SDE Based Software Reliability Growth Model Amalgamating Learning Factor on Fault of Different Severity with Multiple Release

Suneeta Bhati¹, A.R.Prassanan², Jagvinder Singh², Yogesh Sharma¹
¹Department of Mathematics, Jodhpur National University, Rajasthan, India
²Department of Mathematics, Maharaja Agersen College, University of Delhi, Delhi, India

Abstract
Software reliability is the possibility of the failure free operation of software in a given period of time under some certain conditions. It is assumed in many researches available in literature, that a similar testing effort is needed on each debugging effort. Yet, in usance different types of faults may require different amounts of testing effort for their detection and removal. Thereupon, faults are sorted into three categories on the substratum of severity – simple, hard and complex.
As the size of software system is large and the number of faults detected during the testing phase because large, so the change of the number of faults that are detected and removed through each debugging becomes sufficiently small compared with the initial fault content at the beginning of the testing phase. In such circumstances, we can model the software fault detection process as a stochastic process with continuous state space.
In this paper, we propose a new software reliability growth model based on SDE type of stochastic differential equation. We consider SDE-base Goel, Okumoto model for simple faults, Yamada s-shaped model for hard faults, and generalized Erlang model for complex faults with different types of learning factor on various types of faults on multiple release.

Keywords: SDE-srgm, multirelease, faults of different severity - simple, hard and complex, different learning functions.

1. Introduction
The Software reliability engineering is velociously flourishing field. The intricacy of business software application is also increasing due to high cost of fixing failures, safety concerns and legal liabilities; consequently, it urges the organization to actualize reliable software. There are multifarious methodologies to consummate Software that is reliable. Software reliability engineering (SRE) addresses all these issues, from design to testing to maintenance phases.
The software reliability growth model (SRGM) is a tool of SRE that can be bestow to assess the software quantatively, bechance test status, schedule status and analyze the changes in reliability performance [6]. The software consists of differential type of faults and each fault requires distinct strategies and different amounts of testing effort to efface it.
Ohba [4] refined the Goel–Okumoto model by assuming that the faults detection/removal rate increase with time and that there are two types of faults in the software. SRGM proposed by Bittani et al [13] and Kapur and Garg [6] has similar forms as that of Ohba [4] but is developed under different set of assumptions. Bittani et al [13] proposed an SRGM exploiting the fault removal (exposure) rate during the initial and final time echoes of testing. Whereas, Kapur and Garg [6] describe a fault removal phenomenon where they assumes that during a removal process of a fault some of the additional faults might be removed without these faults causing any failure. These models can describe both exponential and s-shaped growth curves and therefore are termed as flexible models [13, 6, 4].
Ohba [4] proposed the Hyper exponential SRGM, assuming that software consist of different modules. Each module has its idiosyncrasy and ergo, the faults detected in specific modules have their own peculiarities. Hence, the fault removal rate for each module is not the similar. He contemplates that fault extraction procedure for each module is modeled distinctly and the total fault removal phenomenon is the addition of the fault removal mechanism of all the modules.
Kapur et al [6] proposed an SRGM with three types of faults. The first type is modeled by an Exponential model of Goel and Okumoto [1]. The
second type is modeled by delayed s-shaped model of Yamada et al [14]. The third type is modeled by at three–stage Erlang model proposed by Kapur et al. [14]. The total removal phenomenon is again modeled by the superposition of the three SRGM’s [6, 8]. Later they extended their model to cater for more types of faults [11] by incorporating logistic rate during the removal process.

Faults are detected and removed, after regress testing before the software is divulged into the market. Due to deduction of new faults by the empor, companies release an updated version of the system. Yamada et al. [15] proposed a simple software reliability growth model to describe the fault detection process during the testing phase by applying Ito type of stochastic differential equation (SDE). In this paper, we utilize SDE to represent fault detection rate that incorporates an irregular fluctuations. We proposed a model that includes three different types of faults, for example, simple, hard and complex. Fault detection rate for simple, hard and complex faults are assumed to be time dependent that can incorporates learning as the testing progress on multiple release. The proposed model is applied on four multiple release.

### 2. Assumptions:

a. The fault detection / correction are modeled by on-homogenous process (NHPP/).

b. The number of faults detected at any time is proportional to the remaining number of faults in the software.

c. The time for faults in the beginning of the testing phase is finite.

d. The software faults detection process modeled as a stochastic process with a continuous state space.

e. The number of faults (simple, hard and complex) in the software system gradually decreases as the testing procedure go on.

f. Software is subjects to failure during execution caused by faults remaining in the software.

g. The faults existing in the software are of three types simple, hard and complex. They are distinguished by the amount of effort needed to remove them.

h. During the fault isolation, no new fault is introduced into the system and faults are debugged perfectly.

### 3. Notations:

- $M(t)$: Number of faults detected during the testing time $t$ and is a random variable.
- $E(m(t))$: The mean value function or the expected number of faults detected or removed by time $t$.
- $F(t)$: Probability distribution function.
- $F_i(t)$: Probability distribution function for $i$-th release and $j$-th type of faults ($i=1$ to 4)
- $a$: Total initial faults content in the software.
- $b_j$: Fault detection rate for type $(j=1$ to 3) release in each release ($i=1$ to 4).
- $\sigma_i$: Positive constant that represents the magnitude of the irregular fluctuation.
- $\gamma(t)$: Standard Gaussian white noise.
- $\beta_i$: Logistic learning constant for $i$-th release ($i=1$ to 4)
- $\alpha_i$: Linear learning constant for $i$-th release ($i=1$ to 4)
- $c_i$: Learning constant for $i$-th release ($i=1$ to 4)
- $t_{i-1}$: Time for $i$-th release ($i=1$ to 4)
- $a_i$: Initial fault content in the software.
- $p_i$: Fraction of new simple faults introduced in $i$-th release removed by new simple fault removal rate.
- $p'_i$: Fraction of new hard faults introduced in $i$-th release removed by new hard fault removal rate.
- $(1-p_i-p'_i)$: Fraction of new complex fault introduced in $i$-th release removed by new complex fault removal rate.
\[ \lambda_{ik} \] \quad \text{Fraction of previous } k^{th} \text{ release simple fault removed by new } i^{th} \text{ release simple fault removal rate.} \\
\[ \lambda'_{ik} \] \quad \text{Fraction of previous } k^{th} \text{ release simple fault removed by new } i^{th} \text{ release hard fault removal rate.} \\
(1 - \lambda_{ik} - \lambda'_{ik}) \quad \text{Fraction of previous } k^{th} \text{ release simple fault by new } i^{th} \text{ release complex fault removal rate.} \\
\[ q_{ik} \] \quad \text{Fraction of previous } k^{th} \text{ release hard fault removed by new } i^{th} \text{ release hard fault removal rate.} \\
(1 - q_{ik}) \quad \text{Fraction of previous } k^{th} \text{ release hard fault removed by new } i^{th} \text{ release complex fault removal rate.} \\
\[ \frac{dm(t)}{dt} = \frac{f(t)}{1 - F(t)} \left( a - m(t) \right) \]

It might happen that the rate is not known completely, but subject to some random environmental effect, so that we have:

\[ r(t) = \frac{f(t)}{1 - F(t)} + "\text{noise}" \]

Where, \( r(t) \) is the time dependent fault detection/correction rate.

Let \( \gamma(t) \) be a standard Gaussian white noise and \( \sigma \) be a positive constant representing a magnitude of the irregular fluctuations. So the above equation can be written as:

\[ \frac{dm(t)}{dt} = \left[ \frac{f(t)}{1 - F(t)} + \sigma \gamma(t) \right] \left( a - m(t) \right) \]

The above equation can be extended to the following stochastic differential equation of a \( \text{ito} \) type:

\[ \frac{dm(t)}{dt} = \left[ \frac{f(t)}{1 - F(t)} - \frac{\sigma^2}{2} \right] \left( a - m(t) \right) dt + \sigma \left( a - m(t) \right) dW(t) \]

Where \( w(t) \) is a one-dimensional Wiener process, which is formally defined as an integration of the white noise \( \gamma(t) \) with respect to time \( t \). Using the fact that the wiener process \( w(t) \), is a Gaussian process and has the following properties:

\[
\Pr[w(0) = 0] = 1, \\
E[w(t)] = 0; \\
E[w(t)w(t')]=\min[t,t']
\]

And on applying initial condition \( m(0) = 0 \); we get \( m(t) \) as follows:

\[ m(t) = a[1 - (1 - F(t))e^{-\sigma \gamma(t)}] \]

As we know that the Brownian motion or wiener Process follows normal distribution. The density function of \( w(t) \) is given by:

\[ f(w(t)) = \frac{1}{\sqrt{2\pi t}} \exp\left\{ -\frac{(w(t))^2}{2t} \right\} \]

Thus the mean number of detected fault is given as:

4. Acronyms
DS    Data Set
\( R^2 \)    Coefficient of Multiple Determinations
SPSS Statistical Package for Social Sciences
MSE    Mean Square Fitting Error
PE    Prediction Error
RMSPE Root Mean Square Prediction Error
FDR    Fault Detection Rate

5. SDE Based Modeling Of Up-Gradation For Each Release

In this section, we formulate software reliability growth models by applying \( \text{ito} \) type stochastic differential equations that incorporates three different types of learning functions. Since the faults in the software systems are detected and eliminated during the testing phase, the number of faults remaining in the software system moderately decreases as the testing process goes on.

Let \( \{m(t), t \geq 0\} \) be a random variable which represents the number of software faults detected in the software system up to testing time \( t \). Suppose that \( m(t) \) takes on continuous real value. The NHPP models have treated the software faults detection process in the testing phase as discrete state space. So the corresponding differential equation is given by [11, 12, 10, 5]:

\[ \frac{dm(t)}{dt} = \frac{f(t)}{1 - F(t)} \left( a - m(t) \right) \]
\[ m^*(t) = E(m(t)) = a[1 - (1 - (F(t)))^{\frac{\sigma_t^2}{2}}] \]

(1)

In this paper, we consider three different detection rates. We assume that simple faults are removed by exponential by incorporating learning function as a function of time. And hard faults are removed using a two stage fault removal phenomenon i.e. by learning function as a function of time and complex faults by using three stages Erlang method incorporating the learning function as a function of time [4].

5.1. Simple faults are modeled as:

\[ \frac{dm(t)}{dt} = b(t)(a - m(t)) \]

\[ b(t) = \frac{\alpha + \beta t}{1 + bt} \]

\[ m(t) = a(1 - (1 + bt)^{-\frac{\alpha}{b}} e^{-bt}) \]

\[ m^*(t) = E(m(t)) = a(1 - (1 + bt)^{-\frac{\alpha}{b}} e^{-(bt+\sigma_t^2/2)}) \]

5.2. Hard faults are modeled as:

Xie et al incorporated the concept of learning factor in the model developed by Goel and Okumoto [2], they have assumed that learning function is proportional to the experience of the tester which augments with time. In this paper, we have assumed different learning function in detection rate and correction process. We had assumed that the testing phase is a two stage process. For first stage of testing process the mean number of faults detection \( m_d(t) \), is proportional to the mean number of undetected faults remaining in the software and can be expressed by following differential equation.

\[ m'_d(t) = b(t)(a - m_d(t)) \]

Where, \( b(t) = \frac{\alpha + \beta t}{1 + bt} \)

Solving the equation (1) with initial conditions \( m_d(0) = 0 \)

We obtain,

\[ m_d(t) = a(1 - (1 + bt)^{-\frac{\alpha}{b}} e^{-(bt+\sigma_t^2/2)}) \]

It can be observed that as \( t \to \infty, b(t) \to \frac{\beta}{b} \)

In the second stage, the fault correction rate is proportional to the mean number of faults detected but not yet corrected faults remaining in the system. In this stage fault correction rate is assumed as logistic learning function and it can be expressed in terms of the differential equation as:

\[ \frac{dm_c(t)}{dt} = b(t)(m_d(t) - m_c(t)) \]

where,

\[ b(t) = \frac{\left(\frac{\beta}{b}\right)}{1 + c e^{-\left(\frac{\beta}{b}\right)t}} \]

Solving equation (2), with initial conditions \( m_c(t) = 0 \),

The mean number of faults corrected is given by,

\[ m_c(t) = a(1 - (1 + c + (\left(1 + (bt)^{-\frac{\alpha}{b}}\right)))^{-\frac{1}{(\beta - \alpha)}})^{-\frac{1}{(\beta - \alpha)}} \]

5.3. Complex fault:

complex faults is modeled as a three stage process to represent the severity of complex faults assuming fault removal rate per remaining fault \( b(t) \) to be logistic function to describe the learning of the testing team.
\[
\begin{align*}
\frac{d(m_i(t))}{dt} &= b(a - m(t)) \\
\frac{d(m_i(t))}{dt} &= b(m_i(t) - m_i(t)) \\
\frac{d(m_i(t))}{dt} &= b(t)(m_i(t) - m_i(t)) \\
\text{Where,} \\
b(t) &= \frac{c}{1 + ce^{-bt}} \\
\text{Thus,} \\
m(t) &= a\left(\frac{1 - \frac{b^2t^2}{2}e^{(-bt)}}{1 + ce^{-bt}}\right) \\
\text{And,} \\
m^*(t) &= E(m(t)) = a\left(1 - \frac{(1 + c + bt + \frac{b^2t^2}{2})}{(1 + ce^{-bt})^2}\right)^2 
\end{align*}
\]

6. Modeling fault removal process for multiple software release

6.1 Release 1:
In release 1, simple faults are removed exponentially by incorporating the learning function as a function of time by testing team, simple faults by exponential model of Goel-Okumoto, hard faults by Yamada’s s-shaped model amalgamate the learning function as a function of time and complex faults by Erlang method imbibing the learning function.

\[
M_1(t) = p_a a_1 F_{11}(t) + p'_a a_1 F_{12}(t) + (1 - p_i - p'_i) a_1 F_{13}(t), \quad 0 < t < t_1
\]

\[
F_{11} = (1 - (1 + bt)) \left(\frac{a}{b} \right)^{-bt + \sigma_i^2} \frac{2}{2} 
\]

\[
F_{12} = \frac{1}{(1 + ce^{-bt})} \left(\frac{a}{b} \right)^{-bt + \sigma_i^2} \frac{2}{2} 
\]

\[
F_{13} = (1 - \frac{1}{1 + ce^{-bt}}) \left(\frac{a}{b} \right)^{-bt + \sigma_i^2} \frac{2}{2} 
\]

The following pie–chart for subsequent for releases-1 is:

6.2 Release-2:
Accession of a few novel functionality to the software beget to modification of the code. These neoteric specifications in the code lead to codicil of the fault content. Now the testing team starts testing the upgraded system, apart from this the testing team heed dependency and effect of adding new functionalities with existing system. Amid testing the newly formed code, there is always a possibility that the testing team may find some faults (simple, hard and complex) which were present in formerly developed code. In this period left over simple faults \( p_a a_1 (1 - F_{11}(t_1)) \), left over hard fault \( p_a a_1 (1 - F_{12}(t_1)) \) and left over complex fault \( (1 - p_i - p'_i) a_1 (1 - F_{13}(t)) \) of the first iteration interacts with new simple, hard and complex detection /correction rate. A fraction \( \lambda_{21} \) of remaining simple faults from first version interacts with new simple rate and a fraction \( \lambda'_{21} \) of remaining simple fault from first version.
interacts with new hard rate whereas the remaining fraction \((1 - \lambda_{21} - \lambda_{21}')\) of faults from first version interacts with new detection/correction rate. Similarly, for the remaining hard faults \(p_1a_1(1 - F_{12}(t_1))\) of the first iteration interacts with new detection/correction rate. A fraction of \(q_1p_1a_1(1 - F_{12}(t_1))\) interacts with new hard rate and remaining fraction \((1 - q_1)p_1a_1(1 - F_{12}(t_1))\) with new complex rate. Similarly, the remaining complex fault from first iteration \((1 - p_1 - p_1')a_1(1 - F_{13}(t_1))\) which is interacted with new complex detection/correction rate. In addition, faults are generated due to the enhancement of the features, a fraction of these faults are also removed during the testing with new detection rate i.e. \(F_{21}(t-t_1)\) for simple faults and \(F_{22}(t-t_1)\) for hard faults and \(F_{23}(t-t_1)\) for complex faults. The change in the fault detection is due to change in time, change in the complexity due to new features, change in testing strategies etc. The resulting equation can be written as:

\[
M_2(t) = p_2a_2F_{21}(t) + p_2'a_2F_{22}(t-t_1) + (1-p_2-p_2')a_2F_{23}(t-t_1)
\]

\[
+ \lambda_{21}p_1a_1(1 - F_{11}(t_1))F_{21}(t-t_1)
\]

\[
+ \lambda_{21}'p_1a_1(1 - F_{12}(t_1))F_{22}(t-t_1) + (1 - \lambda_{21} - \lambda_{21}')[p_1a_1(1 - F_{13}(t_1))F_{23}(t-t_1)]
\]

\[
+ (1 - q_1p_1'a_1(1 - F_{12}(t_1))F_{23}(t-t_1))
\]

\[
+ (1 - p_1 - p_1')a_1(1 - F_{13}(t_1))F_{23}(t-t_1)
\]

\[
t_1 < t < t_2
\]

6.3. Release-3:

Similarly for release 3, we consider faults generated in third release and remaining number of simple, hard and complex faults from the second release. As the parameters are more in the proposed model in release -3 compare to no. of available data points in tendem data. So, we increase the data points by taking series mean of data points of available no. of faults detected in tendem data which is 37.92. The proposed model and the corresponding mathematical equation can be represented as follows:

\[F_{21} = (1-(1+bt)e^{-\frac{(t-bt+\gamma t^2)}{2}})\] simple faults

\[F_{22} = (1-(1+(t+bt)e^{-\frac{(b^2-\alpha +t)}{2}})\] hard faults

\[F_{23} = (1-(1+e+bt+\frac{\delta^2t^2}{2})_{(-bt+\beta t)^2})\] complex faults
6.4. Release-4:
The procedure of inclusion of new functionalities is an ongoing process. These add-ons keep on occurring till software is present in the market. As a result these proceedings help in ameliorableness of software as well as in accretion of reliability of the product because bounteous faults are removed when testing and integration of code is rendered. We have discussed a case when the new features are added in the software for the third time:

$$t_2 < t < t_3$$

$$F_{31} = (1-(1+bt)^{-\frac{\lambda_3}{\beta_3}}) e^{(-bt+\sigma t)^2}$$..............simple faults

$$F_{32} = (1-((1+bt)^{-\frac{\beta}{\alpha} - 1}) \left(\frac{b}{\beta} - \frac{\alpha}{b} + 1\right)/(1 + ce^{-(\frac{\beta}{\alpha} - 1)})) e^{(-bt+\sigma t)^2}$$......Hard faults

$$F_{33} = (1-\left(\frac{1+c+bt + \frac{b^2t^2}{2}}{1+ce^{-bt}}\right)e^{(-bt+\sigma t)^2})$$.............complex faults

<table>
<thead>
<tr>
<th>Faults which are removed by new simple detection rate</th>
<th>Faults which are removed by new hard detection rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leftover simple faults of 1st release</td>
<td>Leftover simple fault of 1st release</td>
</tr>
<tr>
<td>Leftover simple faults of 2nd release</td>
<td>Leftover simple fault of 2nd release</td>
</tr>
<tr>
<td>New simple fault of 3rd release</td>
<td>Leftover hard fault of 1st release</td>
</tr>
<tr>
<td>Leftover hard fault of 2nd release</td>
<td>Leftover simple fault of 2nd release</td>
</tr>
<tr>
<td>Leftover simple fault of 2nd release</td>
<td>Leftover hard fault of 2nd release</td>
</tr>
<tr>
<td>New hard fault of 3rd release</td>
<td>Leftover complex fault of 2nd release</td>
</tr>
</tbody>
</table>

Faults which are removed by new complex detection rate:

$$M_3(t) = p_3a_3F_{31}(t-t_2) + p_3a_3F_{32}(t-t_2)$$

$$(1-p_3-p_3^2)a_3F_{33}(t-t_2) + \lambda_3^2(p_2a_2)$$

$$(1-F_{21}(t_2-t_1))F_{31}(t-t_2) + \lambda_3^2(p_2a_2)$$

$$(1-F_{21}(t_2-t_1))F_{32}(t-t_2) + (1-\lambda_3^2-\lambda_3^3)$$

$$p_2a_2(1-F_{21}(t_2-t_1))F_{33}(t-t_2) + q_3(1-p_2a_2)$$

$$(1-F_{22}(t_2-t_1))F_{32}(t-t_2) + (1-q_3^2)(p_2a_2)$$

$$q_3(1-F_{22}(t_2-t_1))F_{33}(t-t_2) + (1-p_2-p_2^2)a_2$$

$$(1-F_{23}(t_2-t_1))F_{33}(t-t_2) + \lambda_3^1(p_1a_1)$$

$$(1-F_{11}(t_1)) (1-(\lambda_2F_{21}(t_2-t_1)+\lambda_2^2F_{22}(t_2-t_1))$$

$$(1-\lambda_2^3)(p_1a_1(1-F_{11}(t_1))F_{33}(t-t_2)+\lambda_3^1(p_1a_1)$$

$$(1-F_{11}(t_1)) (1-(\lambda_2F_{21}(t_2-t_1)+\lambda_2^2F_{22}(t_2-t_1))$$

$$(1-\lambda_2^3-\lambda_3^1)(p_1a_1(1-F_{11}(t_1))F_{33}(t-t_2)+\lambda_3^1(p_1a_1)$$

$$(1-\lambda_2^2F_{22}(t_2-t_1)+(1-\lambda_2^2-\lambda_2^3))$$

$$F_{33}(t-t_2) + q_3(1-p_1a_1(1-F_{12}(t_1))F_{33}(t-t_2)$$

$$(1-q_3^2)F_{23}(t_2-t_1))F_{33}(t-t_2)+(1-q_3^2)$$
\[ M_4(t) = p_4 a_4 F_{41}(t-t_3) + p_4 a_4 F_{42}(t-t_3) + (1-p_4-p_4') \]

\[ a_4 F_{43}(t-t_3) + \lambda_{43} p_3 a_3 (1-F_{31}(t-t_3)) F_{41}(t-t_3) \]

\[ + \lambda_{43} p_3 a_3 (1-F_{31}(t-t_3) F_{42}(t-t_3) + (1-\lambda_{43} - \lambda_{43}') \]

\[ q_{32} p_3 a_3 (1-F_{32}(t-t_2)) F_{43}(t-t_3) + \lambda_{42} p_2 a_2 \]

\[ F_{42}(t-t_3) + (1-q_{32}) p_2 a_2 (1-F_{32}(t-t_2)) F_{43}(t-t_3) \]

\[ + (1-q_{32}) p_2 a_2 (1-F_{32}(t-t_2)) F_{43}(t-t_3) + \lambda_{42} p_2 a_2 \]

\[ (1-F_{21}(t-t_1))(1-(\lambda_{32} F_{31}(t-t_2) + \lambda_{32} F_{32}(t-t_2) \]

\[ + (1-\lambda_{32} - \lambda_{32}') F_{33}(t-t_2))) F_{41}(t-t_3) + \lambda_{41} \]

\[ (1-\lambda_{21}) p_1 a_1 (1-F_{11}(t_1))(1-(\lambda_{21} F_{21}(t-t_1) \]

\[ + (1-\lambda_{21} - \lambda_{21}') F_{22}(t-t_1))) \]

\[ F_{41}(t-t_3) + \lambda_{41} (1-p_1 a_1 (1-F_{11}(t_1))(1-(\lambda_{21} F_{21}(t-t_1) \]

\[ + (1-\lambda_{21} - \lambda_{21}') F_{22}(t-t_1))) \]

\[ (1-\lambda_{31} - \lambda_{31}')(F_{33}(t-t_2))) F_{42}(t-t_3) \]

\[ + (1-\lambda_{41} - \lambda_{41}')(F_{33}(t-t_2))) F_{42}(t-t_3) \]

\[ (1-\lambda_{21} F_{21}(t-t_1) + \lambda_{21} F_{22}(t-t_1)) (1-\lambda_{21} - \lambda_{21}')(F_{33}(t-t_2)) \]

\[ F_{23}(t-t_1) \]

\[ (1-\lambda_{31} - \lambda_{31}')(F_{33}(t-t_2))) F_{43}(t-t_3) \]

\[ + \lambda_{31} F_{32}(t-t_2) + (1-\lambda_{31} - \lambda_{31}')(F_{33}(t-t_2)) \]

\[ F_{33}(t-t_2))) F_{44}(t-t_3) \]

\[ (1-\lambda_{41} - \lambda_{41}')(F_{33}(t-t_2))) F_{44}(t-t_3) \]

\[ (1-\lambda_{21} F_{21}(t-t_1) + \lambda_{21} F_{22}(t-t_1)) (1-\lambda_{21} - \lambda_{21}')(F_{33}(t-t_2)) \]

\[ F_{23}(t-t_1) \]

\[ (1-\lambda_{31} - \lambda_{31}')(F_{33}(t-t_2))) F_{43}(t-t_3) \]

\[ + \lambda_{31} F_{32}(t-t_2) + (1-\lambda_{31} - \lambda_{31}')(F_{33}(t-t_2)) \]

\[ F_{33}(t-t_2))) F_{44}(t-t_3) \]

\[ (1-\lambda_{41} - \lambda_{41}')(F_{33}(t-t_2))) F_{44}(t-t_3) \]
7. Model validation, Data Set and Data Analysis

To check the validity of the proposed model and to describe the software reliability growth, it has been tested on tandem computer four release data set. Also we have used non linear least square technique in SPSS software for estimation of parameters. Estimated value of parameters of each releases are given in Table 1. Table 2 shows the comparison criterion of the four software releases. Based on data available given in Table 1, the performance analysis of proposed model is measured by the four common criteria that we define as below:

7.1. Criteria for comparisons

To give quantitative comparisons, some criteria were used to judge the performance of the proposed model. Here we let \( n \) represent the sample size of selected data set, \( y_i \) represent the actual number of faults by time \( t_i \) and \( m(t_i) \) represent the estimated number of faults by time \( t_i \). In all mentioned criteria the lower value indicate less fitting error.

7.1.1 The Bias is defined as:

\[
Bias = \frac{1}{n} \sum_{i=1}^{n} \left( \frac{\hat{m}(t_i) - y_i}{n} \right)
\]

The difference between the observation and prediction of number of failures at any instant of time \( i \) is known as PE\(_i\) (Prediction error). The average of PEs is known as bias. Lower the value of Bias better is the goodness of fit.

7.1.2 The Variation is defined as:

\[
Variation = \sqrt{\frac{\sum_{i=1}^{n} (\hat{m}(t_i) - y_i - Bias)^2}{(n-1)}}
\]

The average of the prediction errors is called the prediction Bias, and its standard deviation is often used as a measure of the variation in the predictions.

7.1.3 The Root Mean Square Prediction Error (RMSPE) is defined as:

\[
RMSPE = \sqrt{(Bias^2 + Variation^2)}
\]

RMSPE is a measure of the closeness with which the model predicts the observation.

7.1.3 The Mean Square Error (MSE) is defined as:

The difference between the expected values, \( \hat{m}(t_i) \) and the observed data \( y_i \) is measured by MSE as follows:

\[
MSE = \frac{1}{k} \sum_{i=1}^{k} (\hat{m}(t_i) - y_i)^2
\]

Where \( k \) is the number of observations. The lower MSE indicates less fitting error, thus better goodness of fit.
8. Graphs for four release:

Goodness of fit curve for release-1

Goodness of fit curve for release-2

Goodness of fit curve for release-3

Table 1: Parameter Estimate

<table>
<thead>
<tr>
<th>i=1to4</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>a_i</td>
<td>194.000</td>
<td>20.081</td>
<td>45.837</td>
<td>38.417</td>
</tr>
<tr>
<td>b_{i1}</td>
<td>.005</td>
<td>2.376</td>
<td>3.298</td>
<td>11.606</td>
</tr>
<tr>
<td>b_{i2}</td>
<td>.004</td>
<td>.002</td>
<td>.002</td>
<td>23.280</td>
</tr>
<tr>
<td>b_{i3}</td>
<td>.004</td>
<td>.399</td>
<td>.002</td>
<td>.509</td>
</tr>
<tr>
<td>p_i</td>
<td>.134</td>
<td>.300</td>
<td>.139</td>
<td>.007</td>
</tr>
<tr>
<td>p_i'</td>
<td>.390</td>
<td>.300</td>
<td>.010</td>
<td>.223</td>
</tr>
<tr>
<td>p_i''</td>
<td>.001</td>
<td>5.000</td>
<td>6.000</td>
<td>38.347</td>
</tr>
<tr>
<td>p_i'''</td>
<td>1.000</td>
<td>1.000</td>
<td>12.000</td>
<td>5.000</td>
</tr>
<tr>
<td>c_i</td>
<td>19.586</td>
<td>5.000</td>
<td>90.000</td>
<td>41.593</td>
</tr>
<tr>
<td>σ_i</td>
<td>4.657E-006</td>
<td>4.302E-005</td>
<td>.000</td>
<td>.000</td>
</tr>
<tr>
<td>λ_{i1}</td>
<td>-</td>
<td>.070</td>
<td>.100</td>
<td>.200</td>
</tr>
<tr>
<td>λ_{i1}'</td>
<td>-</td>
<td>.200</td>
<td>.200</td>
<td>.300</td>
</tr>
<tr>
<td>q_{i1}</td>
<td>-</td>
<td>.300</td>
<td>.100</td>
<td>.469</td>
</tr>
<tr>
<td>λ_{i2}</td>
<td>-</td>
<td>-</td>
<td>.100</td>
<td>.200</td>
</tr>
<tr>
<td>λ_{i2}'</td>
<td>-</td>
<td>-</td>
<td>.300</td>
<td>.300</td>
</tr>
<tr>
<td>q_{i2}</td>
<td>-</td>
<td>-</td>
<td>.400</td>
<td>.400</td>
</tr>
<tr>
<td>λ_{i3}</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>.100</td>
</tr>
<tr>
<td>λ_{i3}'</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>.300</td>
</tr>
<tr>
<td>q_{i3}</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>.400</td>
</tr>
</tbody>
</table>

Table 2: Comparison Criteria

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>II</th>
<th>III</th>
<th>IV</th>
</tr>
</thead>
<tbody>
<tr>
<td>M.S.E</td>
<td>5.8635</td>
<td>10.8520</td>
<td>65.0924</td>
<td>10.7718</td>
</tr>
<tr>
<td>Bias</td>
<td>.0301</td>
<td>-.3574</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Variation</td>
<td>2.48416</td>
<td>3.364521</td>
<td>8.316287</td>
<td>3.367335</td>
</tr>
<tr>
<td>R2</td>
<td>.993</td>
<td>.992</td>
<td>.783</td>
<td>.939</td>
</tr>
<tr>
<td>RMSPE</td>
<td>2.484343</td>
<td>3.38345</td>
<td>8.316287</td>
<td>3.367335</td>
</tr>
</tbody>
</table>
Conclusion:
In this paper we have developed the SRGM incorporating stochastic differential equation of Itô type using different learning functions different severity of faults on multiple release. In future we propose to develop a SRGM model of stochastic differential equation of Itô type incorporating learning function on n-types of faults on four multiple releases.

References: