ISSN:0975-9646



# Software Reliability and Quality Analyser with Quality Metric Analysis Along With Software Reliability Growth Model

Madhavi Mane[1], Manjusha Joshi[2], Prof. Amol Kadam[3,] Prof. Dr. S.D. Joshi[4]

[1,2,3,4]*Computer Engineering Department,*

*Bharati Vidyapeeth Deemed University College of Engineering*

*Pune-43(India)*

***Abstract*— Software reliability is an important aspect of software quality. And achieving reliability is the need of today's global competition. Estimation and prediction are the ways to analyze software reliability. Software reliability growth model is used to estimate the reliability through mathematical expression and it also used to interpret software failures as a random process. This paper describes a novel software reliability growth model based on non homogeneous Poisson process with allowing for imperfect debugging. Maintaining and improving quality of the software is a very difficult task due to many factors like ambiguous requirement specification, lack of required resources etc. Many reliability growth models have been proposed until now according to different context and hence there is no globally accepted model. Software quality metric highlights the quality aspects of product, process, and project. As there is proportional relationship between quality and reliability, analyzing quality metrics is also a way to estimate reliability. So, we analyze quality metrics along with maintaining the defect database.**

***Keywords*— software reliability growth model (SRGM), quality metrics, non-homogeneous Poisson process, software reliability.**

## I. INTRODUCTION

The software system is extensively used in all kinds of applications such as banking system, telecommunications, and control structure in nuclear power plant as well as defence system. Also integrated electronic devices and automobiles also make use of softwares. There are many known cases of severe consequences of software failure. So, in order to avoid such consequences we need to accomplish more reliability.

Reliability of the software is defined as the probability that software does not deviate from its intended behaviour for a specified time interval[1]. Software should be reliable for accomplishing the improved performance. Small defects can be a reason for severe failure of the system so system design should be reliable. Software reliability growth models are statistical implementation of defect detection data with mathematical equations[4]. Many software reliability growth models have been proposed. Many SRGM's make use of previous failure record which is composed during testing in order to predict the field behaviour of the software with the assumption that testing is performed in accordance with a given operational profile[1]. Most SRGM were developed considering that faults which are identified during testing phase are eliminated rapidly without introducing new faults [2].

The software reliability growth model proposed by Goel-Okumoto[3] presumes that the faults are eliminated quickly after a failure is observed. But the practically fault elimination mechanism takes a time because each observed fault is reported, diagnosed, corrected, and then verified. Goel-Okumoto's research article also provides a way for research on software reliability growth models based on Non-Homogeneous Poisson Process (NHPP). The model depicts the failure inspection phenomenon by using an exponential curve. Software reliability growth models are categorized into two different types[4]:

1. Concave
2. S-shaped

These two models encompass asymptotic behaviour i.e. detection rate of defect reduces as the number of defects enhances. Concave model presumes these assumptions:

1. The detection rate of defect is proportional to the total number of defects in code.
2. As the defect is renovated, there is reduction of defects in the code and therefore there is a reduction in defect detection rate as the number of defects repaired or identified increases.

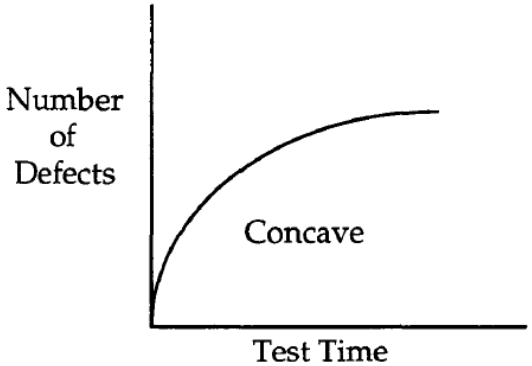


Fig.1 Concave model[4]

S-Shaped model has following assumption:

1. Initial testing is not as enough as later testing, so there is ramp-up period during which the defect detection rate increases.

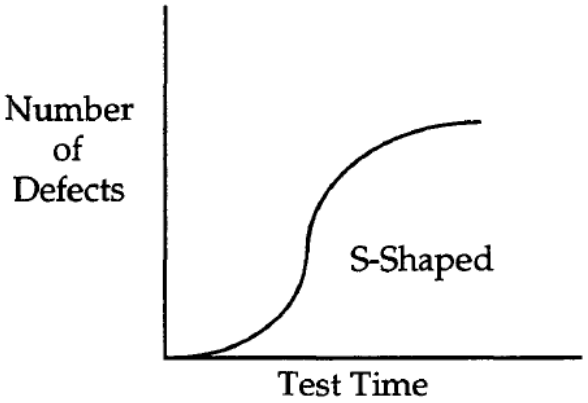


Fig 2. S-Shaped models[4]

There are many software reliability growth models are estimated to assess the problem of reliability measurement. These models consider the previous failure record of software and classify the nature of failure process as follows [5]:

1. Times between failure models:
   This model considers the time interval between failures. This model observes the mean time interval between failure and from that calculates approximately the probable next failure time.
2. Failure count model:
   This class of the model concentrates on total faults or failure count in specified time interval instead of times between failures. The failure counts are assumed to follow a known stochastic process with a time dependant discrete or continuous failure rate.
3. Fault seeding model:
   The idea of this class of model is to induce or "seed" a known number of faults or defects in a program which are assumed to have an unknown number of primitive faults.
4. Input domain based model:
   The class of this model involves creating a set of test cases from an input distribution which is presumed to be representative of the operational usage of the program.

In most of the models it is assumed that fault elimination mechanism is based on perfect debugging i.e. fault elimination process eliminates the fault completely and does not introduce new fault. But practically there is no debugging process which is ideal. Goel[5] introduced the idea of imperfect debugging i.e fault elimination process removes faults without certainty.

Software quality metrics enlighten quality factors or aspects of the product, process, and project [6]. Process metrics are the characteristics or assets of the product such as size, design features, complexity. Process metrics associated with improving software development and maintenance quality. Project metrics associated with project characteristics along with execution. Quality metrics can also organize into the following two types [6]:

1. end product quality metrics
2. In-process quality metrics

This paper proposed Modified Non-Homogeneous Poisson Process, a new software reliability growth model which is based on imperfect debugging. We implement it alongside with software quality metrics and also maintain the defect database. And thus we attempt to achieve more reliability.

## II. BACKGROUND AND MOTIVATION

Software quality is associated with reliability and accomplishing 100% reliability is very complex due to some factors like unambiguous requirement specification, insufficient resources, and complexity. The main inspiration of our project is to achieve more reliability of software so as to maintain quality.

## III. METHODOLOGY:MODIFIED NON HOMOGENEOUS POISSON PROCESS(MNHPP)

The proposed methodology is based on imperfect debugging i.e. detected fault cannot be removed completely and may introduce new errors while recovering the existing defects. We consider that a fault is exponentially distributed over the system. So,

$\mu(t)= (M(t)+Fa)\, e^{-\varphi t}$ …….(1)

Where,

$\mu(t)$ is mean value function for reliability growth model and it represents number of failures expected by time t as estimated by the model.

Fa is the number of faults that are probably activated for some inputs.

M(t) is the total number of faults experienced by system in time t.

And Failure intensity $\lambda(t)$ can be evaluated as follows:

$\lambda(t)= Ne^{-\varphi t}\, (M(t)+Fa)$……..(2)

Where,

N is the initial number of faults exists in software prior to test

So, if we consider that system experience the $m^{th}$ failure at time tm and $(m-1)^{th}$ failure at time $t_{m-1}$ then hazard rate $Z(t_m)$ is estimated as follows:

$Z(t_m)=[( M(t)+Fa)-Id(m-1)]\, \lambda(t)$…….....(3)

Where,

Id is probability of imperfect debugging.

$T_c = N_{pf} / N_{uc}$…….(4)

Where,

$T_c$ = Test coverage,

$N_{pf}$ = Potential faults sites count sensitized by the test,

$N_{uc}$ = Total potential faults sites under consideration,

## IV. PROPOSED SYSTEM ARCHITECTURE

Fig. 3 shows the system architecture. This architecture has the following components:

1. Input software:
   We provide real software as input to the project which is nothing but the whole project along with its coding part.
2. Quality metric analyzer:

Quality metric focuses on quality aspects of software. Quality metric analyzer estimates and analyzes following different quality metrics:

a. Total line count
b. Total number of attributes
c. Number of classes
d. Weighted methods per class
e. Maximum path length from class to root node.

3. Defect database:

This component maintains the defect database like defect tracking system. The defect database includes the details of metrics that exceed than its threshold value.

4. MNHPP Analysis and hazard rate graph generation:

This module evaluates reliability by using Modified Non-Homogeneous Poisson Process and generates the graph for both fault detection rate and test coverage function.

Fig 4 shows the fault detection rate graph. Fig. 5 shows the graph of the test coverage function.

5. Reliability count function

Reliability count function gives the "reliability count value" which can use to identify the extent of reliability. This value is calculated as follows:

$(N/4)K$……(5)

Where,

N is the total number of metrics that exceeds than threshold value.

K is a constant value

K=7.5 because this system is unable to consider all quality metrics and all reliability aspects.

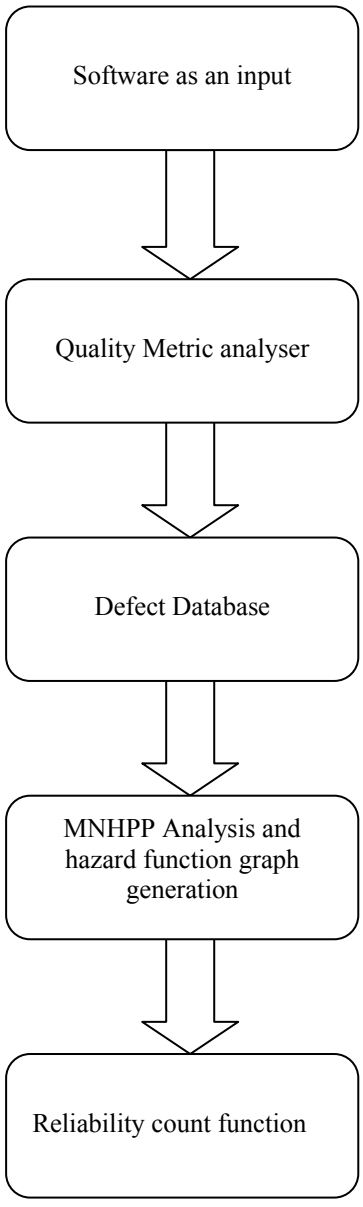


Fig. 3. System architecture

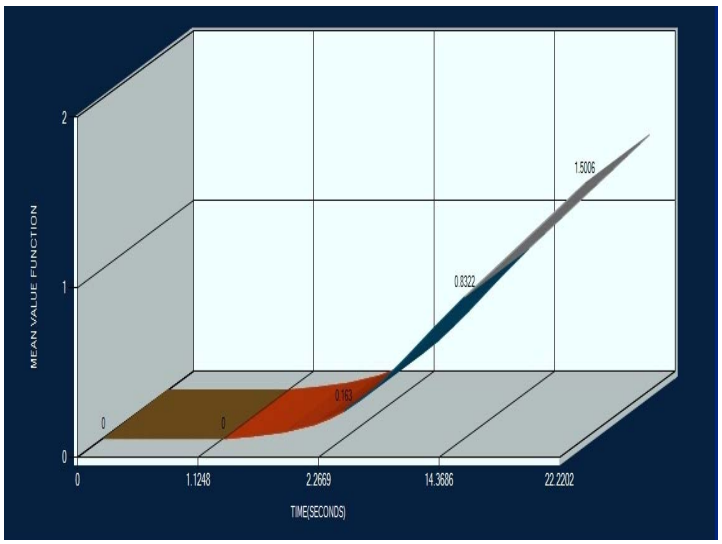


Fig. 4 Defect detection rate

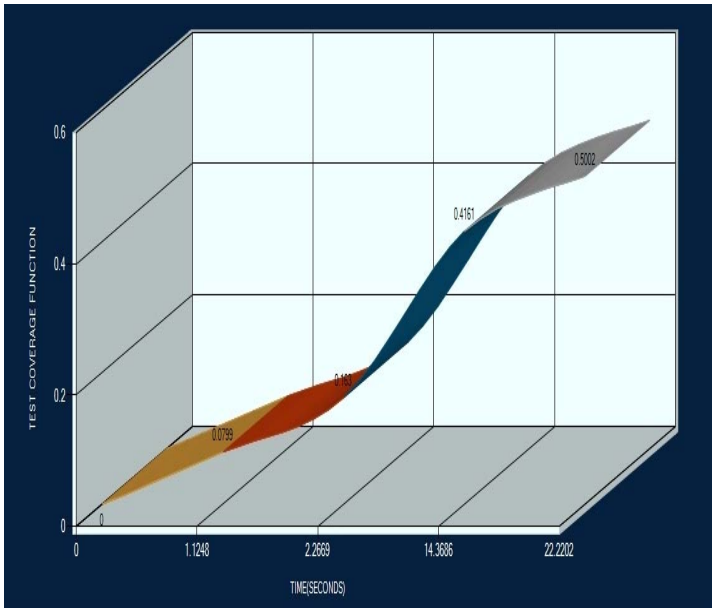


Fig. 5 Test coverage function

## V. CONCLUSION

This paper presented the Modified Non Homogeneous Poisson Process. This method is based on imperfect debugging. We analyse the input software through MNHPP and by using a quality metric analyser. Through our implementation we definitely try to achieve more reliability henceforth enhancing the quality of the software.

## 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 2001

[2] P. K. Kapur, H. Pham, Sameer Yadav, Sameer Anand, Kalpana Yadav, "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.

[3] 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.

[4] Alan Wood, "Software reliabilioty growth models", Technical report 96.1 September 1996.

[5] Amrit Goel, "Software reliability models:Assuptions, Limitations, Applicability", IEEE Transactions on reliability vol SE-11, December 1985.

[6] Stephen H. Kan, "Metrics and models in software quality engineering", Addison-Wesley Professional publication.