case creation, code coverage quality measures determination, decreasing the coverage by redundant

test case identification are some of the process of coverage-based techniques. White box or structural testing techniques are the base of coverage-based technique.

Comparison between the test program behaviours with the apparent intention was given by the structural testing for the source code. Considering the possible pitfall into an account in the structural and logic examination was done how the program works. Without the attention of internal work, functional testing inspects what the program achieves. Prioritize test case based on coverage criteria was provided by a method of coverage base technique such as requirement coverage, additional requirement, total requirement and statement coverage.

Another important coverage based prioritization technique is function-based, this function prioritization consists of various techniques such as total function coverage prioritization, total binary difference function coverage prioritization and additional binary difference function coverage prioritization was given by Sebastian et-al.

Various suggestions on coverage based prioritization techniques was given by,

1. Rothernel proposes nine approaches they are random approaches, optimal prioritization, total branch coverage prioritization, total statement coverage prioritization, additional statement coverage prioritization, total fault-exposing potential prioritization and additional fault exposing-potential prioritization.
2. Leon presents four different techniques they are test suite minimization, prioritization by additional coverage, cluster filtering with one-per-cluster sampling and failure pursuit sampling.
3. Elbaum's gives an additional coverage prioritization approach, which runs a greedy coverage maximum algorithm which was not been prioritized on a set of test cases.

F. Cost Effective-Based prioritization techniques

Based on the cost, prioritization test based methods was given by a techniques of cost effective-based. It involves in cost of analysis and cost of prioritization. Regression test case based on cost was given by white and Leung. Cost of executing and validating test cases and the cost of performing analysis were incorporated by cost of regression testing that supports test selection and relative effectiveness was provided by a way of compare tests. Regression test selection techniques use the above mentioned model for its effectiveness. Because of discarded tests, cost of overlooking faults is not considered by Leung's model. Cost into account based on the cost models for prioritization was given by Alex Malishevsky.

G. History-Based prioritization techniques:

Based on the test execution history prioritization was carried out by a history-based technique [10] and its method. Regression error was identified by the developers at the testing phase this condition occurs during the modifying of software to now to know the unmodified code. To repeat all prevailing test case a simplest regression testing approaches was used. A methodology for subset of test cases are selected and rerun in a regression test selection technique. Cost-benefit tradeoffs of regression test selection techniques are a essential anxiety of regression testing research testing case in optimal test selection. These above techniques have certain limitations in it. Creating base and modified versions of a system and accompanying test suites find out by researches based on the consideration of the limitation.

Size and effectiveness of the original test suite was analysed by a selection algorithm. One-time activity slightly than as the unceasing process and non-consideration of real world time and resource compel are considered as an two serious curb for the mentioned approach. Ordered sequence of testing gathering was the circumstance of regression testing two inferences stream are detected that is alteration between what an perfect regression tester must do and what the real one can afford to do. To expand long run regression testing performance, historical test case performance data which current RTS and test case prioritization techniques was ignored.

H. Chronographic history-based prioritization techniques:

According to Jung-Min this method is based on concepts taken from statistical quality control and statistical forecasting and test case's prior information are used to increase or decrease the probability that is used in present testing session[10]. Selection probabilities of each test case were defined by Kim at time *t*. Source code information is not available in a black box testing environment. In such condition, experts only have output of test cases and other run-time information accessible, such as the running time of test cases prioritization technique was proposed.

I. Knowledge Based prioritization technique:

Depending on the knowledge of the experts and experience gained from various testing processes carried out before, the knowledge based prioritization techniques were formed. In knowledge based prioritization, the test suites were basically prioritized depending on the various types of errors encountered in similar processes before. Human interaction is high in this prioritization method. Based on the various kinds of fault occurred before the fault-based prioritization technique is carried out.

J. Fault-Based prioritization technique

Fault-detection of test cases is the technique in which presented test suite prioritization generally rely on the convention coverage in order or chronological effecting information that provide as indicators. Such indicators are not speculative determined; thus, they do not essentially afford noise estimates and it is experimental in nature. If the resource convention is not presented these techniques are not valid. Relationship between the test cases and the faults in the given fault model, based on which the test cases are generated, fault-based prioritization of test cases which directly utilizes the speculative information of their fault- detecting ability.

Fault-based prioritization is useful to the testing of the execution of logical expressions against their provision. The efficacy of prioritization techniques is evaluated using two diverse metrics and it is authorize using the experimental study.

This technique outperforms other techniques below study and it is assessed by two diverse metrics. It summary the effort need in testing. The fault-based prioritization is flexible to other fault-based testing strategies.

K. Model based test Case Prioritization:

In regression testing the system which has undergone modification should be tested once again. The process of retesting is difficult for the developers, since the size of the test suites were large. Due to this issue the developers need a fault detection technique which is very fast. To achieve this model based test prioritization [5] is proposed. In this model-based testing method, the information about the model is collected at the time of its execution to make prioritization. Basically model-based testing comes under Black box testing. Model based test case prioritization is used to achieve fast detection of faults and less expensive when compared with the code based test case prioritization. It also senses the correctness of the information provided by the tester/developer.

A system model will be used to gather some aspects about various behaviour of the system, based on which test cases were created. Test generation for models with typical languages is done using various aspects such as UML, SysML, and Main stream programming languages, finite machine notations and mathematical formulation.

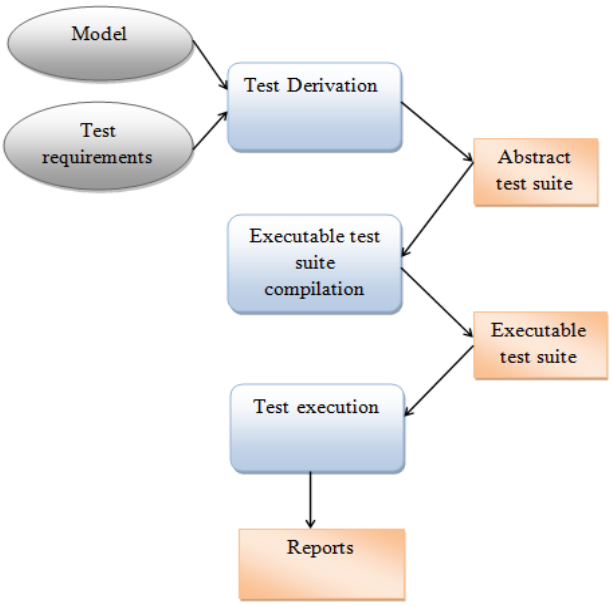

Fig.4 Model based testing

L. Component Based Software development:

The component based Software Development (CBSD) is a method by which the test cases are prioritized as per the dependency of components [11]. This component dependency based test case prioritization is mainly achieved using the Greedy approach. This kind of prioritization is done basically using an object interaction graph (OIG) which can be derived by interlinking the components of the UML sequence diagram.

The calculation of total number of interactions among the inter component objects as well as the intra component objects is done by traversing the OIG. Here the test cases were ordered depending on the number of interactions made by the objective functions. The test cases with the higher number of interactions will be given higher priority for testing. The advantages of CBSD are reduced lead time and enhanced quality.

II. CONCLUSIONS

The technique for test suite minimization gives methodology to separate the program into various test cases. The selection technique helps to group the test cases related with each other and the prioritization techniques were used to give priority to the test cases. This prioritized order is used to carry out the testing process. To make this prioritization of test suites more efficiently various prioritization techniques were discussed in this paper. Following these techniques depending on the project, the software testing can be carried out to find maximum number of errors.

REFERENCES

[1]. Siripong Roongruangsuwan, Jirapun , "Test Case Prioritization Techniques" 2005 – 2010 Jatit & Lls.

[2]. S. Yoo, M. Harman, "Regression Testing Minimisation, Selection and Prioritisation : A Survey", Softw. Test. Verif. Reliab. 2007; 00:1–7 (DOI: 10.1002/000).

[3]. José Campos "Regression Testing with GZoltar: Techniques for Test Suite Minimization, Selection, and Prioritization" 14th February, 2012.

[4]. R.Krishnamoorthi and S.A.Sahaaya Arul Mary, "Regression Test Suite Prioritization using Genetic Algorithms**"** International Journal of Hybrid Information Technology Vol.2, No.3, July, 2009.

[5]. Fevzi Belli, Mubariz Eminov, and Nida Gokce, "Model-Based Test Prioritizing –A Comparative Soft-Computing Approach and Case Studies" KI 2009, LNAI 5803, pp. 425–432, 2009.
[6]. Rene´e C. Bryce, Sreedevi Sampath, Member, IEEE Computer Society, and Atif M. Memon, Member, VOL. 37, NO. 1, JANUARY/FEBRUARY 2011.
[7]. Do, Hyunsook; Rothermel, Gregg; and Kinneer, Alex, "Empirical Studies of Test Case Prioritization in a JUnit Testing Environment" (2004). *CSE Conference and Workshop Papers.* Paper 139.
[8]. Kristen R. Walcott, "Prioritizing Regression Test Suites for Time-Constrained Execution Using a Genetic Algorithm". Shin Yoo & Mark Harman, "Using Hybrid Algorithm For Pareto E_cient Multi-Objective Test Suite Minimisation" November 29, 2009.
[9]. ung-Min Kim, Adam Porter, "A History-Based Test Prioritization Technique for Regression Testing in Resource Constrained Environments".
[10]. Arup Abhinna Acharya, Durga Prasad Mohapatra, Namita Panda, "Model Based Test Case Prioritization for Testing Component Dependency in CBSD Using UML Sequence Diagram" *(IJACSA) International Journal of Advanced Computer Science and Applications,Vol. 1, No. 6, D.*
[11]. Arup Abhinna Acharya, Sonali Khandai, Durga Prasad Mohapatra "A Novel Approach for Test Case Prioritization using Business Criticality Test Value" *International Journal of Computer Applications (0975 – 8887) Volume 46– No.15, May 2012.*
[12]. Shifa-e-Zehra Haidry and Tim Miller "Using Dependency Structures for Prioritisation of Functional Test Suites".