International Journal of Engineering Research and Development
e-ISSN: 2278-067X, **p**-ISSN: 2278-800X, www.ijerd.com
Volume 7, Issue 3 (May 2013), PP. 59-62

Test Case Prioritization Using Genetic Algorithm

Ms.Kirandeep Kaur[1], Mr.Vinay Chopra[2]
[1](CSE, KCCEIT, Nawanshahrr /PTU, INDIA)
[2](CSE, DAVIET, Jalandhar /PTU, INDIA)

Abstract: Test case prioritization is one of the techniques used to reduce the rate of fault detection. Test case prioritization techniques help engineers execute regression tests in an order that achieves testing objectives earlier in the testing process. Test case prioritization techniques schedule test cases for execution in an order that attempts to increase their effectiveness at meeting some performance goal. Various goals are possible; one involves rate of fault detection. Rate of fault detection measure of how quickly faults are detected within the testing process.

Keywords: Testing, Regression Testing, Test case prioritization, Genetic Algorithm.

I. INTRODUCTION

Testing is an important engineering activity responsible for a significant portion of the costs of developing and maintaining software. Testing is the process of analyzing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item. Testing software is operating the software under controlled conditions:

1. To verify that it behaves "as specified",
2. To detect errors, and
3. To validate that what has been specified is what the user actually wanted.

In general, however, testing techniques are heuristics and their performance varies with different scenarios; thus, they must be studied empirically.

II. TESTING LEVELS BASED ON SOFTWARE ACTIVITY

Tests can be derived from requirements and specifications, design artifacts, or the source code. A different level of testing accompanies each distinct software development activity:

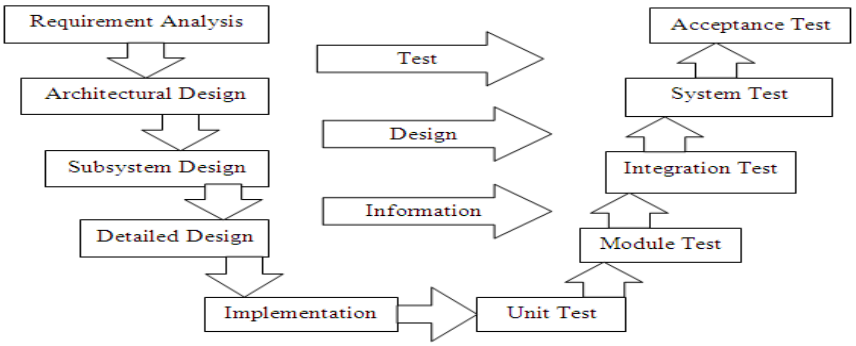

Fig 1: Software development activities and testing levels

Fig 1 illustrates a typical scenario for testing levels and how they relate to software development activities by isolating each step.

1. Acceptance Testing – access software with respect to requirements. The **requirements analysis phase** of software development captures the customer's needs. **Acceptance testing** is designed to determine whether the completed software in fact meets these needs. In other words, acceptance testing probes whether the software does what the users want. Acceptance testing must involve users or other individuals who have strong domain knowledge.
2. System Testing – access software with respect to architectural design. The **architectural design phase** of software development chooses components and connectors that together realize a system whose specification is intended to meet the previously identified requirements. **System testing** is designed to determine whether the assembled system meets its specifications. It assumes that the pieces work individually, and asks if the system works as a whole.
3. Integration Testing – access software with respect to subsystem design. The **subsystem design phase** of software development specifies the structure and behavior of subsystems, each of which is intended to

satisfy some function in the overall architecture. Often, the subsystems are adaptations of previously developed software. **Integration testing** is designed to assess whether the interfaces between modules in a given subsystem have consistent assumptions and communicate correctly. Integration testing must assume that modules work correctly.

4. Module Testing – access software with respect to detailed design. The **detailed design phase** of software development determines the structure and behavior of individual modules. A program unit, or procedure, is one or more contiguous program statements, with a name that other parts of the software use to call it. **Module testing** is designed to assess individual modules in isolation, including how the component units interact with each other and their associated data structures.
5. Unit Testing – access software with respect to implementation.

III. REGRESSION TESTING

Regression testing is an expensive testing process used to validate modified software and detect whether new faults have been introduced into previously tested code. One goal of prioritization is to increase a test suite's rate of fault detection. To reduce the cost of regression testing, software testers may prioritize their test cases so that those which are more important, by some measure, are run earlier in the regression testing process. There are following Methods of Regression Testing:

Table 1: Methods of Regression Testing

Methods	**Descriptions**
Retest all	All the tests in the existing test suite should be re-executed. This is very expensive as it requires huge time & resources.
Regression test selection	Re-executing the entire test suite, it is better to select part of test suite to be run. Test cases selected can be categorized as: • *Reusable Test Case:* It can be used in succeeding regression cycles. • *Obsolete Test Case:* It can't be used in succeeding regression cycles.
Test case prioritization (TCP)	Selection of the Test Cases based on priority will greatly reduce the regression test suite.

IV. TEST CASE PRIORITIZATION (TCP)

Test case prioritization techniques schedule test cases for execution in an order that attempts to increase their effectiveness at meeting some performance goal. Various goals are possible; one involves rate of fault detection. Rate of fault detection measure of how quickly faults are detected within the testing process. An improved rate of fault detection during testing can provide faster feedback on the system under test. Each test cases are assigned a priority. Priority is set according to some criteria like Early Fault detection, Execution time, etc. The effectiveness of any system is calculated on the basis of fault detection when software evolves. An improved fault detection during testing can provide faster feedback on the system under test. Test case prioritization is one of the techniques used to reduce the fault detection. There are following Prioritization Factors:

Table 2: Prioritization Factors

S.NO	**PRIORITIZTION FACTORS**
1	*Customer-Assigned Priority (CP):* It is the measurement of importance to the customer's need. Customer's need vary from 1 to 10 and assigned by customer itself. Where 10 is used to identify the highest customer priority.
2	*Implementation complexity (IC):* No. of faults increases as the requirement is become high in implementation complexity. Value from 0 to 10 assigned by the developer on the basis of its application complexity and by using larger value higher complexity is denoted.
3	*Changes in Requirements (RC):* The no. of times of requirement has been changed with respect to when the requirement was first introduced. It also ranges from 1 to 10. **Formula is: Ri= (M/N)*10** Where i= No. of changes for any requirements. Ri=Change in requirements. M=No. of times i^{th} requirement change. N=Max. no. of requirement.
4	*Fault Impact of Requirements (FI):* Development team to identify the requirement which has had customer reported failures. The developer can use the previous data to reduce the fault

	impact.
5	*Traceability (TR):* Traceability calculates the mapping that exists between requirements and test. It helps in determining whether a requirement is enough tested is difficult for the testers if the test cases are not relevant to particular requirements. A problem which is very common is insufficiency of traceability. If traceability is not good or poor that may cause project over runs and failures.
6	*Execution Time (ET):* Execution time of the test case as cost of test case. Test case costs should have a large impact on the TCP. In terms of test case cost, It can be relate to the resources, such as execution time of test case, Hardware costs or even engineers salaries. In our case, test case cost is the execution time of test case.

V. TEST CASE PRIORITIZTION TECHNIQUES

There are following TCP techniques :

1. *No prioritization:* No prioritization tests in test suite. It is simply the application of no technique apply.
2. *Random Prioritization:* A test case is selected randomly for execution.
3. *Optimal Prioritization:* A test case which determines maximum new faults is executed first.
4. *Code-coverage based prioritization:* It covers maximum functions, classes or code fragments of software program within minimum time have highest priority in the prioritize test suite. If multiple test cases cover the same number of functions, orders these randomly.
5. *Mutation based prioritization:* It is based on the fault-based technique. A mutant of statement is generated (automatically) by randomly changing the statement.

VI. GENETIC ALGORITHMS

G A are inspired by Darwin's theory about evolution. Simply said, a solution to a problem solve by G A is evolved. Each individual is represented by a sequence of variables/parameters (called genes), known as the chromosome. The chromosome encodes a possible solution to a given problem. The encoding can take many forms, for example, binary, real-valued, or character-based.

1. *Selection:* A selection depending on the fitness value decides which individuals are to be used as the "parents" for producing the next generation.
2. *Crossover:* Crossover is a genetic operator that combines two individuals (the parents) to produce a new individual (the offspring).
3. *Mutation:* After a crossover is performed, Mutation take place. This is to prevent falling all solutions in population into a local optimum of a solved problem. Mutation changes randomly the new individual.

Table 3: Advantages & limitations of Genetic Algorithms

ADVANTAGES OF GENETIC ALGORITHMS	LIMITATIONS OF GENETIC ALGORITHMS
1. It solve every optimization problem	1. They find a solution to evolution.
2. Solve problem with multiple solution.	2. A better solution is only in comparison to other solutions.

VII. CONCLUSION

In Previous studies have shown that several prioritization techniques can significantly improve rate of fault detection. But we have shown that the effectiveness of these techniques varies considerably across various attributes of the program, test suites, and modifications being considered. This variation makes it difficult for a practitioner to choose an appropriate prioritization technique for a given testing scenario. The prioritization techniques also include coverage and change attributes but it is observed that engineers choosing to prioritize for both coverage and change attributes may actually achieve poorer rates of fault detection.

REFERENCESS

[1]. G. Rothermel, R.H. Untch, C. Chu, & M. J Harrold, **"Test Case Prioritization: An Empirical Study"**, Proceedings of the International Conference on Software Maintenance, Oxford, UK, IEEE, Vol. 20, Issue no. 6, pp: 1-10, 1999.

[2]. Praveen Ranjan Srivastava, "**Test Case Prioritization",** Journal of Theoretical and Applied Information Technology, pp: 178-181, 2008.

[3]. P. Ammann, P. E. Black, and W. Ding, **"Model Checkers in Software Testing Technical report"**, National Institute of Standards and Technology, 2002.

[4]. Sebastian Elbaum,¤ Gregg Rothermel,*y* Satya Kanduri,*z* Alexey G. Malishevsky, "**Selecting a Cost-Effective Test Case Prioritization Technique",** pp: 1-26, 2004.

[5]. Sujata, Mohit Kumar, Dr. Varun Kumar, **"Requirements based Test Case Prioritization using Genetic Algorithm",** IJCST Vol. 1, Issue 2, pp: 189-191, 2010.

[6]. Ruchika Malhotra, Arvinder Kaur and Yogesh Singh, **"A Regression Test Selection and Prioritization Technique,"** Journal of Information Processing Systems, Vol.6, No.2, pp.235- 252, Jun 2010.

[7]. Peter M. Kruse1 and Kiran Lakhotia2, **"Multi Objective Algorithms for Automated Generation of Combinatorial Test Cases with the Classification Tree Method"**, pp: 1-6, 2011.

[8]. Chhabi Rani,Rajibb mall, **"Test Case Prioritization of object-oriented programs",** Setlabs briefings, Vol.9, No.4, pp. 31-40, 2011.

[9]. S. Raju, G. V. Uma, **"Factors Oriented Test Case Prioritization Technique in Regression Testing using Genetic Algorithm",** European Journal of Scientific Research, ISSN 1450-216X, Vol.74, No.3, pp. 389-402, 2012.

[10]. Miso Yoon, Eunyoung Lee, Mikyoung Song, Byoungju Choi, **"A Test Case Prioritization through Correlation of Requirement and Risk"**, Journal of Software Engineering and Applications, Vol. 5, 823-835, 2012.

[11]. Prem Parashar, Arvind Kalia, Rajesh Bhatia, "**How Time-Fault Ratio helps in Test Case Prioritization for Regression Testing",** Int.J. of Software Engineering, IJSE Vol.5 No.2, pp: 25-35, 2012.

[12]. E. Ashraf, A. Rauf, and K. Mahmood, **"Value based Regression Test Case Prioritization",** Proceedings of the World Congress on Engineering and Computer Science, ISBN: 978-988-19251-6-9, ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online), Vol. 1, 2012.