30 INTRODUCTION

In this chapter we propose a discrete software reliability growth model based on Non Homogeneous Poisson Process to describe the fault removal phenomenon under imperfect debugging environment. The learning process is taken into consideration by assuming that the probability of imperfect debugging phenomenon is independent on the faults remaining. The model has a flexible structure as it can describe different growth curves ranging from exponential to highly S-Shaped. The applicability of the model is shown by applying on data obtained from different software development projects.

Software Reliability Growth Models (SRGMs) are generally classified into two groups. The first contains the models which use the execution or calendar time as a unit of fault detection (Removal) period and such models are called continuous time models. The second group contains models which use the test occasion (Cases) as a unit of fault
detector (Removal) period and such models are called Discrete SRGMs. A test occasions (Case) can be a single computer test run or a series of computer test runs executed in an hour, day, week or even month. The test occasion (Case) includes the computer test run as well as length of time spent to visually inspect the software source code, Brooks and Motley [1], whereas a computer test run is a set of software input variables arranged in a certain manner to test the functional performance of a particular part of the software system. A large number of models have been developed in the first group while fewer are there in the second group Yamada and Csa1 [8] proposed two discrete SRGM's, Kapur et al [6] proposed a general discrete software reliability growth model based on the assumptions that software may contain several type of faults. In all these models the fault removal process (Fault Debugging) is assumed to be perfect i.e. when an attempt is made to remove a fault, it is removed with certainty. This assumption may not be realistic. Due to the complexity of the software system and the incomplete understanding of the software requirements, Specifications and structure, the testing team may not be able to remove the fault perfectly and the original fault is replaced by another fault. The new fault may generate new failure when this part of the software
system is transversed during the testing. The fault can be removed perfectly when the testing team properly understand the nature of the fault and takes the necessary steps to remove it. The multiple removal of the original fault and their successors i.e. the fault which replaces the original fault, slows down the removal of the original fault and gives rise to S-Shaped Growth Curve. The concept of imperfect debugging was first introduced by Goel [21]. He introduced the probability of imperfect debugging in J.M. Model [21]. Kapur and Garg [5] introduced the imperfect debugging in Goel and Kupoto [3] Model. They assumed that the fault removal rate per remaining faults is reduced due to imperfect debugging. Thus the number of failures observed by time infinity is more than the initial fault content. Although these two models describe the imperfect debugging phenomenon yet the software reliability growth curve of these models is always exponential. Moreover, they assume that the probability of imperfect debugging is independent of the testing time. Thus they ignore the role of the learning process during the testing phase by not accounting for the experience gained with the progress of software testing. Actually, the probability of imperfect debugging is supposed to be maximum in the early stage of testing phase and is supposed to reduce with the progress of testing. Xia
et al [7] also proposed an SRGM considering the role of learning process in the education of the probability of imperfect debugging. This model is based on sound assumptions but the determination of its parameters requires extra information such as initial value of the probability of perfect debugging and the value of the learning factor. This information requires collection of extra data to use the model. Moreover, all these models are continuous time models.

In this chapter, we propose a discrete time SRGM based on Non-Homogeneous Poisson Process (NHPP) to describe the fault removal phenomenon under imperfect debugging environment. The learning process is taken into consideration by assuming that the probability of imperfect debugging is dependent on the number of faults remaining (Removed). The model has a flexible structure and can thus describe different growth curves. Further, the model is tested on a real software fault data obtained from various software development projects. The data sets are cited from Brooks and Motley [1].
ASSUMPTIONS


2. The Software System is subject to failures at random times caused by software faults remaining in the software.

3. The expected number of failures observed between the nth and (n+1)th test run occasions is proportional to the expected number of faults remaining in the software.

4. On the observation of a software failure, the efforts to remove the cause of the failure (the fault) may not be perfect and thus another version of the fault may replace the original fault.

5. The rate of imperfect debugging is decreasing with the testing time and is proportional to the number of faults remaining in the software.

6. The imperfect fault debugging does not increase the initial fault content.

NOTATIONS

a = The initial fault content in the beginning of the testing.

b = The Removal rate per remaining Fault.
\[ c = \text{The Initial imperfect debugging rate per fault.} \]
\[ m_i(n) = \text{The Expected Mean Number of Original faults removed by the } n^{\text{th}} \text{ test occasion (Run).} \]

### 3.3 Model Analysis and Formulations

Under assumptions (3.4), the expected number of original faults removed between the \( n^{\text{th}} \) and \( (n+1)^{\text{th}} \) test runs satisfies the following difference equation:

\[
m_r(n+1) - m_r(n) = b \left[ a - m_r(n) \right] - c \left[ \frac{a - m_r(n+1)}{a} \right] \left[ a - m_r(n) \right] \tag{3.1}
\]

The first term \( b \left[ a - m_r(n) \right] \) represents the intensity of faults debugged, while the negative term represents the intensity of imperfectly debugged faults. In other words, the intensity of faults removed is the intensity of faults debugged minus the expected intensity of the imperfectly debugged faults. To elaborate further, the initial imperfect debugging rate \( c \) is decreasing in the proportion of \( \frac{a - m_r(n+1)}{a} \) as the testing progresses. Therefore, the remaining fault content \( (a - m_r(n)) \) is imperfectly debugged at the rate \( c \left[ \frac{a - m_r(n+1)}{a} \right] \). As the imperfectly debugged faults spawn new version of their own, consequently, these faults will generate more faults. Solving (3.1) using \( m_r(0) = 0 \), we get

\[
m_r(n) = a \frac{(b-c)(1-(1-b)^n)}{(b-c) + c(1-b)^n} \tag{Considering } \phi = \frac{c}{(b-c)}
\]

84
We get
\[ m_r(n) = a \frac{(1-(1-b)^n)}{1 + \phi (1-b)^n} \quad \cdots (3.2) \]
From (3.2), we can see that \( m_r(\omega) = a \). This indicates that the original faults are removed completely after a long time of testing.

34 PARAMETER ESTIMATION :

The Maximum Likelihood Estimation (M.L.E) method is used to obtain the parameter estimates of the model given in (3.2). As the fault removal data used in this chapter are given in the form of pairs \((n_i, x_i)\) \((i=1,2,\ldots,k)\) where \(x_i\) is the cumulative number of faults removed by \(n_i\) test occasion \(0 < n_i < n_k\).

The likelihood function is given by
\[ L(a,b,c|n_i,x_i) = \prod_{i=1}^{k} \frac{[m(n_i) - m(n_i-1)]^{x_i-x_i-1}}{(x_i - x_{i-1})} e^{(-m(n_k))} \quad \cdots (3.3) \]
The parameter estimates are obtained by maximizing (3.3) with respect to each of the parameters. The DMCONE subroutine of IMSL MATH Library is used to maximize (3.3) and obtain the parameters estimates.
DATA ANALYSIS

To check the validity of the model, it is tested on two data sets cited from Brooke and Motley [1].

DS-1 :

The data is given in the form \((n_i, x_i)\); \(i=1, 2, \ldots, 12\) and the number of faults detected by the 12th test occasion is 2657. The estimated values of the model parameters are:

\[
\hat{a} = 3169, \quad \hat{b} = 0.148, \quad \hat{c} = 0.0173
\]

The proposed model estimates presence of imperfect debugging phenomenon in this project. This rate is much lower than the fault debugging rate per remaining fault \((b)\). The fitting of the model is graphically illustrated in Fig. (1). It is clear that the model fits the fault the data excellently. It may be noted that the relationship between the cumulative number of faults and the test occasions is exponential.

DS-2 :

The software fault data is given in the form \((n_i, x_i)\); \(i=1, 2, \ldots, 35\), the number of faults detected by the 35th test occasion is 1301. The estimated value of the model parameters are:

\[
a = 1325, \quad b = 0.181, \quad c = 0.172
\]

The proposed model estimates the presence of imperfect debugging in this project. The initial value of imperfect
debugging rate (c) is close to the value of (b).
This indicates that the imperfect debugging had a significant impact on the progress of the test at the early stages of the testing. This hypothesis is supported by fig. (2) as it is clearly seen that the relationship between the cumulative number of faults removed and the test occasions is not exponential (as the case in DS-1 Fig. (1)) but S-Shaped. In other words, the S-Shapedness in the reliability growth is attributed to the significant presence of imperfect debugging phenomenon. Further, fig. (2) graphically illustrates the goodness of fit of this model. It is clearly seen that the proposed model fits the observed fault data excellently.
Goodness of Fit

Number of Events (Thousands)

[Graph showing goodness of fit with axes labeled: X-axis: [0, 12], Y-axis: [0, 100]. Graph includes a legend with symbols for 'observed' and 'estimated'.]
Goodness of Fit