Software reliability growth models (SRGMs) are fundamental base to assess the reliability growth quantitatively. Discrete SRGMs are required to calculate the accurate software reliability growth. In this chapter, a discrete flexible software reliability growth model (SRGM) based on non-homogeneous poisson process (NHPP) is developed to predict the error removal phenomenon using probability generating function (PGF). Two different forms of failure detection and removal are considered that are dependent on the complexity of the fault. The parameters of fault detection and removal rates depict both exponential and S-shaped curves which show the flexibility of the model. The proposed model incorporates the fault generation function and imperfect debugging phenomenon with learning factors. For estimating the unknown parameters of the proposed discrete model, MLE technique is discussed.

8.1 Introduction

Software reliability engineering has played a vital role during the entire software development life cycle (SDLC) to improve the software quality. The development of the software consists of a sequence of the activities so that the developer may achieve perfection in his effort. A software reliability model should give good predictions of future failure behavior, compute useful quantities, and be widely applicable. Existing models are based on the various assumptions such as development environment, the nature of software failures, the probability of individual failure occurrences, applicability of testing effort, etc..

SRGMs are developed for the analysis of software failures during testing phase. These models are generally classified into two groups (i) Continuous Time Models (ii) Discrete Time Models. In a continuous time model, the execution time (i.e., CPU time) or calendar time is considered whereas in a discrete time model, number of test cases as a unit of fault detection period is used. A test case can be a single computer test run executed in an hour, day, week or even month. Since test cases provide more appropriate measure of the fault detection/removal period than the CPU/calendar time
period used by the continuous time model so the utility of discrete time models cannot be underestimated.

Software development managers require SRGMs which enable us to assess software reliability more accurately than conventional SRGMs proposed so far. A large number of models have been developed in the first group (i.e. continuous time models) while because of the mathematical complexity, there are fewer models in the second group (i.e. discrete time model). Brooks and Motley (1980) made the analysis of discrete software reliability models. Yamada and Osaki (1985a) developed the discrete software reliability growth models. Satoh (2001) studied a discrete bass model and its parameter estimation. Software reliability growth models were also discussed by Satoh and Yamada in (2001) and Kapur et al. (2005b). Satoh and Yamada (2002) did the parameter estimation of discrete logistic curve models for software reliability assessment. Inoue and Yamada (2006) and Kapur et al. (2006) described the discrete software reliability assessment with discretized NHPP models. Shatnawi (2009) studied the discrete time NHPP models for analyzing the software reliability growth phenomenon.

The continuous software reliability growth models (SRGMs) based on non-homogeneous poisson process (NHPP) have proved quite successful in software reliability prediction. Kuo et al. (2001) considered the framework for modelling software reliability using various testing-effort and fault-detection rates. Kapur et al. (2004a) developed a software fault classification model. Shatnawi (2007) proposed the modelling of software fault dependency using lag function.

In the realistic situation during the testing phase, faults are not removed perfectly and some times new errors are introduced during removal process. This phenomenon results in situation where the actual fault removals are less than that of the removal attempts. The concept of imperfect debugging was first introduced by Goel (1985) and till now this concept is widely used in SRGMs. Pham et al. (1999) studied a general imperfect software-debugging model with S-shaped fault detection rate. Kapur et al. (2002) analyzed the discrete imperfect software reliability growth model under imperfect debugging environment. Kapur et al. (2004b) discussed a discrete non-homogeneous poisson process model for software reliability growth with imperfect debugging and fault generation.
Goel and Okumoto (1979) proposed a software reliability growth model assuming that the failure intensity is proportional to the number of faults remaining in the software. This model is very simple and the failures are distributed as exponentially. After then in 1983, Yamada et al. discussed a model whose failures can be described as S-shaped curve. Obha (1984) refined the both models developed by Goel and Okumoto (1979) and by Yamada et al. (1983). He made assumption that the fault detection/removal rate increases with time and there are two types of faults in the software. Some of them were of exponential type and the rests of them were of S-shaped curve types. The models in which failures describe both exponential and S-shaped pattern, are termed as flexible models. Tamura and Yamada (2006) considered a flexible stochastic differential equation model in distributed development environment. Kapur et al. (2008a) proposed the flexible software reliability growth model with testing effort dependent learning process.

In this investigation, we study a flexible discrete NHPP based SRGM having different fault detection and removal rates. This chapter is categorized into six sections. Section 8.1, reviews the previous studies about the discrete software reliability growth model. In order to construct the model mathematically, we present some notations and assumptions in section 8.2. Section 8.3 is devoted to the analysis of reliability of discrete SRGM. In the next section 8.4, we discuss a technique which has been used for parameter estimation. Numerical application and sensitivity analysis is presented in section 8.5. The whole chapter is concluded with a summary in section 8.6.

### 8.1.1 General Discrete NHPP Model

NHPP based SRGMs postulate that the number of faults detected up to testing time \( t \) follows a poisson distribution with mean value function \( m(n) \). We also assume that the software failures occur at random times during testing caused by faults lying dormant in the software. A discrete counting process \( \{N_n, n \geq 0 | n = 0, 1, 2, \ldots\} \) represents the cumulative number of faults detected/removed up to the testing period \( n \). This process is based on a continuous NHPP and has the following properties:

\[
\Pr\{N_n = x | N_0 = 0\} = \frac{\{m(n)\}^x}{x!}\exp\{m(n)\}, \quad n, x = 0, 1, 2, \ldots, \quad \ldots (8.1)
\]
where \( m(n) \) is a mean value function of the discrete counting process, i.e. the expected number of cumulative faults detected up to the \( n^{th} \) testing-period.

### 8.2 Model Description

A recent trend in the development of NHPP SRGMs is the incorporation of some additional information other than the testing time. In many SRGM, it is assumed that during software testing, faults are detected, isolated and removed. But this assumption is not always true. Sometimes during correction or removal process some types of new faults are introduced into the software. We consider a software reliability growth model which is flexible i.e. whose failures depict both exponential and S-shaped curves. The unit of testing is considered as executed cases or runs of the software. In this section we develop NHPP model to present discrete SRG by considering changing fault detection and removal rates.

The proposed model is based on the following assumptions.

- The failure detection/removal phenomenon is modeled by NHPP.
- All faults in a program are mutually independent from the failure detection process.
- Fault detection/isolation rate depends on the number of test cases.
- Fault content function is linearly dependent on the number of test cases.
- Fault introduction rate is a linear function and reflects fault generation.

For formulating the model, the following notations have been used:

- \( m(n) \) : Expected number of faults removed by \( n^{th} \) test case.
- \( a(n) \) : Total number of faults in the software, i.e., fault content function.
- \( b_1(n) \) : Fault detection/isolation rate.
- \( b_2(n) \) : Fault removal rate.
- \( b_1 \) : Constant fault detection rate.
- \( b_2 \) : Constant fault removal rate.
\[ b_i : \text{Initial fault detection rate, (i=3,4,...,k)}. \]
\[ b_f : \text{Final fault detection rate}. \]
\[ \alpha : \text{Fault introduction rate}. \]
\[ \delta : \text{Constant time interval}. \]

### 8.2.1 Model Formulation

We assume that fault content function \( a(n) \) depends on the various factors such as testing strategy, software size, complexity and number of test cases. The expected cumulative number of faults detected between the \( n^{th} \) and \( (n+1)^{th} \) test is given by the following difference equation

\[
\frac{m(n+1) - m(n)}{\delta} = b_1(n)[a(n) - m(n)] + \frac{b_2(n)}{a(n)} m(n+1)[a(n) - m(n)]
\]  \( \text{... (8.2)} \)

**Case 1:** In this case, we assume that the fault detection and removal rates are constant and error content function is increasing linearly. The cause of fault is immediately detected and removed, i.e., debugging is perfect. For this case, we use

\[
a(n) = a(1 + \alpha n),
\]
\[
b_1(n) = b_1,
\]
\[
b_2(n) = b_2,
\]

By substituting the values of \( a(n), b_1(n), \) and \( b_2(n) \) in equation (8.2) and then solving it with initial condition \( m(0) = 0 \), by using probability generating function (PGF) approach, we get

\[
m(n) = a(1 + \alpha n) \left[ \frac{1 - \left[ 1 - \delta(b_1 + b_2) \right]^n}{1 + \frac{b_2}{b_1} \left[ 1 - \delta(b_1 + b_2) \right]^n} \right]. \]  \( \text{... (8.3)} \)

The probability generating function approach for solving equation (8.2) is given in Appendix-1.
Case 2: In this case, the shape of growth curve of the fault detection rate can be both exponential and S-shaped for the parameters $b_r$ and $b_i$; the fault removal rate is constant. In this case debugging process is assumed to be perfect. Now

\[ a(n) = a(1 + \alpha n), \]
\[ b_r(n) = b_r + (b_r - b_i) \frac{m(n+1)}{a}, \]
\[ b_i(n) = b_i, \]

By substituting these values in eq. (8.2) and then solving this with initial condition $m(0) = 0$, we get

\[ m(n) = a(1 + \alpha n) \frac{\left[ 1 - \{1 - \delta(b_2 + b_i)\}^n \right]}{\left[ 1 - \{1 - \delta(b_2 + b_i)\}^n \{1 - \left(\frac{b_2 + b_i}{b_i}\right)\} \right]}, \quad \ldots \text{(8.4)} \]

Special Case:-

(a) When in case (i) error content function is constant, i.e., $a(n) = a$ then our proposed model deduced to Kapur et al. (2005b) model. For this particular case, we have

\[ m(n) = a \left[ 1 - \frac{\{1 - \delta(b_2 + b_i)\}^n}{1 + \frac{b_2}{b_i} \{1 - \delta(b_1 + b_2)\}^n} \right], \quad \ldots \text{(8.5)} \]

Let $n = \frac{t}{\delta}$, then above discrete SRGM converges to continuous time SRGM, if $\delta \to 0$. 

166
Thus \[ a \left[ 1 - \left(1 - \delta(b_1 + b_2)\right)^n \right] \rightarrow a \left[ 1 - \exp\left[-(b_1 + b_2)t\right]\right] \] \[ \frac{1 + b_2 \left(1 - \delta(b_1 + b_2)\right)^n}{1 + \frac{b_2}{b_1} \left(1 - \delta(b_1 + b_2)\right)^n} \rightarrow \frac{1 + b_2 \exp\left[-(b_1 + b_2)t\right]}{1 + \frac{b_2}{b_1} \exp\left[-(b_1 + b_2)t\right]} \] 

\[ \text{(8.6)} \]

(b) In case (ii), if error content function is constant, the proposed discrete time SRGM also converted to continuous time SRGM and can be expressed as

\[ \left[ a \left[ 1 - \left(1 - \delta(b_1 + b_2)\right)^n \right] \right] \rightarrow \left[ a \frac{1 - \exp\left[-(b_1 + b_2)t\right]}{1 - \left\{ \frac{b_2}{b_1} \right\} \exp\left[-(b_1 + b_2)t\right]} \right] \]

\[ \text{(8.7)} \]

8.3 Discrete Reliability Assessment

The failure history of the software may be continuous or discrete. Here we are discussing discrete software reliability model in which the testing has been going on up to \( n^{th} \) testing period by which \( x \) software faults have been detected. The probability that a software failure does not occur in the time interval \( (n, n+h] \) \((h=0, 1, 2, \ldots,)\) can be expressed as

\[ R(n, h) \equiv \Pr\{N_{n+h} - N_n = 0 \mid N_n = x\} \]

\[ = \exp\left[-\left\{ m(n+h) - m(n) \right\} \right], \quad n, h = 0, 1, 2, \ldots, \]

\[ \text{(8.8)} \]

\[ \text{(8.9)} \]

Here \( R(n, h) \) is the software reliability based on the discrete NHPP.

8.3.1 Reliability Growth

In this section, in order to assess the influence of debugging effort during each testing-period on the software reliability, we develop NHPP growth model. We consider the software reliability growth rate which can reflect the influence of the debugging effort during each testing-period to analyze the testing effect of each period on the software reliability.
Chapter-8: Discrete Time based Flexible Software Reliability ....

The software reliability growth rate is formulated as

\[ r(n, h) = R(n, h) - R(n-1, h) \] \( \ldots (8.10) \)

### 8.4 Parameter Estimation

We suggest the maximum likelihood estimation (MLE) technique for estimating the unknown parameters of the developed model. Since data sets are given in the form of pairs \((n_i, y_i; i = 0, 1, 2, \ldots)\) where \(y_i\) is the cumulative number of faults removed by \(n_i\) test cases. Then the likelihood function with mean value function \(m(n)\) for the unknown parameters is given as

\[
L(\text{parameters } / (y_i, n_i)) = \prod_{i=1}^{m} \left[ \frac{m(n_i) - m(n_{i-1})}{(y_i - y_{i-1})!} \right] \exp\left[-\left\{m(n_i) - m(n_{i-1})\right\}\right] \quad \ldots (8.11)
\]

Taking logarithm of eq. (8.11), we get

\[
\ln L = \sum_{i=1}^{m} \left[ (y_i - y_{i-1}) \ln[m(n_i) - m(n_{i-1})] - [m(n_i) - m(n_{i-1})] - \ln(y_i - y_{i-1})! \right] \quad \ldots (8.12)
\]

The parameters of the discrete software reliability growth model can be estimated by maximizing \(L\) from equations (8.11) and (8.12), and the parameters should be satisfied as \(a > 0, b_i > 0, b_f > 0, 0 < p \leq 1, \) and \(\alpha \geq 0\).

### 8.5 Sensitivity Analysis

In this section, we check the validity of the proposed discrete time model to describe the SRGM by taking illustration. We compute the numerical results in software “MATLAB” to facilitate the sensitivity of the proposed model. The cumulative number of errors detected \(m(n)\), reliability of software \(R(n)\) and growth rate of the software are visualized in figs 8.1-8.5, by taking the default parameters as \(a = 400, \alpha = 0.0016, b_1 = 0.055, b_2 = 0.026, b_f = 0.055, b_i = 0.036, \) and \(h = 1\).

For different values of \(a, \alpha, b_1, b_2\) and \(b_f\), figs 8.1 and 8.2 depict the cumulative number of faults detected for cases 1 and 2, respectively. The reliability is displayed for both cases in figs 8.3 and 8.4. In fig. 8.5, the reliability growth rate of the software is examined.
Chapter-8: Discrete Time based Flexible Software Reliability …. 

In figs 8.1(i-ii), we notice that the cumulative number of faults detected increases on increasing either testing time or fault content function (a) or fault introduction rate ($\alpha$). From figs 8.1(iii) and 8.1 (iv), it is clear that m(n) increase with time but with respect to fault detection rate ($b_1$) and fault removal rate ($b_2$), m(n) remains almost constant. It is also seen that m(n) increases for some time (hours) with respect to $b_1$ and $b_2$ and finally it become constant. Fig. 8.2 reveals the same pattern as in fig. 8.1 for all parameters.

The reliability graphs for different parameters are displayed in figs 8.3 and 8.4. We observe that the reliability increases as testing time increases. From figs 8.3(i-ii), we note that as a and $\alpha$ increase, the reliability decreases. But when $b_1$ and $b_2$ increase, the reliability also increases for some time (from 10 to 50 hours) and finally becomes constant. The same trend of reliability is also realized for case 2 for all the parameters.

In figs 8.5(i-ii), we exhibit that there is no significant change in reliability upto testing time $t = 20$ (hours). After that, the reliability starts increasing upto $t = 45$ (hours) and then after tends to decrease with the increase in testing time.

8.6 Concluding Remarks

In this chapter, we have proposed a discrete NHPP software reliability growth model (SRGM) for different fault detection and removal rates but the initial modeling framework was same. The approach taken in this chapter can be applied to other models if a discrete equation that has an exact solution is derived. The approach for estimation of parameters (maximum likelihood estimation technique) of the proposed models has also been discussed.
Fig. 8.1: Cumulative number of faults detected Vs T for different values of (i) \(a\) (ii) \(a\) (iii) \(b_1\) (iv) \(b_2\) for case 1.
Fig. 8.2: Cumulative number of faults detected Vs T for different values of (i) $a$ (ii) $\alpha$ (iii) $b_1$ (iv) $b_2$ for case 2.
Fig. 8.3: Software reliability Vs T for different values of
(i) a (ii) a (iii) b_1 (iv) b_2 for case 1.
Fig. 8.4: Software reliability Vs T for different values of (i) $a$ (ii) $a$ (iii) $b_1$ (iv) $b_2$ for case 2.
Fig. 8.5: Software reliability growth rate Vs T for (i) case 1 and (ii) case 2.
Appendix-1

From equation (8.2), we have

\[ m(n + 1) - m(n) = \delta b_1 [(1 + \alpha n) - m(n)] + \frac{b_2}{a(1 + \alpha n)} \delta m(n + 1) [(1 + \alpha n) - m(n)] \]

\[ \Rightarrow m(n + 1)[1 - \delta b_2] = \delta a b_1 + (1 - \delta b_1) m(n) - \frac{b_2}{a} \delta m(n + 1) m(n) \]

Now, multiplying both side by \( z^n \) and summing over \( n \) from 0 to \( \infty \), we get

\[ \sum_{n=0}^{\infty} z^n m(n + 1) = \delta a b_1 \sum_{n=0}^{\infty} z^n + (1 - \delta b_1) \sum_{n=0}^{\infty} z^n m(n) - \frac{\delta b_2}{a} \sum_{n=0}^{\infty} z^n m(n) m(n + 1) \]

\[ \Rightarrow \frac{1 - \delta b_2}{z} p(z) = \delta a b_1 \sum_{n=0}^{\infty} z^n + (1 - \delta b_1) p(z) - \frac{\delta b_2}{a} \sum_{n=0}^{\infty} z^n m(n) m(n + 1) \]

\[ \Rightarrow (1 - \delta b_2) p(z) = \delta a b_1 \left( z + z^2 + z^3 + \ldots \right) - \frac{\delta b_2}{a} \left[ z^2 m(1)m(2) + z^3 m(2)m(3) + \ldots \right] \]

where \( p(z) = z m(1) + z^2 m(2) + \ldots \)

Now comparing the coefficients of \( z^n \) and \( m(0) = 0 \)

\[ m(1) = \frac{a(1 + \alpha n) b_1}{(1 - b_2)} \] … (8.13)

Again comparing the coefficients of \( z^2 \), we get

\[ (1 - \delta b_2) m(2) - (1 - b_2) m(1) = \delta a b_1 (1 + \alpha n) - \frac{\delta b_2}{a(1 + \alpha n)} m(1) m(2) \] … (8.14)

Substituting the value of \( m(1) \) in eq. (8.14), we get

\[ m(2) = a(1 + \alpha n) \left[ \frac{1 - [(1 - b_1 + b_2)]^2}{1 + \frac{b_2}{b_1} [(1 - b_1 + b_2)]^2} \right] \] … (8.15)

By mathematical induction, we get

\[ m(n) = a(1 + \alpha n) \left[ \frac{1 - [(1 - b_1 + b_2)]^n}{1 + \frac{b_2}{b_1} [(1 - b_1 + b_2)]^n} \right] \] … (8.16)