ISSN: 2321-7782 (Online)
Volume 3, Issue 5, May 2015
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

Analysis of Software Reliability using Testing Time and Testing Coverage

Vaibhav E. Pawar[1]
Department of Computer Engineering
Bharati Vidyapeeth Deemed University College of Engineering,
Pune – India

Amol K. Kadam[2]
Department of Computer Engineering
Bharati Vidyapeeth Deemed University College of Engineering,
Pune – India

Dr. S. D. Joshi[3]
Department of Computer Engineering
Bharati Vidyapeeth Deemed University College of Engineering,
Pune – India

Abstract: ***Software reliability testing comprises the analyzing software's capability to perform tasks, specified environmental circumstances, intended for a specific time interim. Reliability testing aids to uncover numerous problems in design and functional aspects of software. This paper considers mainly the 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 quantity of tests exercised by the test batch or set. By considering these two aspects we try to assess the reliability of software with specific circumstances. We also consider the Non-Homogeneous Poisson Process (NHPP) dependant software reliability growth model to enhance the efficiency of this project in order to increase the accuracy.***

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

I. INTRODUCTION

Reliable Software can perform its intended tasks more easily and effectively[15]. Softwares are becoming the backbone of the many systems as well as organizations. Sometimes softwares have to perform its task with the accurate precision and these complex softwares are somewhat difficult to design and test so have to design the software with less complexity and higher reliability. Therefore we have to reduce the complexity of code while designing in order to examine 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 plus directed based 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 dependant only upon the testing-period like the necessary reliability growth factor or aspect. Test coverage that contains significant interior software information, be called as significant factors being associated to the software reliability growth process [3], plus a measure demonstrating that test cases are developed efficiently intended for program paths. For the incorporation of consequence 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 means of 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. SRGMs are designed with different circumstances under the particular assumptions. The (ENHPP) model distinguishes from precedent models in that it combines unambiguously the instance conflicting testscope capability in its diagnostic designation, plus accommodates spoiled blameworthiness detection within the testing stage plus test scope throughout the testing plus operational stages[8][10]. By means of SRGM, software developers can simply compute the software reliability, plus design

software reliability growth graph(or chart)[11][13]. Hoang Pham et al proposed N-version programming SRGM. It deals with the dynamic reliability, which cannot be attained by other SRGM[12].

Several scientists put focus on relation of test time and test coverage and generate the mathematical expression in order to evaluate the reliability factor of the software. We also consider some quality factors which can be alternatively known as metric to assess the quality of code. These factors can also consider the object-oriented properties.

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

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

II. Background and Motivation

There is no field where the software has not reached. Softwares may be application level or system level. Software which requires functionality with high precision needed to design carefully with the reduction in code complexity. So, developing the new approach towards the reliability is the main motivation.

III. Software Development Process

Process is the set of actions and events that carried out in order to develop software. From the perspective of software engineering the process is nothing but adjustable view which facilitates the people performing the task to select the proper bunch of events plus actions. There are various approaches towards the development of software. This processes are not performed within single iteration if the customer will not satisfy with the developed software product then it can be an iterative process. According to the feedback of the customer the software product is revised to fulfil the requirements of user and to achieve the accuracy of the software. Objective is to release software in an appropriate time along with adequate superiority so as to satisfy the customers. Standard process for software development comprises subsequent tasks[5]:

A. Communication

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

B. Planning

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

C. Modeling

In order to develop the software product we have to design the basic architecture. For modeling we have to realize the problem definition thoroughly and accordingly design the architecture

D. Construction

Construction process encompasses the 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 utilizes to make software error or bug free.

E. Deployment

The software after the development is transported to the client or customer who will going to estimate the product plus gives 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 to determine the quantity of additional efforts needed for the progression of software reliability. Consequent are the classes for testing coverage [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 utilize for the description of the measure towards which the source code within a program is examined by means of a scrupulous test suite. The program which has high code coverage is tested much scrupulously and it has the less possibility of comprising the bugs than a program with low code coverage.

V. TESTING TIME

Testing time comprises the time required to execute test process. Testing time depends on the 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 algebraic expression that shows the quantitative stimulation of software analysis.

Block coverage: overall quantity of blocks that have been executed by test cases

Branch coverage: overall quantity of branches that have been executed by test cases

$$Block\ coverage = \frac{Number\ of\ the\ blocks\ covered\ by\ the\ case}{total\ no\ of\ the\ blocks}$$

Notations:

m(t) = Expected no of Faults detected at time t

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

c(t) = Coverage function over a time interval t

α = Expected no. of faults that may be detected given infinite testing time

Basic Non-Homogeneous Poisson Process (NHPP):

$$\frac{dm(t)}{d(t)} = \lambda(t)$$

Test Case1:

$$\frac{dm_1(t)}{dt} = \propto \int_0^{t_1} c_1\, dt$$

Where

t_1 is initial phase end of testing,

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

C_1 is coverage function over interval time $0 \leq t < t_1$

Test Case 2 :

$$\frac{dm_2(t)}{dt} = (\propto -(m_1(t)) \int_{t_1}^{t_2} c_2\, dt$$

Where

t_2 is second phase end of testing

C_2 is coverage function over interval time $t_1 \leq t < t_2$

Test Case3:

$$\frac{dm_3(t)}{dt} = (\propto -(m_1(t) + m_2(t)) \int_{t_2}^{t_3} c_3\, dt$$

Where

t_3 is third phase end of testing

C_3 is coverage function over interval time $t_2 \leq t < t_3$

Proposed ENHPP Model:

$$\frac{dm_i(t)}{dt} = (\propto -\sum_{k=1}^{i} (m_k(t)) \int_{t_{i-1}}^{t_i} c_i\, dt$$

Where

t_i is end of i^{th} phase of testing

C_i is coverage function over interval time $t_{i-1} \leq t < t_i$

VII. Proposed System Architecture

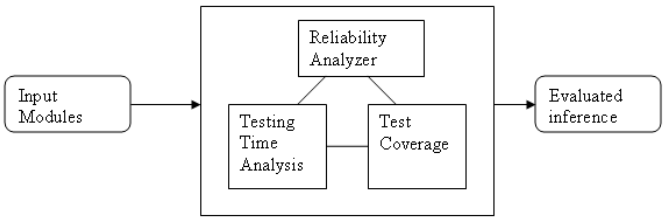

Figure 1 System Architecture

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 component is the input modules. Our project based on the white box testing strategy as it associated with the coding structure so we will give the modules within the software as an input. And the coding part within these modules is analyzed.

Analysis of inputted modules is carried out in the second component i.e. software reliability analysis. This component comprises the evaluation of reliability by means of testing time and coverage estimation. Many factors are calculated in this component for the sake of the estimating the 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 the code superiority and reliability. And we accomplish this task by means of superiority factors. Superiority factors include the various properties associated with code. We analyse these factors and generate the results and by means of the combination of this result with reliability analyser we definitely generate the good results.

IX. Conclusion

This paper presents the technique to analyze reliability with the combination of testing time analyzer, test coverage, and reliability analyzer. Through this technique we have tried to analyze the software from its internal structure i.e. coding structure. And if the code structure is improved the reliability automatically increases. This paper definitely becomes a useful factor in testing circumstances so that programmer can identify the complexity within the code and try to make it simple and reliable so that software is very reliable before dispatching it for the 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. Poorva Sabnis, 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

15. 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

