International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Using Functional Point Analysis and Test Point Analysis Reducing Maintenance Cost of Software

Samadhan E. Kadam[1], Amol K. Kadam[2], Dr. S. D. Joshi[3]

Department of Computer Engineering, Bharati Vidyapeeth Deemed University College of Engineering Pune.

Abstract: ***Maintaining reliability is the difficult task while developing software. Software reliability is defined as the possibility of failure free functioning for a particular period of time within precise condition. Software reliability growth models are utilising for the estimation of reliability by means of statistical expression. It is also useful for interpretation of software failure as an arbitrary procedure. Testing is one of the important phases within software development lifecycle. The Testing technique is the organized procedure for the detection of defects or bugs in the software. Nevertheless the only testing is purely unable to eliminate all the bugs or defect within the software due to factors such as complexity of software, inefficiency of understanding the software functionality. And sometimes there is a possibility of error generation and imperfect debugging. So, we have to apply the SRGM within testing phase to measure the reliability quantitatively. This paper presents novel approach towards the software reliability growth model.***

Keywords: Software reliability, Software Reliability Growth Model

1. Introduction

Software industry is gradually progresses on large scale day-by-day. Computer technology has reached almost in every field including banking sector, educational institute, government organization or institute, satellite launching system etc. With this development in software industry there are various software development methodologies has proposed. But development of software is not enough to maintain the entire quality and reliability of software. Software quality degrades due to presence of bugs or defects in software. Application of testing technique removes the defect within software up to certain extent. But there is no testing technique which can completely make software defect free. So, we have to apply the software reliability growth model at the testing phase to make testing mechanism more effective so as to obtain the reliability. It is very important to achieve reliability to maintain quality of software. Software reliability is defined as the possibility of failure free functioning for a particular period of time within precise condition. [1]

Software reliability is the generally dynamic attribute that assesses plus envisages the operational (functioning) superiority of the software product [2]. A software reliability growth model expresses fault recognition as a mathematical implementation [3]. There are various SRGMs has been proposed in accordance with the different parameters. Various SRGMs are appropriate during specific situations so there is no SRGM which is globally accepted as a standard model. The majority of SRGMs had constructed with the assumption that faults or defects which are discovered while testing the software product are abolished quickly without introducing new faults [4].

Software reliability growth model which is proposed by Goel-Okumoto [5] consider the presumption that the defects or faults abolished immediately after fault or defect discovery. Nevertheless from the practical point of view it is not possible because fault removal mechanism takes time as it involves the series of various steps. And also there is a possibility of imperfect debugging and error generation. Sometimes the testing technique is unable to remove fault and fault remains in the software even after the application of testing technique this phenomenon is known as imperfect debugging. Insertion of novel faults or defects takes place during the fault removal process. This event is called as error generation. So, it is necessary to consider the probability of imperfect debugging while designing the SRGM.

Generally SRGMs based on the Non-Homogeneous Poisson Process(NHPP) predicts better reliability. Poisson process is the mechanism for calculating the amount of events plus time instance at which these events take place in a given time interim. NHPP calculates the amount of actions or events takes place at inconsistent rate. Goel-Okumoto's research gives a means for research on SRGM based on Non-Homogeneous Poisson Process (NHPP).

2. Background and Motivation

With the growing software industry the challenge of maintaining the quality of software also increases. Quality and reliability are the relative concepts as quality factors affects greatly on reliability of the software. So we try to achieve reliability with the application of SRGM at the testing phase.

So, achieving reliability is the main motivation behind this research work.

3. Software Development Life Cycle (SDLC) Phases

Software development is the systematic mechanism which involves the series of various steps. SDLC phases are the set of different procedures for the software development. Subsequent are the various phases within SDLC:

5.1 Feasibility Analysis

Before developing any software product we have to check the feasibility of the proposed system. Feasibility analysis includes the evaluation of various factors like intended input, desired output, and cost estimation or analysis required to

develop software product. By means of feasibility study we can decide whether to proceed with given proposed system or not.

5.1 Requirement analysis and specification

This is the important phase. Requirement analysis involves the understanding the actual need of the customer. Software product design is mainly depend on accuracy of requirements. Collection of accurate software requirement is the main aspect in this phase. After accumulating requirements together a document named as software requirement specification has prepared. SRS contains various important requirements (needs) based on which software has been developed i.e. SRS can be act as an input to the next phase (design phase).

5.1 Design

Design phase encompasses the conversion of the requirements included in the SRS into logical formation which later can be implemented in particular programming language. The logical formation obtain through design phase will be given to the next phase as an input.

5.1 Coding

Coding phase comprises the actual implementation of software design obtained previously via the design phase. This phase converts the logical formation into computer program. All programming modules are integrated together to form the software product.

5.1 Testing

Testing comprises the mechanisms or systematic procedure to remove the defects or errors with in the software developed previously. The testing practice initiates by means of a assessment map which identifies test associated actions such as generation of test case, testing standards, plus resource allotment for the sake of testing. The code is analyzed plus atlases alongside the intended design manuscript formed during the design phase.

5.1 Maintenance

The maintenance stage incorporates managing the lingering errors which might exist within the software still subsequent to the testing. It also includes the maintaining effort for the software deployment.

4. Testing Coverage

Testing Coverage supposes to perform an extremely significant task in envisaging the reliability of software. TC helps software programmers to assess the superiority of the tested software product plus decide the amount of extra efforts required to progress software reliability. Subsequent are the categories of testing coverage [6]:

1) Statement coverage:- This coverage is estimated by evaluating 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 calculated as the fraction of execution paths or conducts throughout a program covered by the bunch of planned test cases.
4) Function coverage:- Total numbers of functions exercised by test cases is nothing but the functional coverage.

5. Mathematical Model

5.1 Function Point Analysis

Rate Amendment Factor (RAF) is mainly rely on the Basic System Parameters (BSP) which is the composition of various 14 system characteristics. These characteristics includes factors such as data interchange system, complex mechanism, ease to reuse, execution and installation ease, user friendliness, ease to operate etc. After analyzing the 14 BSP's we have to evaluate Rate Amendment Factor (RAF).

RAF = 0.65 + [(Di) / 100]. i = is from 1 to 14 representing each BSP.
Where,
Di = level of influence intended for every basic system parameter
i = ∑(all BSPs) i.e. Summation of all BSP's
Function Point is obtained as
Function Point (FP) = UAF (Unadjusted Function Point) + RAF (Rate Amendment Factor)

5.2 Test Point Analysis

Testing evaluation mainly rely on amount of code lines (ACL) within the program. Presume 100ACL (Amount of Code Lines) = 10FP (Function point) then 1000ACL=100FP
FP*3TECH (BVA,EP, Error prediction) = TC
where,
TC=Test Case
100FP*3=300TC
we can evaluate 40 TC for every day i.e. 400TC/40=10DAYS to manipulate test cases.
Test plan takes one half of the test case manipulation i.e. 5 days.
Test case execution takes one and half of the test case manipulation
i.e. 15 days
Buffer time =20 days

5.3 Test Strategy

Test Strategy comprises the various techniques and tools required for testing purpose and also involve preparation of testing document.

6. Proposed System Architecture

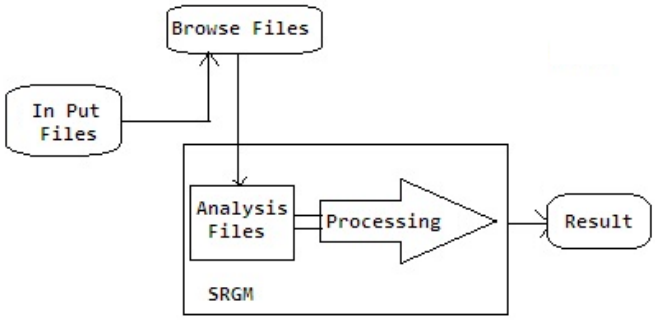

Figure 1: Architecture of Proposed System.

Figure 1 shows the diagram for the architecture of proposed system. Architecture can be depicted as subsequent way:

1) In Put Files is the actual input to the project. It contains the various code modules to be analyzed
2) Analysis on input file has performed by means of function point analysis and test point analysis along with SRGM.
3) Then finally result of the processing will shown in th form of graph.

7. Function Point Analysis And Test Point Analysis

7.1 Function Point Analysis (FPA)

FPA is utilized to quantify the dimension of the system by estimating the intricacy of system functionalities. FPA has the following components [7]:

1) External Inputs (EI): - It is a basic mechanism that encompasses the data or information crossing the border from exterior are to interior area.
2) External Outputs: - It is a basic mechanism that encompasses the resultant data or information crossing the border from exterior are to interior area.
3) External Inquiry: - It is a basic mechanism by means of together input plus output elements which leads information or data retrieval from single or more interior rational documentation (files) plus exterior interface records.
4) Internal Logical Files (ILF's): – It is a customer detectable assemblage of rationally connected data which exist in completely within the application border plus be sustained throughout external inputs.
5) External Interface Files (EIF's): – It is a customer detectable assemblage of rationally connected data which is utilized for the sake of reference only.

7.2 Test Point Analysis(TPA)

Test point analysis is the mechanism for the test evaluation. When devise estimation intended for a black box testing, three essentials are appropriate [8]:

1) Size: It is nothing but dimension of information system which has undergone the testing
2) Test strategy: Choice of system elements plus superiority properties to be examined along with test coverage.
3) Productivity: Productivity is associated to the time interim required to understand single test point, as previously decided by means of size and test strategy.

8. Conclusion

Through this work we have tried to improve the reliability of the software product from the context of internal programming structure. We have also taken the points such as function point analysis and test point analysis into consideration for the sake reduction in software maintenance cost.

References

[1] Mei-Hwa Chen, Michael R. Ly,W. Eric WongJ., "Effect of code coverage on software reliability measurement", IEEE TRANSACTIONS ON RELIABILITY, VOL. 50, NO. 2, JUNE
[2] Khurshid Ahmad Mir, "A Software Reliability Growth Model," Journal of Modern Mathematics and Statistics, Vol. 5, No. 1, pp. 13-16, 2011.
[3] Alan wood, "Software reliability growth models", Technical report September 1996.
[4] P. K. Kapur, H. Pham, Sameer Yadav, Sameer Anand, KalpanaYadav, "A Unified Approach fo Developing Software Reliability Models in the Presence of Imperfect Debugging and Error Generation ", IEEE Transactions on reliability, vol 60, pp. 331-340, March 2011.S.
[5] Amrit L Goel, Kazu Okumoto, "Time dependent error detection rate model for software reliability and other performance measures", IEEE transactions on reliability, vol R-28, pp. 206-211, 1979.
[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] http://www.softwaremetrics.com/fpafund.htm
[8] Drs. Erik P.W. M. van Veenedaal CISA, Tom Dekkers, "Test point analysis: a method for test estimation", Project Control for Software Quality, shaker publishing BV, Maastricht, The Netherlands, 1999.