Volume 3, Issue 8, August 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Metric Based Prioritization Approach for Basic Path Testing

Seema [*]
Student, N.C.College of Engineering Israna
Panipat, Haryana, India

Sukhvir Singh
Associate Professor, N.C.College of Engineering Israna
Panipat, Haryana, India

Abstract*— White Box testing is the internal Analysis approach to test a particular code or the system. While analyzing the code, it is required to separate the code blocks respective to the code criticality. There are different vectors to categorize these code statements or blocks called prioritization approaches. The priorities to these modules are assigned based on associated test case criticality, interaction between modules, dependency vector etc. Based on this prioritization, test cost of a software module is estimated. In this paper, some of these prioritization approaches are been discussed so that effective path testing can be performed.

***Keywords*— Path Testing, Prioritization, White Box Testing, Path Coverage, Code Criticality**

I. Introduction

Testing of a software or the code module is basically required to detect the errors in the software product or code as well as to estimate the cost of efforts. The main objective of testing is to identify the problems, error, bugs associated with the software system. To identify these bug, errors, different kind of testing approaches are been implemented in different ways. Testing is about to analyzing the software or the module behaviour. In case of an error there may be change in the format of out, some unexpected behavior from system, or some value different from the expected is obtained. These errors can due to wrong analysis, wrong design, or some fault on developer's part [1]. All these errors need to be discovered before the system is implemented at the customer's site. Because having a system that does not perform as desired be of no use. All the effort put in to build it goes waste. So testing is done. And it is equally important and crucial as any other stage of system development. For different types of errors there are different types of testing techniques.[2]
Fundamental process of testing:

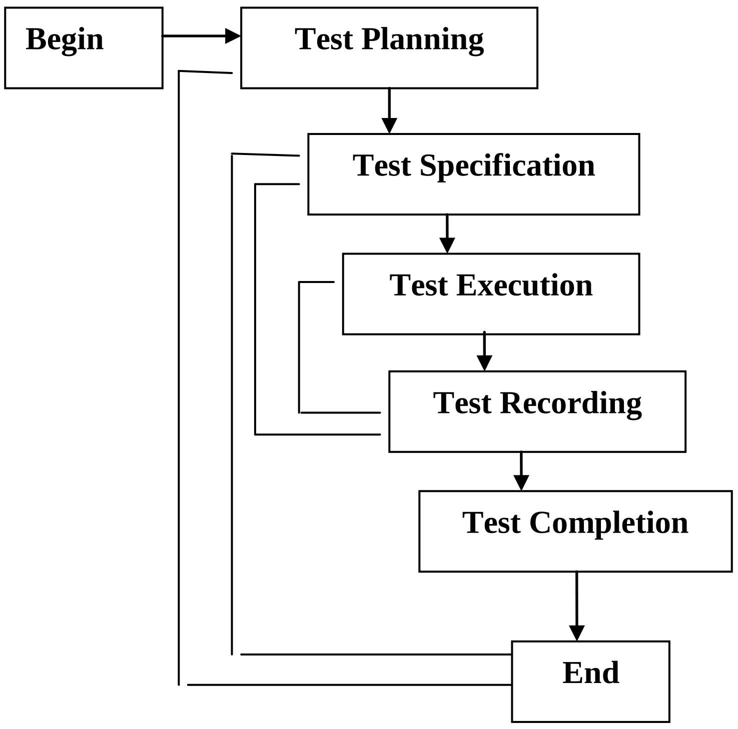

Figure 1: Testing Process

First of all the objective of the testing should be clear. We can define testing as a process of executing a program with the aim of finding errors. To perform testing, test cases are designed. A test case is a particular made up artificial situation upon which a program is exposed so as to find errors. So a good test case is one that finds undiscovered errors. If testing is done properly, it uncovers errors and after fixing those errors we have software that is being developed according to specifications[3]

One of the most common aspect for testing the particular software module is White box testing. It is basically an internal analysis approach in which each the software paths are analyzed to discover the bugs within the system. To analyze the software quality, effectiveness of each statement and the code block will be estimated under the error pruning concept. White box testing is used to ensure that the code is complete and to the correct standard of the software mechanism. Statistics haves proven that by using a complete and precise systematic test design, will ensure that the majority of bugs within the system will be identified. When looking at white box testing in more detail it involves checking and ensuring that every program statement is error free.[7]

White box testing (WBT) of software is designed for close examination of procedural detail. Providing test cases that exercise specific sets of conditions and/or loops tests logical paths through the software. Unfortunately, even for few LOC, the number of paths become too many and presents certain logistic problems. Due to this, a limited number of important logical paths can be selected and exercised. Important data structures can be probed for validity. WBT is a test case design method that uses the control structure of the procedural design to derive test cases. The test cases derived from White box testing methods will:

- ❖ Guarantee that all independent paths within a module have been exercised atleast ones.
- ❖ Exercise all logical decisions on their true and false side.
- ❖ Execute all loops at their boundaries and within their operational bounds.
- ❖ Exercise internal data structures to ensure their validity.

WBT needs to be adopted under unit level testing strategy. It can be adopted to a limited extent under integration testing if situation warrants for it. Basic path testing and control structure testing are some of the most widely used White box testing technique. It is the testing method in which the user will test the application structure how it is acting. Usually developers will perform the White box testing.

II. LITERATURE REVIEW

He *et al.*[10] proposed a fast path selection approach based on graph partition in this paper. First, a critical path graph (CPG) was generated to implicitly enumerate almost all candidate critical paths, and then the CPG was partitioned into several sub graphs which contain limited numbers of paths using two graph partition approaches. After that, Monte Carlo simulation was applied on each sub graph for path selection. At last, according to the partition topology of the CPG and path sets selected from each sub graph, a path set for this original CPG was generated using Union and Cartesian product operations for testing SDDs. Jiang *et al*.[11] selected the test cases that can test part of changes, and then did the reduction for these selected test cases. Based on the requirements of static path coverage testing, this article proposed an algorithm for test suite optimization. According to the algorithm, Author can find a change point with the minimum set of using cases for regression testing, and give application examples. The results show that, the algorithms can significantly reduced the size and the cost of the test suite for regression testing, and achieved good cost effectiveness. He *et al*.[12] carried out the work in which the path selection problem was converted to a minimal space intersection problem, and a greedy path selection heuristics was proposed, the key point of which was to calculate the probability that the paths in a specified path set all meet the delay constraint. Statistical timing analysis technologies and heuristics were used in the calculation. Experimental results show that the proposed approach was time-efficient and achieved a higher probability of capturing delay failures in comparison with conventional path selection approaches.

Zheng *et al*.[13] in this article, according to the structure of C program, a new method of generating the global static function call path and getting the function call graph as well was presented through analyzing control flow and data flow. Author empirically showed that this algorithm had high accuracy and efficiency and can help to improve coverage rate of function call path coverage testing and software stability. Wang[14] proposed a method of dividing the program under test (PUT) into a several program segments with small cyclomatic complexity. This paper analyzed the relationship between the input variables and the fragments, gave the approach of processing the split point and illustrated the benefits of the method for generating test data both theoretically and experimentally. For each fragment, the method decreased the complexity of large program while structural testing, increased the efficiency of the test data generation. Lun and Chi[15] presented a testing technique of software architecture models was described and the three testing criteria based on architecture interface connectivity graph (ICG) were proposed, and then generated testing coverage path of the ICG according to testing criteria and algorithms, Author verifed the testing technology in the TRMCS example and realized software architecture testing. Finally, the strict proof was given that the numbers of path generation between the three testing criteria, it provided a theoretical basis for applying the criteria. Zhonglin and Lingxia[16] combined the baseline method with the dependence relationship analysis, can avoid selecting infeasible paths from the control flow graph. Example had proved that it is effective.

III. PRIORTIZATION APPROACHES

In this paper, we have defined an analysis on test case prioritization. As we know the reliability of a software depends on the effectiveness of software testing. To design a reliable software system different type of testing is performed. Each kind of testing further defined some test cases. These test cases are associated with the system, modules or the code blocks. In this paper, a metric based prioritization approach is been discussed This kind of prioritization is used to identify most crucial faults in a software system. It means the test cases that represents the crucial faults will be removed first. If no such fault can be removed we cannot continue with the system. The method is quite simple to represent. It is the customize approach in which system will be analyzed respective to the developer point of view. The system will be analyzed in terms of some matrices.

Table 1: Metrics Classification

Categories	Matrices
System View	M1 Size of Function M2 Complexity of Function M3 Ratio of Newly Developed M4 Ratio of Reuse M5 Quality of Reuse
User View	M6 Frequency of Use M7 Complexity of Use Scenario M8 Impact of Function
Developer View	M9 Developer Skill M10 Experience of Similar Objects

The complete test cases represents the black box testing approach. According to these categories and subcategories some weightage is assigned to each kind of matrics. Let some score assignment given here is shown in table 2.

Table 2 : Metric Scores

Matric	Score
M1	1
M2	2
M3	3
M4	4
M5	5
M6	1
M7	2
M8	3
M9	1
M10	2

When we analyze a particular test case of the software module, it can be taken under one of more of these matrices. In such case the sum of these matrices score collectively help to find the test case. Criticality. The borders are set by the test-developers before the weighing starts. For example, if the score is between 4 and 6, the function has a medium priority. If it is less than 4 then it is of low priority and if it is more than 6 the test case is critical and having the highest priority. The evaluation of different prioritization is shown in table 3.

Table 3 : Priority Assignment

Module Id	Function	Matrices	Score	Priority
1	Evaluation of Graphical Window	M6, M7, M9	4	Low priority
2.	Database Backup	M9, M10, M8,M5	11	High Priority
3.	Data Recovery	M9, M10, M8,M5, M4	15	High Prioirty

IV. CONCLUSIONS

In this paper, a basic path testing approach is been discussed as the white box testing approach to analyze and estimate the software product. Once the path is generated, the next work is about to analyze the software system under different prioritization approach. In this work, a metric based analysis is been performed to prioritize the software modules so that relative action can be performed.

REFERENCES

[1] Roger S.Pressman “Software engineering-A Practitioner’s approach” sixth edition.

[2] Pan,J. “Software Testing”, Carnegie Mellon University.

[3] Kaner, Cem; Falk, Jack and Nguyen, Hung Quoc “Testing Computer Software”, 2nd Ed.. New York, et al: John Wiley and Sons, Inc.. pp. 480 pages. ISBN 0-471-35846-0Tran, Eushiuan,1999.

[4] Koopman, P."Verification/ Validation/ Certification". Topics in Dependable Embedded Systems. USA: Carnegie Mellon University.

[5] Hetzel, William C., “The Complete Guide to Software Testing”, 2nd ed. Publication info: Wellesley, Mass. : QED Information Sciences, 1988. ISBN: 0894352423.

[6] William,L., and Heckman,S.,“Software Engineering”,2008.

[7] Beizer, Boris. “Software Testing Techniques”. New York, NY: van Nostrand Reinhold,ISBN 0-442-20672-0,1990.

[8] Dandoti,V., “White Box Testing: An Overview”.

[9] S. Dasgupta, C.H. Papadimitriou, and U.V. Vazirani, “Algorithms”, p 173, available at http://www.cs.berkeley.edu/~vazirani/algorithms.htm.

[10] He,Z., Lv,T., Li,H., and Li,X., “Graph Partition Based Path Selection For Testing of Small Delay Defects”, 15th Asia and South Pacific Design Automation Conference(ASP-DAC),978-1-4244-5767-0/10@2010IEEE, pp.499-504, 2010.

[11] Jiang,B., Mu,Y., and Zhang,Z., “Research Of Optimization Algorithm For Path-Based Regression Testing Suit”, Second International Workshop on Education Technology and Computer Science,978-0-7695-3987-4/10© 2010 IEEE,pp.303-306,2010.

[12] He,Z., Lv,T., Li,H., and Li,X., “Fast Path Selection For Testing Of Small Delay Defects Considering Path Correlations”, 28th IEEE VLSI Test Symposium, 978-1-4244-6650-4/10©2010 IEEE,pp.3-8,2010.

[13] Zheng,Y., Mu,Y., and Zhang,Z., “ Research On The Static Function Call Path Generating Automatically”, The 2nd IEEE International Conference on Information Management and Engineering(ICIME),978-1-4244-5265-1/10©2010 IEEE, pp.405-409,2010.

[14] Wang,L., “A Program Segmentation Method For Testing Data Generating Based On Path Coverage”, IEEE International Conference on Software Engineering and Service Sciences(ICSESS), 978-1-4244-6055-7/10©2010 IEEE, pp.565-568,2010.

[15] Lun,L., and Chi,X., “Path Numbers Analysis Of Relationships On Software Architecture Testing Criteria”, 3rd International Conference on Advanced Computer Theory and Engineering(1CACTE),97S-1-4244-6542-2© 2010 IEEE,pp.118-122,2010.