A User View Specification Approach for Test Case Prioritization

Ritu[1],Tulika Das[2]

Student, PDM college of engineering and technology[1]
Bahadurgarh, Haryana

Assistent Professor, PDM college of engineering and technolog[1]
Bahadurgarh, Haryana

***Abstract*— Black Box Testing actually gives the outer view of a software system analysis based on the results and metrics. One of such metric that represents the module reliability and responsibility is its significance analysis. This significance analysis can be performed under different aspects specified by different stakeholders of projects. These stakeholders include end user, developer, tester etc. The test case significance or criticality is different for different stakeholders. In this work, an aggregative approach is defined to assign the test cases based on significance analysis. The paper has defined a metric based representation to assign the prioritization to test cases so that effective test sequence will be identified over the project.**

***Index Terms*— Black Box Testing, Significance Analysis, Module Criticality, Reliability.**

I. INTRODUCTION

Software Testing is considered as the effective derivation to the modular estimation of software system so that the effective software analysis will be done. The testing is having its significance over the software system to increase the software reliability under various analysis and problem specifications. This approach is able to identify the software bugs and to identify the software modules based on the behavior, integration and significance. This kind of software system can be tested under some organized approach. In software development, the testing is considered as the integrated phase defined with each stage of software system. It is integrated respective to the module as well as the stakeholder. It is not only able to identify the software bugs but also to helpful to reduce the complexities over the software system. The reliability of a software system depends on effective testing. This testing can be performed either at code level or it can be defined as the output driven. The black box testing is the approach in which the tests are performed based on the result analysis specific to the modules. During this, the run time bugs and behavior of software system and modules are identified[1][2][3].

).

To defined an effective and reliable testing, it is required to process the testing in a scientific way. According to this, the testing model is able to provide the desired output based on the effort analysis so that the reliable and equalize software development over the system will be obtained. The basic model of software testing adapted with each stage of software development is defined here in figure 1.

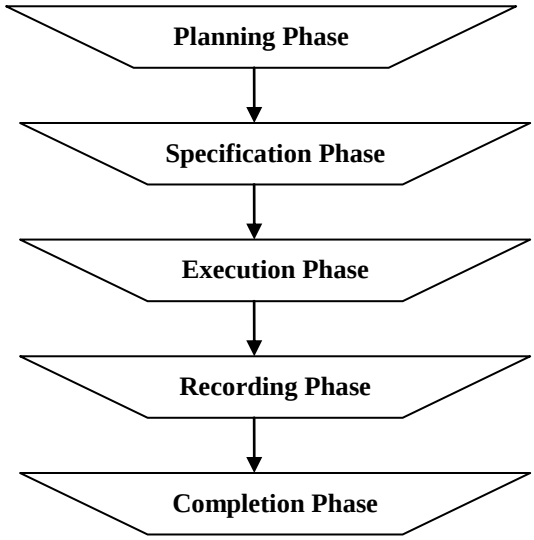

Figure 1: Black Box Testing Model

As the figure shows, the earlier stage of work is to analyze the software system under different kind of criticalities. To perform the effective and reliable system, it is required to design a software test plan at earlier phase. This plan will be able to identify the possible test cases and its integration with the software modules. During this phase, the specification of the test cases under different parameters. The parameters include the specifications of constraints respective to project, environment, stakeholders etc. After specification of these parameters and test cases, the next work is to apply these test cases over the project modules. The execution includes is performed with different test cases and with different data samples. Along with this execution phase, the recording phase is performed. During this phase, all the values obtained from the execution phase are documented. This documentation includes the specification of positive and negative test cases.

At the final stage, the testing is by representing the concluding results obtained from the test process[4][5][6].

A) Path Testing

One of the major aspects of software testing is to perform the internal analysis so that the effective test path over the software path will be performed. These test paths are analyzed under the module connectivity analysis. The path testing is also effective to analyze software system quality, module interactivity analysis and system component integration analysis. This kind of analysis is also defined with the specification semantic design of the software system. It is able to identify majority of bugs over the software system and ensure the software system reliability[6][7].

Black Box Testing includes the effective modular examination of software system under different values. This specification includes the conditional analysis and loop based analysis performed to generate the possible paths. The selection of a test case to the path can be obtained under specification of certain constraints. This kind of functional design specification so that effective control interaction can be identified and measured. This specification is also done under test case derivation. This kind of analysis is here defined under certain properties given here under :

- ❖ Generate the possible independent paths over the software module flow so that all the necessary test cases will be processed.
- ❖ The metrics will be defined for selection of specific test case.
- ❖ No loop should occur during the generation of test path.
- ❖ Overall cost of path generation should be minimized.

The module level testing is here adopted to perform the overall system testing. This includes the limit specification under integration of various situations and behaviors so that effective system testing will be obtained.

The presented work is here defined to analyze the software modules and module criticality respective to different stakeholders. Based on this user view analysis, the priority to the software test cases will be assigned. In this section, an exploration to the testing architecture is defined. The section has defined the significance and properties of black box testing. Section also include the characteristics of path testing. In section II, the work defined by earlier researchers is discussed. In section III, the proposed prioritization method is explained. In section IV, the results obtained from the work are explored.

I. LITERATURE REVIEW

Lot of work is already defined by various researchers on test case prioritization. Some of the work defined by earlier researchers on test case prioritization is discussed in this section. He *et al*.[10] defined a work on path selection based on graph partitioning approach. Author has defined critical path estimation and graph generation respective the critical path estimation and constraint specification. This specification is here been estimation under graph partitioning approach. The topology specific partitioning is performed by the author and later on simulation is performed via some tool so that the sub graphs over the software system will be obtained. This kind of estimation is done under the specification of union and categsion propduct so that the test case filteration will be done and the effective test case selection will be obtained. Another work on test case selection was performed by Jiang *et al*.[11]**.** Author has defined the test case minimization and pruning approach based on the path coverage analysis so that the effective static path generation and the point based minimum regression test generation will be performed. This kind of estimation can be performed based on the minimum set specification and use case generation. Author defined the regression testing based on the defined criteria as well as defined the cost effectivess phenomenon for effective test path generation. Author also presented the work respective to different black box testing approaches so that the significance of the test generation will be improved and the failure rate over the software system will be improved. He *et al*.[12] has presented a work on path selection under space interaction analysis and intersection problem specification approach. Author has defined a greedy approach to generate the heuristic paths so that the delay analysis over these paths will be done and the effective path will be obtained over the software system. Author defined a statistical approach for path generation so that high probability over the software system will be obtained. Author analyzed the probability under different vectors so that the conventional path estimation. Author defined the failure rate estimation , conventional path criterias and various information representation vector so that the probability of the estimation process will be improved.

Zheng *et al*.[13] has defined the software test programs based on the functional call analysis. Author has generated the graph to perform the path testing and later on flow analysis and the interaction analysis is performed between the module to generate the effective test path. Author defined the algorithmic approach for path generation and to obtain the accurate and efficient test paths. Author defined the path coverage testing based on the stability of the software system. Author has explored the involvement of the modules under interactivity vector so that better exploration of the probability vector for the software projects will be done. Wang[14] defined a work on program test plan so that the project will be divided in smaller modules and the cyclomatic complexity will be analyzed so that effective module relationship and test relationship will be obtained. This all can be done by dividing the software project in smaller fragments and the split point analysis can be performed for test data generation so that effective structural testing and path testing will be obtained from the system. Lun and Chi[15] presented a work on reliable testing under different criterias. These criteria includes the connectivity analysis so that the architectural specification to the software system will be improved.

III PRIORTIZATION APPROACHES

In this paper, an exploration to the software testing and test case prioritization is defined. To assign the priority to test cases there are number of methods can be adopted. One of such prioritization approach is defined in this section. The work is here defined based on the module level analysis respective to different stakeholders involved in the software projects. These stakeholders include end user, developer and the tester. According to this approach each stakeholder having the different view for the particular test case based on the test case criticality, significance etc. This test case estimation can be here obtained under specification over various associated vectors. This specification is here defined under different metrics and each metric is defined along with various scores. Later on specification of these all modules is done so that the effective estimation of test cases is done. This specification includes the identification of reliable software system under specification of user type. This specification includes the analysis based on the metric type and based on it, priority to the software test case is assigned.

This presented prioritization approach also includes the test case criticality and the test fault analysis in an integrated way because the actual derivation of the score is under these vectors. But each module itself will be analyzed under all criterias by all the associated stakeholders. To represent the work, a sample set of test cases is taken. Here table 1 is showing the list of test cases.

Table 1 : Test Cases

Test Case Id	Name	Type
T1	Module Size	System Based
T2	Module Complexity	System Based
T3	Reusability	System Based
T4	Component Specification	System Based
T5	Quality	System Based
T6	Skill	Developer
T7	Experience	Developer
T8	Graphical Interface	User Specific
T9	Usage	User Specific
T10	Failure/Fault	User Specific

Once the test cases are categorized, the next work is to assign the score by each stakeholder. Each stakeholder assigns the score between 1 and 5. This scoring to the test cases is shown in table 2

Table 2 : Metric Score Specification

Test Case	User	Developer	Tester
T1	1	5	5
T2	3	4	5
T3	1	5	1
T4	1	5	1
T5	3	3	5
T6	1	5	3
T7	2	5	4
T8	5	3	2
T9	5	1	1
T10	5	5	5

Once the scores will be obtained under different criterias, the identification of the overall cost to the software test cases will be done. This score identification will be based on the ratio to the maximum scoring that can be obtained. Overall score value will lie between 0 and 1. Higher the score value more critical the particular test case will be. Here the scoring obtained from this prioritization method is shown in table 3.

Table 3 : Score Table

Test Case	Aggregative Score	Max Score	Actual Score
T1	11	15	0.733333
T2	12	16	0.75
T3	7	15	0.466667
T4	7	15	0.466667
T5	11	15	0.733333
T6	9	15	0.6
T7	11	15	0.733333
T8	10	15	0.666667
T9	7	15	0.466667
T10	15	15	1

As we can see, some of the test vectors are criticality valuable for the user but they are at lesser priority for the developer and the tester. In same way, the internal analysis metrics are lesser important for user. Here fault is identified on the metric that is equally important for all kind of stakeholders. Once the scores will be obtained, the next work is to assign the priority. To assign the priority, the threshold range is defined. Here table 4 is showing the threshold range based on scores

Table 4: Priority Score

Score Range	Priority
<.25	1
.25 to .5	2
.5 to .75	3
.75	4

Based on this threshold ranging the priorities to the test cases can be assigned. Here table 5 is showing the final prioritization,

Table 5 : Priority Assignment

Test Case	Priority
T1	3
T2	3
T3	2
T4	2
T5	3
T6	3
T7	3
T8	3
T9	2
T10	4

As we can see, most of the software test cases are defined with priority 2 or 3. The paper has defined an effective and reliable approach for prioritization.

IV. CONCLUSION

In this paper, a metric based prioritization approach is defined. The metrics here considered based on the views of different kind of stakeholders to the test cases. Based on this view, the scores are assigned to the test cases and overall test case prioritization can be assigned.

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.

