INVESTIGATION OF SOFTWARE CONSISTENCY WITH TAXING MOMENT AND TOUGH EXPOSURE

R. Nithya [1]

ABSTRACT

Software reliability testing comprises analysis of software's capability to do tasks, specified environmental circumstances, intended for a specific time interim and reliability testing aids to uncover many problems in design and functional aspects of software. This paper considers mainly two aspects, i.e. testing time plus testing coverage. Testing time evaluates the testing interim requires to test specific module. And testing coverage encompasses the measure of tests exercised by the test batch or set. By considering these two aspects the reliability of software with specific circumstances is assessed. The Non-Homogeneous Poisson Process (NHPP) dependent software reliability growth model to enhance the efficiency of this project to increase the accuracy is also considered.

Keywords : Testing time, testing coverage, Poisson Process, Software reliability, test set.

I. INTRODUCTION

Reliable Software can do its intended tasks more easily and effectively[14]. Software are becoming the backbone of the many systems as well as organizations. Sometimes, software have to perform their tasks with precision and these complex software are somewhat difficult to design and test. So, there is a need to design software with less complexity and higher reliability. Therefore, the complexity of the code must be cut, while designing to check it easily. And it is also necessary to carry out reliability testing to verify the effectiveness of the software[9]. But, practically, it is very hard to execute reliability testing adequately [1]. This problem is overcome by considering the testing time and testing coverage. Steven J. Zeil and Brian Mitchell [2] put forth a reliability model merging envoy and focus based on test technique.

Nevertheless, similar to the majority of software reliability growth models, models that rely upon hastened test technique have been designed with the presumption that the reliability growth process depends only upon the testing-period like the necessary reliability growth factor or aspect. Test coverage that has significant interior software information, can be called as significant factors which are associated with the software reliability growth process [3], plus a measure demonstrating that test cases are developed efficientlyfor intended program paths. For the incorporation of result of test coverage upon reliability, Xia Cai and Michael R. Lyu [4] presented a narrative technique for the integration of instance plus test coverage dimensions collectively for the prediction of the reliability by integrating them into particular arithmetical expression. Goel-okumoto's reliability model is considered as the pioneering attempt in the development of software reliability growth model.

[1]Assistant Professor, Department of CS, CA & IT, KAHE, Coimbatore - Tamilnadu

SRGMs are designed with different circumstances under the particular assumptions. The (ENHPP) model distinguishes itself from preceding models in that it combines unambiguously the instance conflicting test scope ability in its diagnostic designation, accommodates spoiled blameworthiness detection within the testing stage,and test scope throughout the testing and working stages[8][10]. By means of SRGM, software developers can simply compute the software reliabilityand design and software reliability growth graph(or chart)[11][13]. Hoang Pham et al. proposed N-version programming SRGM. It deals with dynamic reliability, which cannot be attained by other SRGM[12].

Several scientists put their focus on the relation between test time and test coverage and generate the mathematical expression to evaluate the reliability reason of the software. Some quality factors, which can be alternatively known as metric to assess the quality of code, can also be considered. These factors can also consider the object-oriented properties.

Superiority factors focus on the quality characteristics of the software. These characteristics are utilized for tactical reasons. Itis also useful to cut the progress schedule by means of preparing the amendments required to evade delays and alleviate prospective difficulties and risks. In addition,it is also used to appraise product superiority. Therefore we include the analysis of superiority characteristics through the evaluation of object-based quality metrics.

This article presents a new approach towards reliability testing along with testing time and test coverage. Section two represents the background and motivation; section three covers the software development process; section four encompasses the concepts within the testing coverage and testing time respectively; section six comprises the mathematical model and section seven shows the architecture of the system.

II. BACKGROUND AND MOTIVATION

There is no field where software have not reached. Software may be at the application level or system level. Software which require functionality with high precision need to be designed carefully with the reduction in code complexity. So, developing anew approach towards reliability is the main motivation.

III. SOFTWARE DEVELOPMENT PROCESS

Process is a set of actions and events that are carried out to develop asoftware. From the perspective of software engineering, the process is nothing but adjustable view which facilitates people for performing the task to select a proper bunch of events and actions. There are various approaches towards the development of software. This processes are not performed in a single iteration.If the customer is not satisfied with the developed software product, it can be an iterative process. According to the feedback of the customer, the software product is revised to fulfill his requirement of accuracy.The objective is to release the software within a time frame, along with adequate superiority, to the satisfaction of the customers. Standard process for software development comprises subsequent tasks[5]:

A. Communication

Before starting any technical mechanism, it is very important to converse and collaborate with the client to recognize the purpose of the product to be developed and collect the needs which determine the defining features and functionality of the software.

B. Planning

Like a map is utilized to simplify any complex journey, proper planning is used to define the software engineering efforts. The planning includes tasks such as managing resources, scheduling etc.

C. Modeling

In order to develop the software product, the basic architecture needs to designed. For modeling, the problem that requires a solution must be thoroughly defined before designing the architecture.

D. Construction

Construction process encompasses two activities collectively. These two activities are coding and testing. Coding is the activity that translates the design into actual program for the sake of execution. Testing is the procedure that is adopted to make the software error or bug free.

E. Deployment

The software, after its development is transported to the client, who will estimate the product and give his feedback based on estimation.

IV. TESTING COVERAGE

Testing Coverage presumes to execute a tremendously considerable task in visualizing the software reliability. TC aids software designers for the assessment of the superiority of examined software project as well as helps them decide the measure of extra efforts needed for the progression of software reliability. The classes for testing coverage are given below[5]:

1. Statement coverage: This class is expressed by estimating the fraction of statements covered by the bunch of planned test cases.
2. Decision/condition coverage: This coverage is estimated by evaluating the fraction of decision branches covered by the bunch of planned test cases.
3. Path coverage: This coverage is intended to calculate the fraction of execution paths or conducts throughout a program covered by the bunch of planned test cases.
4. Function coverage: Total function count exercised by test cases is nothing but the functional coverage.

The coverage is a computation unit use for the description of the measure towards which the source code within a program is examined by a scrupulous test suite. The program which has high code coverage is tested much scrupulously and it has less possibility of comprising bugs than a program with low code coverage.

V. TESTING TIME

Testing time comprises the time required to execute the test process. Testing time depends on various factors such as testing technique, size of code and, sometimes, the efficiency of testing tool etc. Total testing time includes summation of time count required to test all the modules within the software.

VI. MATHEMATICAL MODEL

Mathematical model is an algebraic expression that shows the quantitative stimulation of software analysis.

Block coverage: overall measure of blocks that has been executed by test cases

Branch coverage: overall measure of branches that has been executed by test cases

Notations :

m(t) = Expected number of Faults detected at time t

λ (t) = Failure Intensity of the software at time t

c(t) = Coverage function over a time interval

α = Expected number of faults that may be

detected given infinite testing time

Basic Non-Homogeneous Poisson Process (NHPP):

Test Case1:

Where

t1 is first phase end of testing,

α is expected measure of defects discovered with the provision of given infinite quantity of testing time interim.

C1 is coverage function over interval time ⩽ ? t < t1

Test Case 2 :

Where

t2 is second phase end of testing

C2 is coverage function over interval time t1 ⩽ t < t2

Test Case3:

Where

t3 is third phase end of testing

C3 is coverage function over interval time t2 ⩽ t < t3

Proposed ENHPP Model:

Where

ti is end of ith phase of testing

Ci is coverage function over interval time ti-1 ? t <ti

VII. PROPOSED SYSTEM ARCHITECTURE

Figure 1 System Architecture

Figure 1 shows the proposed architecture of system. System involves three main components as follows:

1. Input modules
2. Software reliability analysis based on testing time and coverage
3. Evaluated inference

The first part is the response modules. The project is based on the white box testing strategy, as it is associated with the coding structure.So, the modules within the software is given as an input. And the coding part within these modules is analyzed.

Analysis of consulted modules is carried out in the second part, i.e. software reliability analysis. This part comprises the evaluation of reliability by means of testing time and coverage estimation. Many factors are calculated in this component for the sake of estimating reliability.

After analysis the evaluated inference is generated. This inference expresses the reliability from the context of internal structure.

VIII. CONTRIBUTION WORK

Contribution of this work is towards the improvement of code superiority and reliability. And the task is carried out by means of superiority factors. Superiority factors include various properties associated with codes. These factors are analysed to generate the results and, by means of the combination of this result with reliability analyzer,good results can be definitely generated.

IX. CONCLUSION

This paper presents the technique to analyze reliability with the combined use of testing time analyzer, test coverage, and reliability analyzer. The software from

its internal structure is attempted to be analyzed though this technique, i.e. coding structure. And if the code structure is improved, the reliability automatically increases too. This paper will definitely becomes a useful factor in testing circumstances, so that programmers can show the complexity within the code and try to make it simple and reliable, so that software is very reliable before dispatching it for testing.

References

1. Shuanqi Wang, Yumei Wu, Minyan Lu, Haifeng Li, "Software Reliability Accelerated Testing Method Based on Test Coverage", IEEE 2011.
2. Mitchell, B. and S.J. Zeil, "A Reliability Model Combining Representative and Directed Testing," in Proceedings of ICSE-18, 1996.
3. Inoue, S. and S. Yamada, "Two-Dimensional Software Reliability Assessment with Testing-Coverage," in the Second International Conference on Secure System Integration and Reliability Improvement. 2008.
4. Cai, X. and M.R. Lyu, "Software Reliability Modeling with Test Coverage: Experimentation and Measurement with A Fault-Tolerant Software Project," in 18th IEEE International Symposium on Software Reliability Engineering. 2007.
5. Roger S. Pressman, "Software Engineering: A Practitioner's Approach", McGRAW Hill international publication, seventh edition, 2004.
6. Inoue S, Yamada S "Testing Coverage Dependent Software Reliability Growth Modeling" International Journal of Reliability, Quality and Safety Engineering, Vol. 11, No. 4, 303312
7. Jing Zhao Hong-Wei Liu Gang Cui Xiao-Zong Yang, "A Software Reliability Growth Model from Testing to Operation", Proceedings of the 21st IEEE International Conference on Software Maintenance, 2005
8. Mr. Sandeep P. Chavan, Dr. S. H. Patil, Mr. Amol K. Kadam, "DEVELOPING SOFTWARE ANALYZERS TOOL USING SOFTWARE RELIABILITY GROWTH MODEL" International Journal of Computer Engineering and Technology (IJCET),ISSN 0976 - 6367(Print), ISSN 0976 - 6375(Online), © IAEME, pp 448-453, Volume 4, Issue 2.
9. Ms. PoorvaSabnis, Mr. Amol Kadam, Dr.S.D.Joshi, "Analysis and Design of Software Reliability growth Model Using Bug Cycle and Duplicate Detection", International Journal of Emerging Trends & Technology in Computer Science (IJETTCS) September - October 2013 ISSN 2278-6856 , Volume 2, Issue 5
10. Mr. Sandeep P. Chavan, Dr. S. H. Patil, "SRGM Analyzers Tool of SDLC to Ensure Software reliability and quality", International Journal of Computer Science and Technology (IJCST),ISSN 2229 - 4333 (Print), ISSN 0976-8491 (Online),COSMIC JOURNALS, pp 438-441, Volume 4, Issue 2.
11. P. K. Kapur, H. Pham, Fellow, IEEE, Sameer Anand, and Kalpana Yadav, "A Unified Approach for Developing Software Reliability Growth Models in the Presence of Imperfect Debugging and Error Generation", IEEE TRANSACTIONS ON RELIABILITY, VOL. 60, NO. 1, MARCH 2011

12. Xiaolin Teng Hoang Pham, "A Software-Reliability Growth Model for N-Version Programming Systems", IEEE TRANSACTIONS ON RELIABILITY, VOL. 51, NO. 3, SEPTEMBER 2002

13. Alan wood, "Software reliability growth models", Technical report September 1996.

14. Mei-Hwa Chen, Michael R. Lyu W. Eric Wong," Effect of Code Coverage on Software Reliability Measurement", IEEE TRANSACTIONS ON RELIABILITY, VOL. 50, NO. 2, JUNE 2001