SRGM Analyzers Tool of SDLC to Ensure Software Reliability and Quality

[1]Sandeep P. Chavan, [2]Dr. S. H. Patil
[1,2]Dept. of CE, Bharati Vidyapeeth Deemed University College of Engg. Pune, India

Abstract
In this paper, we have developed software analyzers tool for deriving several software reliability growth models based on Enhanced nonhomogeneous Poisson process (ENHPP) in the presence of imperfect debugging and error generation. The offered models are initially formulated for the case when there is no differentiation between failure observation and fault removal testing processes, and then continued for the case when there is a clear differentiation between failure observation and fault removal testing processes. Many software reliability growth models (SRGM) have been developed to describe software failures as a random process, and can be used to classify development status during testing. With software reliability growth, software engineers can easily measure (or forecast) the software reliability (or quality), and design software reliability growth charts. It is not easy to select the best tool for improving software quality. There are few SRGM in the literature of software engineering that differentiates between failure observation and fault removal processes. In real software development background, the number of failures checked need not be the same as the number of faults removed. Due to the elaboration of software systems, and an defective understanding of software, the testing team may not be able to discard the fault perfectly on observation of a failure, and the authentic fault may remain, resulting in a phenomenon known as defective debugging, or get replaced by another fault causing error generation. In the case of defective debugging, the error content of the software remains the same; while in the case of error generation, the error content increases as the testing progresses. Replacement of observed faults may result in the introduction of new faults.

Keywords
Software Reliability Growth Models (SRGM), ENHPP (Enhanced Non Homogeneous Poisson Process), SDLC(Software Development Life Cycle), FDR (Fault Detection Rate).

I. Introduction

Software reliability assessment is an important issue for planning release of high-quality software products to users. many Developers have proposed software reliability growth models (SRGMs) to assess software quantitatively from fault data observed in software testing phase [1-2, 4-5]. Software reliability is defined as the probability of failure free software operation for a specified period of time in a specified environment. It is used to assess the reliability of the software during testing and operational phases. Software testing cover running the software and checking for unexpected behavior in software output. The achieving test can be considered to be one, which come out with the presence of the latent faults. The process of locating the errors and designing the procedures to detect them is called the remove errors process. The chronology of failure occurrence and fault detections can be utilized to provide an estimate of the software reliability and the level of fault content. In particular, Enhanced non-homogeneous Poisson process (ENHPP) based SRGMs are quite popular due to their mathematical tractability, and there have been a number of ENHPP-based SRGMs proposed by many Developers. In spite of the diversity and elegance of many of these, no single model can be readily recommended as best to represent the challenging nature of the software testing [8, 10].

In this paper, we have developed a generalized framework for deriving several testing-time and testing coverage depends on Enhanced Non-Homogeneous Poisson Process software reliability growth model which incorporate Change point. Change point is one of the interesting phenomen on observed during software development. Inclusion of change point in software reliability growth modeling enhances the predictive accuracy of the model. Many Developers have incorporated change point in software reliability growth modeling. Firstly Zhao [6] incorporated change-point in software and hardware reliability. Huang et al. [5] used change-point in software reliability growth modeling with testing effort functions. The defective debugging with change-point has been introduced in software reliability growth modeling by Shyur [3]. Kapur et al.[7,9] introduced various testing effort functions and testing effort control with change-point in software reliability growth modeling. The multiple change-points in software reliability growth modeling for fielded software has been proposed by Kapur et al. [7-8]. A testing coverage based SRGM was proposed by Malaiya [7]. Inoue and Yamada [6] developed SRGM to describe a time-dependent behavior of a testing-coverage attainment process with the testing-skill of test-case designer. Although, the above mentioned developed models proved to be more accurate than the only time governing SRGMs, but they failed to incorporate the concurrent effect of time and Testing Coverage. Recently, Inoue and Yamada [10] proposed a two dimensional software reliability growth model that considered the simultaneous effect of time and Testing-Coverage based on the Cobb Douglas production function. The Cobb–Douglas creation function has been extensively used to characterize the relationship of an output to inputs namely capital and labor. Yamada [10] modeling framework was not directly based on using mean value functions to represent of fault removal process. They discussed software reliability assessment method by using two dimensional Weibull-types SRGM.

II. Background and Motivation

Program testability and test coverage are related concepts.the first refers to the ease with which one may test a program while the second provides a measure of how thoroughly a test has covered all potential fault site is defined very broadly to mean any structurally or functionally described program element whose integrity may require verification and validation.the factors which impact program testability and test coverage include:

- Program complexity.
- Software tools.
- Test quality.

III. Our Analysis

From our studies, many existing SRGM can be unified under a more general formulation. In fact, model unification is an penetrative investigation for the study of general models without making many assumptions. In this paper, we present Enhanced Non-

homogeneous Poisson Process (ENHPP) models, which account for imperfect debugging and error generation, and show that previously reported Enhanced Non-Homogeneous Poisson Process (ENHPP) software reliability models with imperfect debugging and error generation are special cases of the ENHPP models. The ENHPP models are capable of handling any general distribution function, and are thus an important step towards the unification of the NHPP models, which rely on specific distribution functions. From this approach, we can not only obtain existing NHPP models, but also develop new ENHPP models. The proposed models are initially formulated for the case when there is a clear differentiation between failure observation and fault removal processes (ENHPP). While other NHPP models involve the concept of imperfect debugging, most ENHPP based SRGM are also special cases of the general models. The existing and proposed models have been validated and evaluated on actual software fault removal data sets [10].

The advanced models are based upon the following basic assumptions.

1. The failure observation and error removal phenomenon is modeled using a ENHPP.
2. Software is subject to errors during execution caused by faults remaining in the software.
3. Each time a errors is observed, an immediate debugging effort takes place to find the cause of the errors to remove it.
4. The error rate is equally affected by all the errors remaining in the software.
5. When a software errors occurs, an instantaneous repair effort starts, and then either (a) the error content is reduced by one with probability , or (b) the error content remains unchanged with probability .
6. During the errors removal process, whether the fault is removed successfully or not, new errors are generated with a constant probability.

A. System Description

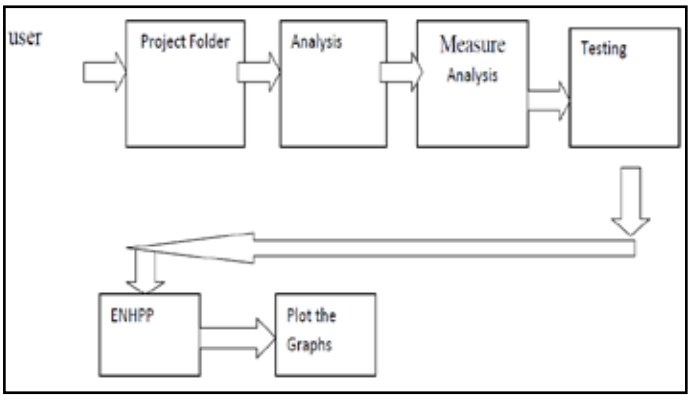

Fig. 1: ENHPP Model

A] ENHPP Model Phases

i] Phases Of Analyzer:

1. Identifying Phase

This phase consists identifying software such as TNOA,WMPC,NOCP etc. as shown below.

TNOC: It counts total of attributes for a class.

WMPC: It counts all class methods per class.

NOCP: It Count the Number of classes in a package.

MIT: It calculates the longest path from the class to the root of the inheritance tree.

CCIM: It measures the number of linearly independent paths through a program module.i.e, the amount of decision logic in a single software factor. It is calculated per class.

SIZE: It refers to the count total lines of code.

2. Evaluation Phase

This phase consists of evaluating the various software measure written as a part of "Analyzer" tool development which are provided for examination

3. Interpretation and Advising Phase

The interpretation part of the measure Analyzer takes the extracted measure values in measure Evaluation Phase from the source code and compare these values with the threshold values of the corresponding metric values. If the extracted metric values lies below the corresponding metric threshold value means that no occurrence of the issue that is being observed and extracted measure values lies above the corresponding metric threshold value means maximum occurance of the observed issue.

4. Application Phase

In this phase,software refactoring is done .Our tool " measure Analyzer" applies the Object Oriented metrics on the code base and these metric values are then interpreted. Then various refactoring techniques were used to improve the code design and along with that we also studied the impact of refactoring on the software quality through various measure .

IV. Methodologies And Techniques –ENHPP (Enhanced Non-Homogeneous Poisson Process) Model

The ENHPP model provides a unifying framework for finite failure software reliability growth models According to this model, the expected number of errors detected by time t, called the mean value function, n(t) is of the form :

$n(t) = b * c(t)$

Where b is the needed number of shortcomings in the programming (before testing/debugging starts), and c (t) is the scope capacity. the ENHPP model utilized by SREPT gives by default to reflect four sorts of inadequacy event rates for every fault. Inter flop times information acquired from the testing stage could be utilized to parameterize the ENHPP (Enhanced Non Homogeneous Poisson Process) model to acquire gauges of the inadequacy force, number of flaws remaining, dependability after discharge, and scope for the programming. When complexity metrics are available, the total number of errors in the software can be estimated using the fault density approach or the regression tree model. If the number of lines ofcode in the software is NL, the expected number of errors can be estimated as, [5] :

$F = NL * FD$

The relapse tree model is an objective arranged statistical method, which endeavors to foresee the amount of deficiencies in a programming module dependent upon the static unpredictability measurements. The structure of the ENHPP model might additionally be utilized to join the appraisal of the aggregate number of issues acquired after the testing stage dependent upon unpredictability measurements (parameter a), with the scope informative content got throughout the testing stage (c (t)). The ENHPP model can also use inter failure times from the testing phase to obtain release times (optimal time to stop testing) for the software on the basis of a specified release criteria.

Release criteria could be of the following types –

Number of remaining faults - In this case, the release time is when

a fraction of all detectable faults has been removed.
Failure intensity requirements - The criterion based on failure intensity suggests that the software should be released when the failure intensity measured at the end of the development test phase reaches a specified value f.
Reliability requirements - This criteria could be used to specify that the required conditional reliability in the operational phase is, say Rr at time t0 after product release.
Cost requirements - From a knowledge of the expected cost of removing a fault during testing, the expected cost of removing a fault during operation, and the expected cost of software testing per unit time, the total cost can be estimated. The release time is obtained by determining the time that minimizes this total cost.
Availability requirements - The release time can be estimated based on an operational availability requirement [2].

V. Model Validation

To illustrate the estimation procedure and application of the SRGM (existing as well as proposed), we have carried out the data analysis of a real software data set.
1. Test coverage is the ratio of the number of potential faults sites sensitized by the test divide by total number of potential fault sites under consideration.

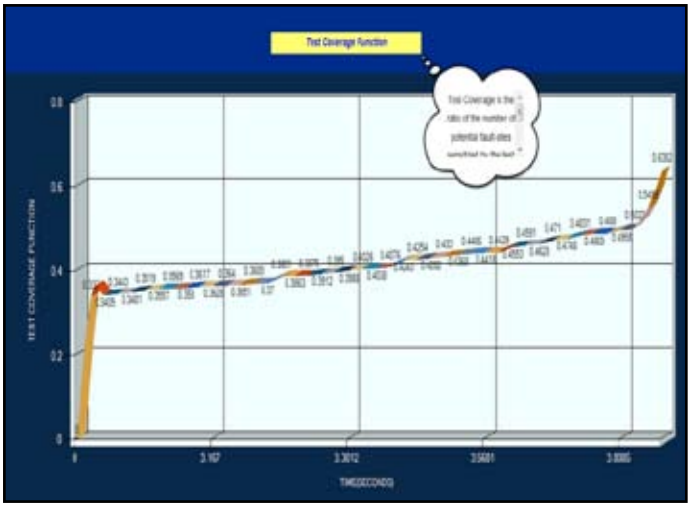

Fig. 2: Test Coverage Function

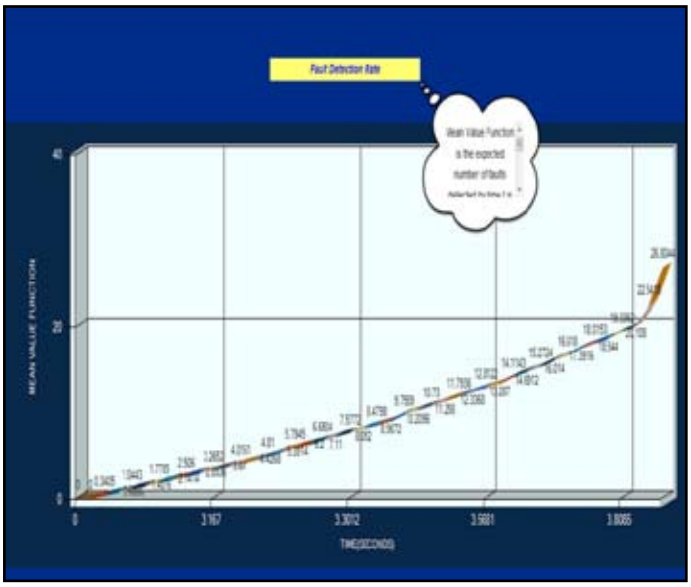

Fig. 3: Fault Detection Rate

2. Mean value function is the expected number of faults detected by time t is equal to total number of faults in the program times the probability of detecting a fault times the fraction of potential fault sites covered by time t.

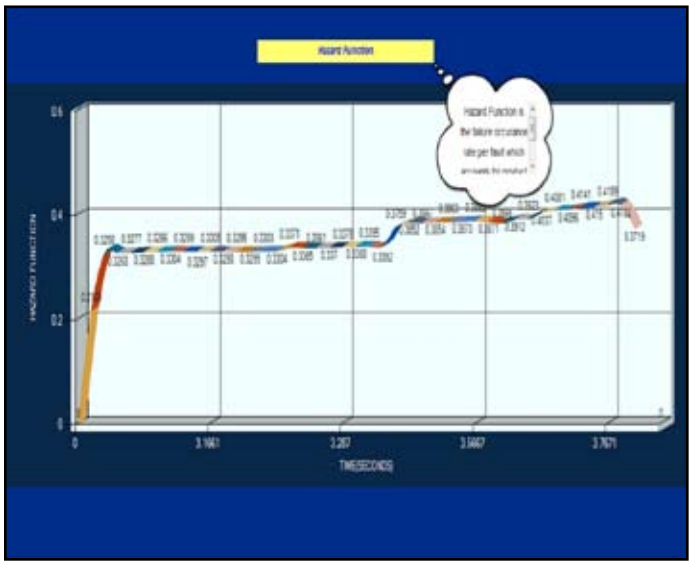

Fig. 4: Hazard Function

3. Hazard Function is the failure occurrence rate per fault which accounts for product testability and test effectiveness.

VI. Conclusion

We have presented software reliability growth model based on ENHPP (Enhanced Non Homogeneous Poisson Process). First, phase space reconstruction can be used to analyze the rule of failure data,SRGM in testing phase of SDLC to ensure the most reliable software and quality.We have to develop ENHPP (Enhanced Non-Homogeneous Poisson Process) model to find out errors and faults in the current and existing system .It will improve the quality and reliability of the software and moreover ,it will eliminate the errors and faults in the current andexisting system .Therefore the SRGM testing will become more authentic .

References

[1] "A Detailed Study of NHPP Software Reliability Models". Journal of Software, Vol. 7, No. 6, June 2012.
[2] P.K. Kapur, Anu G. Aggarwal, Abhishek Tandon,"Two Dimensional Flexible Software Reliability Growth Model With Two Types of Imperfect Debugging", Department of Operational Research, University of Delhi.
[3] Purnaiah B., Rama Krishna V., Bala Venkata Brooks Wd, Motley Rw,"Fault Removal Efficiency in Software Reliability Growth Model", Advances In Computational Research, Vol. 4, Issue 1, 2012.
[4] "Analysis of Discrete Software Reliability Models- Technical Report", (Radctr- 80-84" 1980; New York: Rome Air Development Center.
[5] Vishwas Massey, Prof. K.J.Satao,"Comparing Various SDLC Models and the New Proposed Model on the Basis of Available Methodology".
[6] Inoue S, Yamada S,"Testing-Coverage Dependent Software Reliability Growth Modeling", International Journal of Reliability, Quality,2004.
[7] S.Saira Thabasum,"Need For Design Patterns and Frameworks For Quality Software Development", International Journal of Computer Engineering & Technology (IJCET), Vol. 3, Issue

1, 2012,.
[8] S.Manivannan, Dr.S.Balasubramanian, "Software Process And Product Quality Assurance in it Organizations", International Journal Of Computer Engineering & Technology (Ijcet), Vol. 1, Issue 1, 2010.
[9] Software Reliability Growth Model Based on Fuzzy Wavelet Neural Network- 2010 2nd International Conference on Future Computer And Communication.
[10] International Journal of Computer Engineering and Technology (IJCET), 2012.

Sandeep P. Chavan, Student of M.Tech Computer BVDUCOE, Pune, Maharashtra, India.

Dr. S. H. Patil, Professor, HOD, Department of Computer Engg, BVDUCOE Pune, Maharashtra, India.