Stochastic Equation Based Modeling on Multi Release Incorporating Learning Effect and Two Types of Imperfect Debugging on Faults of Different Severity

A.R. Prasanan¹, Suneeta Bhati², Javinder Singh¹, Yogesh Sharma²
¹ (Department Of Mathematics, Maharaja Agersen College, University Of Delhi)
² (Department Of Mathematics, Jodhpur National University, Rajasthan)
arprasanan@gmail.com  suneetachaudhary2012@gmail.com
jagvinder.singh@gmail.com dr.ashokvyas@gmail.com

Abstract
Software reliability refers to the likelihood of software to function without failure for a specified duration of time under few conditions. Reliability is very imperative to amplify the software persistence and its longevity. Software testing illustrates the process that ferrets out the faults in whole and worth of developed computer software. Entire eradication of faults from software is nonviable due to software intricacy and caliber of testing team, this event is termed as imperfect debugging. Error generation is defined as the procedure in which the faults are imperfectly removed and additional faults emanate from these existing faults. The inbuilt flexibility of the proposed model takes care of different environment ranging from exponential to s-shaped to three stage Erlang model. As the size of software system is large and the number of faults detected during the testing phase becomes 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 a situation, we can model the software fault detection process as stochastic process with continuous state space. In this paper, we propose a new software reliability growth model based on Itô type of stochastic differential equation. In this paper we have proposed an SDE based SRGM which consider the learning effect and experienced gained by the testing team as testing progress in the presence of two types of imperfect debugging.

Key words: NHPP, SDE-Itô type, imperfect debugging, error generation, multi up-gradation, severity of faults

1. Introduction
Computers become the imperative part of every company in the 21st century which amplifies the dependency on computer software consequently repeatedly facing the failures due to increment in size and complexity of software has grown up. During development of software, software reliability should be considered so as, to abstain the failure and faults which results in reliable software. There are several projects executed by NASA, and DOD that deal with highly sophisticated software. Due to change in the three lines of code in a single program me in 1991 the telephone system was collapsed in California and eastern parts.

Software reliability is pivotal characteristic of software quality. Reliability is the property of referring how well software meet its requirement and also the probability of failure free operation for the specified period of time in a specified environment. Among all software reliability model SRGM is probably one of the most efficient mechanism in the literature with more than hundred models functioning in assorted forms. In application, srgps confront distinguished challenges. Firstly, software testers, scarcely pursue the operation profile to test the software, so, what is perceived during software testing is indirectly protractible to operation use. Secondly, when the chunk of failures accumulated in a project is circumscribed, it is tough to get numerically relevant reliable predictions.

And lastly, the most of assumptions, of srgps are indiscreet e.g. assumptions that faults are independent of each other, each fault has the same chance to be spotted in one chance, and correction of a fault conceal new faults. The testing team may not be able to remove the fault perfectly on the detection of failure and the original fault may remain or replace by another fault. Because of incomplete understanding of the internal structure of the software while the first phenomenon is termed as imperfect fault removal the second is called error generation. In case of imperfect fault removal, the fault content increases as the testing progress and removal results in introduction of new faults while removing old ones. It was Goel [3] who first introduces the concept of imperfect debugging. He introduced the probability of imperfect debugging in Jenski and Moranda model, due to Chou and Ohhba [1] an error generation model is applied and is removed as complex fault in new release, also [6] introduced the imperfect fault thus the number of failures observed by time infinity is more than the initial fault content. Although these model describe two types of imperfect debugging phenomenon yet the software reliability growth curve of these models is always exponential. Moreover, they assume that probability of imperfect debugging is independent of the testing team.
Moreover, several continuous state space srgm based on stochastic differential equation of It\^o type to measure the reliability growth of software have been developed corresponding to discrete state space NHPP based SRGM. Yamada et. al [13] proposed a simplified software reliability growth model to describe the fault detection during the testing phase by applying It\^o type stochastic differential (SDE) and have compared the continuous state space srgm with the NHPP. Yamada et. al [14] has developed several SRGM based on stochastic differential equations of It\^o type such as exponential delayed s-shaped and inflection s-shaped. Ke et al.[7] used SDE to represents a per- fault detection rate that incorporates an irregular fluctuation instead of an NHPP, and consider a per- fault detection rate that depends on the testing time t. Yamada et al [15] have proposed a flexible stochastic differential equation model describing a fault detection process during the system testing team phase of the distributed development environment. Recently, Kapur et al [5] proposed, a composite model called generalized SRGM based on stochastic differential equation of It\^o type which incorporates three different types of faults e.g. Simple, hard and complex [9]. In the proposed model, we consider that undetected simple fault of previous release are removed as hard fault during new release but hard fault of previous release are assumed not to change its nature during testing of next release and it is removed as hard only. Similarly, for complex fault of previous release which is considered as complex fault and is removed by new complex fault detection rate.

1.1 Assumptions:

1) Failure observation / fault removal phenomenon is modeled by NHPP stochastic process with a continuous state space.
2) The software fault detection process is modeled as a stochastic process with a continuous state space.
3) Software is subject to failure s during execution caused by faults removing in the software.
4) Each time software a failure is observed, an immediate effort takes place to find the cause of the failure in order to remove it.
5) Failure rate is equally affected by all the faults remaining in the software.
6) When a software failure occurs, an instantaneous repair effort starts and the following may occur.
   i) Fault content is reduced by one with probability p
   ii) Fault content remains unchanged with probability 1-p.
7) During the fault removed process, whether the fault is removed successfully or not, new faults are generated with a constant probability.
8) Fault removal rate is considered to be learning function to describe the learning effect of the fault removal team.

1.2 Notation:

\[ m(t) : \text{Number of faults detected during the testing time } t \]
\[ E m(t) : \text{The mean value function or the expected number of faults detected or removed by time } t. \]
\[ f(t) : \text{Probability density function.} \]
\[ F(t) : \text{Probability distribution function.} \]
\[ F_{ij}(t) : \text{Probability distribution function for } i \text{th release and } j \text{th type of faults } (i=1to4) \]
\[ a : \text{Total initial faults content in the software} \]
\[ b_{ij}(t) : \text{Fault detection rate for type } (i=1to3) \text{ release in each release } (i=1to4) \]
\[ \alpha_{ij} : \text{Fraction of new hard faults introduced in ith release removed by new hard fault removal rate.} \]
\[ \lambda_{ik} : \text{Fraction of previous kth release simple fault removed by new ith release simple fault removal rate.} \]
\[ q_{ik} : \text{Fraction of previous kth release hard fault removed by new ith release hard fault removal rate.} \]
\[ \mu_i : \text{Probability of fault removal on a failure i.e. probability of perfect debugging.} \]
\[ \theta_i : \text{Rate at which error may be introduced during debugging process per detected faults.} \]
\[ \alpha_i : \text{Linear learning constant for ith release } (i=1to4) \]

Acronyms
\[ DS : \text{Data Set} \]
\[ R^2 : \text{Coefficient of Multiple Determinations} \]
\[ SPSS : \text{Statistical Package for Social Sciences} \]
\[ MSE : \text{Mean Square Fitting Error} \]
\[ PE : \text{Prediction Error} \]
\[ RMPSE : \text{Root Mean Square Prediction Error} \]
\[ FDR : \text{Fault Detection Rate} \]

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

The srgm presented in this paper are based upon NHPP and SDE of It\^o type. The NHPP models are based on the assumptions that the software system is subject to failures.
at random times caused by manifestation of remaining faults in the describe the failure phenomenon during the testing phase. The counting process \( \{N(t), t \geq 0\} \) of an NHPP process is given as follows:

\[
\Pr\{N(t) = k\} = \frac{(m(t))^k}{k!} e^{-m(t)}, k = 0, 1, 2,
\]

And

\[
m(t) = \int_0^t \lambda(x)dx(x)
\]

The learning of testing team grows as testing progress. Xie et al [12] has proposed a learning function in testing. Hence, we considered imperfect debugging and fault, generation during testing along with learning of testing team. Let \( N(t) \) be a random variable which represents the number of software faults detected in the software system up to testing time \( t \). the faults detected in \( (t, t + \Delta t) \) are proportional to mean number of faults remaining in the system. So, the corresponding differential equation will be given by,

\[
\frac{dN(t)}{dt} = pb(t)(a(t) - N(t)) \cdots (1)
\]

Above differential equation can be written as

\[
\frac{dN(t)}{dt} = pb(t)(f(t) + \sigma N(t)) - w(t)) \cdots (2)
\]

This can be re-written as

\[
\frac{dN(t)}{dt} = p(1-\gamma)b(t)(a - N(t)) \cdots (3)
\]

Where \( b(t) \) is a fault detection rate and \( p \) is the probability of perfect debugging. So, in the above case fault detection rate reduces to \( Pb(t) \). In case of error generation the number of faults will increase therefore the fault detection rate decreases to \( p(1-\gamma)b(t) \)

Now the equation (3) can be written as

\[
\frac{dN(t)}{dt} = h(t)(a_1 - N(t)) \cdots (4)
\]

Where \( h(t) = p(1-\gamma)b(t) \) and \( a_1 = \frac{a}{1-\gamma} \)

It might happen that \( h(t) \) is not completely known, but subject to same random environment effects, such as the testing effort expenditure, the skill level of the testers and so, on and thus might have irregular fluctuations thus

\[
h(t) = p(1-\gamma)b(t) + \sigma y(t) \cdots (5)
\]

In above we don’t know the behavior of noise term only; its probability distribution is known. The functions \( h(t) \) is assumed to be non random.

Let \( \gamma(t) \) be a standard Gaussian white noise and “s” is a possible constant representing a magnitude of the irregular fluctuation. So, equation (5) can be written as:

\[
h(t) = p(1-\gamma)b(t) + s \gamma(t) \cdots (6)
\]

Hence, equation (4) can be written as:

\[
\frac{dN(t)}{dt} = p(1-\gamma)b(t)[a_1 - N(t)]dt + s[a_1 - N(t)]dW(t) \cdots (7)
\]

The equation (7) can be extended to the following stochastic equation of an Itô:

\[
\frac{dN(t)}{dt} = [p(1-\gamma)b(t) - \frac{1}{2}s^2]dt + s[a_1 - N(t)]dW(t) \cdots (8)
\]

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 Ito formula, solution to equation (8) using initial condition \( N(t=0)=0 \) we get \( N(t) \) as follows [1].

\[
N(t) = a_1[1-e^{-\frac{1}{b} \int_{0}^{t} \gamma(x)dx(t) + \gamma(t)}] \cdots (9)
\]

On solving the above equation

For simple faults:

\[
b(t) = \frac{a + \beta }{1 + bt} \quad \text{And} \quad a_1 = \frac{a}{1-\gamma} \cdots (10)
\]

We get

\[
N(t) = \frac{a}{1-\gamma}[1-(1+bt)e^{-\frac{1}{b} \int_{0}^{t} \gamma(x)dx(t) + \gamma(t)}] \cdots (11)
\]

Now,

\[
m(t) = E(N(t)) = \frac{a}{1-\gamma}[1-(1+bt)e^{-\frac{1}{b} \int_{0}^{t} \gamma(x)dx(t) + \gamma(t)}] \cdots (12)
\]

For hard fault:

It is assumed that hard fault detection/removal takes more time compared to simple faults detection/ remove [11]. Hence the removal process for two stage process and the differential equation for the failure and removal phenomenon are given as

\[
\frac{dm_f(t)}{dt} = b(a - m_f(t)) \cdots (13)
\]

\[
\frac{dm_r(t)}{dt} = b(t)(m_f(t) - m_r(t)) \cdots (14)
\]

\[
b(t) = \frac{b}{1 + \beta e^{-\beta t}} \cdots (15)
\]

The first stage of two stage process is given as by equation (13). The second stage of the two stage process is given by equation (14). During these stages the fault removal rate is assumed to be time dependent considering the learning effect of the debugging team.

Solving the above system of differential equation with \( m_f(t=0) = 0 \) and \( m_r(t=0) = 0 \)

The continuous srgm is given as
As follows:

\[ m_r(t) = a \left( 1 - (1 + bt) e^{-bt} \right) \cdots \]  

(16)

The removal phenomenon of the model can be derived in one stage directly assuming fault detection rate per remaining fault equals to

\[ b \left( \frac{1}{1 + be^{-bt}} - \frac{1}{1 + \beta + bt} \right) \cdots \]  

(17)

As follows

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

(18)

Equivalently, the differential equation describing the removal process under two types of imperfect debugging can be directly written as:

\[ \frac{dm_r(t)}{dt} = pb(t)(a(t) - m_r(t)) \cdots \]  

Where, \( b(t) = b \left( \frac{1}{1 + be^{-bt}} - \frac{1}{1 + \beta + bt} \right) \cdots \)  

(20)

And \( a(t) = a + am(t) \cdots \)  

(21)

Similarly, using the equation (19), (20), (21) and (9) we get

\[ N(t) = a_1(1 - e^{-\left( \frac{p(\beta + b)}{\sigma^2} \right) t}) \cdots \]  

(22)

\[ N(t) = a_1 \left(1 - \left(1 + \beta + bt \right) e^{-\left( p(\beta + b) + \sigma^2 \right) t} \right) \cdots \]  

(23)

Where, \( a_1 = \frac{a}{1 - \beta} \cdots \)  

For complex fault:

If removal of a fault after its detection involves even a greater time delay, it is classified as complex fault. Hence, they are modeled as three stage process as follows:

\[ \frac{dm_f(t)}{dt} = b(a - m_f(t)) \]  

\[ \frac{dm_u(t)}{dt} = b(m_f(t) - m_u(t)) \]  

\[ \frac{dm_r(t)}{dt} = b(t)(m_u(t) - m_r(t)) \]  

where, \( b(t) = \frac{b}{1 + be^{-bt}} \cdots \)  

(24)

The first stage process describes the failure observation process. The second stage process describes the fault isolation process. And the third stage process describes the fault removal process. During this stage the fault removal rate is assumed to be logistic learning function which represents the learning efficiency of testing team. Solving the above system of differential equation with initial conditions,

\[ m_f(t) = 0, m_u(t) = 0, m_r(t) = 0 \]

We get,

\[ m_r(t) = a \left( 1 - \left(1 + bt + \frac{b^2 t^2}{2} \right) e^{-bt} \right) \cdots \]  

(25)

Where, \( b(t) = b \left( \frac{1}{1 + be^{-bt}} - \frac{1}{1 + \beta + bt + \frac{b^2 t^2}{2}} \right) \cdots \)  

(26)

And,

\[ \frac{dm_r(t)}{dt} = b \left( 1 - \left(1 + bt + \frac{b^2 t^2}{2} \right) e^{-bt} \right) \cdots \]  

(27)

Equivalently, the differential equation for the removal process under two types of imperfect debugging can be directly written as:

\[ \frac{dm_r(t)}{dt} = pb(t)(a(t) - m_r(t)) \cdots \]  

Where,

\[ b(t) = b \left( \frac{1}{1 + be^{-bt}} - \frac{1 + bt}{1 + \beta + bt + \frac{b^2 t^2}{2}} \right) \cdots \]  

(29)

And,

\[ a(t) = a + am(t) \cdots \]  

(30)

Solving the above differential equation under initial condition \( m_r(t_0) = 0 \), we get,

\[ m(t) = \frac{a}{1 - \beta} \left(1 - \left(1 + \beta + bt + \frac{b^2 t^2}{2} \right) e^{-bt} \right) \cdots \]  

(31)

Similarly, using the equation (28),(29),(30) and (9), we get,

\[ N(t) = a_1(1 - e^{-\left( \frac{p(\beta + b)}{\sigma^2} \right) t}) \cdots \]  

(32)

\[ N(t) = a_1 \left(1 - \left(1 + \beta + bt + \frac{b^2 t^2}{2} \right) e^{-\left( p(\beta + b) + \sigma^2 \right) t} \right) \cdots \]  

(33)

3. Modeling fault removal process for multiple software release

3.1 Modeling for release-1:

In today’s scenario, up-gradation becomes vital part of any software company to be able to sustain in the market. Foundation and structure of software is represented in market in the first release of the software product [1]. Testing team at the release not only has to detect and remove fault as much as possible but also minimize the possibility of occurrence of errors in the future. This
model differentiates faults in three types of faults as simple, hard and complex. Some faults are removed as simple faults i.e. $p_1a_1^*F_{12}(t)$ and some faults are removed as hard faults i.e. $p_1' (1 - p_1')a_1^* F_{12}(t)$ and left faults are removed as complex faults $(1 - p_1 - p_1')a_1^* F_{13}(t)$

So,

$$M_1(t) = p_1a_1^* F_{13}(t) + p_1' a_1^* F_{12}(t) + (1 - p_1 - p_1')a_1^* F_{13}(t)$$

$$0 \leq t \leq t_1$$

Where $a_1^* = a_1 / 1 - \alpha_1$

$$F_{11}(t) = (1 - ((1 + \beta_1) / (1 + \beta_1 e^{-b_1t} \mu_1(1-\alpha_1) e^{-(\mu_1(1-\alpha_1)b_1t + \sigma_1^2t^2/2)})) \cdots \text{Simple faults}$$

$$F_{12}(t) = (1 - (((1 + \beta_1 + b_2t) / (1 + \beta_1 e^{-b_2t} \mu_1(1-\alpha_1) e^{-(\mu_1(1-\alpha_1)b_2t + \sigma_1^2t^2/2)})) \cdots \text{Hard faults}$$

$$F_{13}(t) = (1 - (((1 + \beta_1 + b_3t + b_3^2t^2/2) / (1 + \beta_1 e^{-b_3t} \mu_1(1-\alpha_1) e^{-(\mu_1(1-\alpha_1)b_3t + \sigma_1^2t^2/2)})) \cdots \text{Complex faults}$$

So, $F(t) = F_{11}(t) + F_{12}(t) + F_{13}(t)$

$$t_1 \leq t \leq t_2$$

Where $a_2^* = a_2 / 1 - \alpha_2$

$$F_{21}(t) = (1 - (((1 + \beta_2) / (1 + \beta_2 e^{-b_2t} \mu_2(1-\alpha_2) e^{-(\mu_2(1-\alpha_2)b_2t + \sigma_2^2t^2/2)})) \cdots \text{Simple faults}$$

$$F_{22}(t) = (1 - (((1 + \beta_2 + b_2t) / (1 + \beta_2 e^{-b_2t} \mu_2(1-\alpha_2) e^{-(\mu_2(1-\alpha_2)b_2t + \sigma_2^2t^2/2)})) \cdots \text{Hard faults}$$

$$F_{23}(t) = (1 - (((1 + \beta_2 + b_3t + b_3^2t^2/2) / (1 + \beta_2 e^{-b_3t} \mu_2(1-\alpha_2) e^{-(\mu_2(1-\alpha_2)b_3t + \sigma_2^2t^2/2)})) \cdots \text{Complex faults}$$

3.2 Modeling for release-2:

After first release, company came to know the result of first release by having information about the reported bugs from the users. Now to attract new customers company up-graded the existing software by adding new functions which leads to change in existing code and thus introduction of new errors. Now, the testing team starts testing the up-graded system. At this stage faults removed by testing team may be of previous release leftover or it may be of new introduced faults due to up-gradation. This model assumed that new faults which are introduced due to up-gradation are removed as simple faults i.e. $a_2^* F_{23}(t)$. In this paper we assumed that, some portion of left over simple faults i.e. $a_1^*(1 - F_{11}(t))$ interacts with new portion of hard detected faults i.e. $F_{22}(t - t_1)$ and some portion of left over simple faults i.e. $(1 - \lambda_2) a_1^*(1 - F_{11}(t))$ interacts with new portion of complex detected faults i.e. $F_{23}(t - t_1)$.

Similarly, some portion of left over hard faults i.e. $q_{21} a_1^*(1 - F_{12}(t))$ interacts with new portion of hard detected faults i.e. $F_{22}(t - t_1)$ and some portion of left over hard faults i.e. $(1 - q_{21}) a_1^*(1 - F_{12}(t))$ interacts with new portion of complex detected faults i.e. $F_{23}(t - t_1)$.

Similarly for the leftover complex faults i.e. $a_1^*(1 - F_{13}(t)) a_1^*(1 - F_{13}(t))$, interacts with new portion of complex detected faults i.e. $F_{23}(t - t_1)$.

$$M_2(t) = p_2 a_2^* F_{23}(t - t_1) + p_2 a_2^* F_{22}(t - t_1) + (1 - p_2 - p_2') a_2^* F_{23}(t - t_1) + (1 - p_2 - p_2') p_1 a_1^*(1 - F_{11}(t_1))$$
3.3 Modeling for release-3:

Similarly for release 3, leftover from 1st release interacts with new detection rate and similarly for leftover faults from 2nd release which also interacts with new detection rate. Some portion of leftover simple faults from 1st release, 
\[ a_t(1-F_{t1}(t))\{(1-\lambda_{3t}F_{23}(t-t_1)+(1-\lambda_{3t}F_{22}(t-t_1))\}
\]
Interacts with new hard fault detection rate i.e.
\[ F_{32}(t_2-t_1) \]
\[ a_t(1-F_{t1}(t))\{(1-\lambda_{3t}F_{21}(t-t_1)+(1-\lambda_{3t}F_{22}(t-t_1))\}\] 
\[ F_{32}(t_2-t_1) \]

Similarly, some portion of leftover simple faults from 1st release interacts with new complex fault detection rate i.e. 
\[ q_{33}a^{*}_t(1-F_{13}(t))\{(1-\lambda_{3t}F_{23}(t-t_1)+(1-\lambda_{3t}F_{22}(t-t_1))\}
\]
\[ (1-\lambda_{3t}F_{23}(t-t_1))F_{32}(t_2-t_1) \]

And
\[ (1-q_{33})a^{*}_t(1-F_{13}(t))\{(1-\lambda_{3t}F_{23}(t-t_1)+(1-\lambda_{3t}F_{22}(t-t_1))\}
\]
\[ (1-q_{33})F_{32}(t_2-t_1) \]

Similarly, leftover complex faults from 1st release interacts with new complex fault detection rate i.e.
\[ a^* (1-F_{13}(t))F_{33}(t_2-t_1) \]

Now some portion of leftover simple fault from 2nd release interacts new hard fault detection rate and some portion interacts with new complex fault detection rate i.e.

\[ \lambda_{32}a^*_t(1-F_{21}(t-t_1))F_{32}(t_2-t_1) \]

And
\[ (1-\lambda_{32})a^*_t(1-F_{21}(t-t_1))F_{32}(t_2-t_1) \]

In the same way, some portion of leftover hard fault from 2nd release interacts with new hard fault detection rate and some portion interacts with complex fault detection rate i.e.

\[ q_{32}a^*_t(1-F_{22}(t-t_1))F_{32}(t_2-t_1) \]

And
\[ (1-q_{32})a^*_t(1-F_{22}(t-t_1))F_{32}(t_2-t_1) \]

Similarly, leftover complex fault of 2nd release interacts with new complex fault detection rate i.e.

\[ a^*_t(1-F_{23}(t-t_1))F_{33}(t_2-t_1) \]

As the parameters are more in the proposed model in release -3 compare to no. of available data points in tandem data. So, we increase the data points by taking series mean of data points of available no. of faults detected in tandem data which is 37.92. Now,

\[ M_{i}(t) = p_{3i}a^*_tF_{31}(t-t_2)+p_{3i}a^*_tF_{32}(t-t_2) \]

\[ + (1-p_{3i}-p_{3i})a^*_tF_{33}(t-t_2)+\lambda_{3i}(p_{a}a^*_t(1-F_{1i}(t_1)) \]

\[ (1-\lambda_{31}F_{22}(t-t_1)+(1-\lambda_{31}F_{23}(t-t_1)))] \]

\[ F_{32}(t-t_2)+(1-\lambda_{31})(p_{a}a^*_t(1-F_{1i}(t_1})) \]

\[ (1-\lambda_{31}F_{22}(t-t_1)+(1-\lambda_{31}F_{23}(t-t_1)))] \]

\[ F_{33}(t-t_2)+(1-q_{31})(p_{a}a^*_t(1-F_{12}(t_i))) \]

\[ (1-q_{31}F_{32}(t-t_2)+(1-q_{31}F_{33}(t-t_2)))] \]

\[ F_{33}(t-t_2)+(1-p_{i}-p_{i})\lambda_{a}a^*_t(1-F_{33}(t_3)(t_3(t_3-t_2)))F_{33}(t-t_2) \]

\[ + \lambda_{32p}a^*_t(1-F_{13}(t_1))F_{32}(t-t_2) \]

\[ + (1-\lambda_{32})p_{a}a^*_t(1-F_{21}(t-t_1))F_{32}(t-t_2) \]

\[ + q_{32}a^*_t(1-F_{23}(t-t_1))F_{33}(t-t_2) \]

\[ + (1-\lambda_{32})a^*_t(1-F_{13}(t_1))F_{33}(t-t_2) \]

\[ + (1-p_{2i}-p_{2i})a^*_t(1-F_{23}(t_2-t_1))F_{33}(t-t_2) \]

\[ t_2 \leq t \leq t_3 \]
3.4 Modeling for release-4:

Similar for the release 4, leftover simple, hard and complex fault of the first release are removed with new detection rate. Leftover simple, hard and complex faults of second and third release are removed with new corrected/detected rate, including the new faults which are introduced due to up gradation are corrected by new correction/detection rate. Mathematical expression for release 4:

\[ M_4(t) = p_4 a_4^* F_4(t-t_4) + p_4' a_4^* F_4(t-t_4) \]

\[ + (1 - p_4) - p_4' a_4^* F_2(t-t_3) + (1 - p_4') a_4^* F_3(t-t_3) \]

\[ + (1 - \lambda_2) F_{32}(t-t_2) + (1 - \lambda_3) F_{32}(t-t_2) \}

\[ F_4(t-t_4) + q_4 \{ p_4 a_4^* (1 - F_{33}(t-t_3)) \} \}

\[ + (1 - q_4) \{ p_4' a_4^* (1 - F_{33}(t-t_3)) \} F_3(t-t_4) \]

\[ + (1 - p_4 - p_4') a_4^* (1 - F_{33}(t-t_3)) F_4(t-t_4) \]
4. 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 [14] 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 Table1, the performance analysis of proposed model is measured by the four common criteria that we define as below:

4.1 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, \( m_i \) represent the actual number of faults by time \( t_i \), and \( m_i(t_i) \) represent the estimated number of faults by time \( t_i \). In all mentioned criteria the lower value indicate less fitting error [4].

4.1.1 The Bias is defined as:

\[
\text{Bias} = \frac{1}{n} \sum_{k=1}^{n} (m(t_k) - m_k)
\]

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

4.1.2 The Variation is defined as:

\[
\text{Variance} = \frac{1}{n-1} \sum_{k=1}^{n} (m(t_k) - m_k - \text{Bias})^2
\]

The average of the prediction errors is called the prediction \( \text{Bias} \), and its standard deviation is often used as a measure of the variation in the predictions.

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

\[
\text{RMSPE} = \sqrt{\text{Variance}^2 + \text{Bias}^2}
\]

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

4.1.4 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:

\[
\text{MSE} = \frac{1}{n} \sum_{k=1}^{n} (m(t_k) - m_k)^2 (1 - 1)
\]

Where \( k \) is the number of observations. The lower MSE indicates less fitting error, thus better goodness of fit.

Table 1: Parameter Estimates

<table>
<thead>
<tr>
<th>i=1 to 4</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_i )</td>
<td>30.00</td>
<td>42.298</td>
<td>204.772</td>
<td>49.354</td>
</tr>
<tr>
<td>( p_i )</td>
<td>.290</td>
<td>.200</td>
<td>.200</td>
<td>.200</td>
</tr>
<tr>
<td>( p'_i )</td>
<td>.120</td>
<td>.010</td>
<td>.010</td>
<td>.140</td>
</tr>
<tr>
<td>( \beta_i )</td>
<td>1.000</td>
<td>1.075</td>
<td>1.000</td>
<td>1.765</td>
</tr>
<tr>
<td>( a'_{i1} )</td>
<td>7.000</td>
<td>3.516</td>
<td>.010</td>
<td>1.000</td>
</tr>
<tr>
<td>( a'_{i2} )</td>
<td>.909</td>
<td>.932</td>
<td>.030</td>
<td>.754</td>
</tr>
<tr>
<td>( a'_{i3} )</td>
<td>.139</td>
<td>.207</td>
<td>.100</td>
<td>.065</td>
</tr>
<tr>
<td>( a'_{i4} )</td>
<td>.000</td>
<td>.000</td>
<td>1.000E-006</td>
<td>.000</td>
</tr>
<tr>
<td>( b_{i1} )</td>
<td>.025</td>
<td>.010</td>
<td>.020</td>
<td>.058</td>
</tr>
<tr>
<td>( b_{i2} )</td>
<td>.033</td>
<td>.659</td>
<td>3.351</td>
<td>.422</td>
</tr>
<tr>
<td>( b_{i3} )</td>
<td>.429</td>
<td>.019</td>
<td>.020</td>
<td>.011</td>
</tr>
<tr>
<td>( \lambda'_{i1} )</td>
<td>-</td>
<td>.100</td>
<td>.100</td>
<td>.491</td>
</tr>
<tr>
<td>( \lambda'_{i2} )</td>
<td>-</td>
<td>-</td>
<td>.100</td>
<td>.100</td>
</tr>
<tr>
<td>( \lambda'_{i3} )</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>.087</td>
</tr>
<tr>
<td>( q_{i1} )</td>
<td>-</td>
<td>.100</td>
<td>.100</td>
<td>.300</td>
</tr>
<tr>
<td>( q_{i2} )</td>
<td>-</td>
<td>-</td>
<td>.100</td>
<td>.880</td>
</tr>
<tr>
<td>( q_{i3} )</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>.337</td>
</tr>
</tbody>
</table>

Table 2: Comparison Criteria

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R^2 )</td>
<td>.982</td>
<td>.994</td>
<td>.782</td>
<td>.996</td>
</tr>
<tr>
<td>Bias</td>
<td>-0.4357</td>
<td>-0.3551</td>
<td>0.0441</td>
<td>-0.0383</td>
</tr>
<tr>
<td>MSE</td>
<td>14.6268</td>
<td>7.4901</td>
<td>69.3028</td>
<td>0.895359</td>
</tr>
<tr>
<td>Variance</td>
<td>3.898313</td>
<td>2.78807</td>
<td>8.597713</td>
<td>0.896177</td>
</tr>
</tbody>
</table>
5. Graphs for Four Releases:

![Goodness of fit curve for release-1](image1)

![Goodness of fit curve for release-2](image2)

![Goodness of fit curve for release-3](image3)

![Goodness of fit curve for release-4](image4)

6. Conclusion:
In this paper we have developed the srgm incorporating stochastic differential equation of Itô type using different learning functions different severity of faults and two types of imperfect debugging. 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.

7. References:


