International Journal of Mathematical, Engineering and Management Sciences
Vol. 2, No. 3, 172–184, 2017
https://dx.doi.org/10.33889/IJMEMS.2017.2.3-015

Stochastic Formulation of Fault Severity Based Multi Release SRGM Using the Effect of Logistic Learning

Jagvinder Singh[1], Suneeta Bhati[2*], A. R. Prasanan[3], Ashok Vayas[4]

Department of Mathematics
[1,3]Maharaja Agersen College, University of Delhi, Delhi, India
[2,4]Jodhpur National University, Rajasthan, India
E-mails: [1]jagvinder.singh@gmail.com, [2*]suneetachaudhary2012@gmail.com, [3]arprasannan@gmail.com
[4]dr.ashokvyas@gmail.com
[*]*Corresponding author*

(Received December 18, 2016; Accepted February 2, 2017)

Abstract
In today's environment, software reliability is one of the major concerns for Software firms. Many Software Reliability Growth Model (SRGM) has been developed and many are under process. In order to meet the requirements of consumer and to excel in competitive environment, companies are coming up with multiple add –ons. We design the model as stochastic with continuous state space because of large software system, the count of failures observed is huge and so, the variation in count of errors detected/ removed in each debugging is petite compared to original error content at the beginning of testing. This study is an add on to the software reliability literature where we have developed multi release SRGM's based on available concept of depending on previous releases. The errors have been categorically divided upon the severity of their removal as one stage, two stage, three stage fault removal process is applied in an environment of irregular fluctuations.

Keywords- Software Reliability Growth Modeling (SRGM), Stochastic Differential Equation (SDE), Multi up-gradation, Learning effect, Severity of faults.

1. Introduction

In today's world, computer is vital part of our day to day activities. Since software is embedded in everything, hence the need of reliable software. The need for reliable software gives birth to software reliability engineering. Not only to enhance but also for longitivity and need of nowadays reliable complex systems,, motivates the researchers to design tools and techniques not only to evaluate software quantitatively but also estimate important measures such as software reliability , mean time to failure, the number of remaining faults , failure intensity as well while testing and operation phase and thus named as Software Reliability Growth Modeling (SRGM'S).Numerous SRGM's have been proponed and authenticated under various presumptions until now by ample of analyzers across world. An SRGM which dictates the fault detection process as NHPP is proponed by Goel and Okumoto (1979) which hypothesized that fault removal rate is proportional to remaining fault number. The conjunction between testing and the corresponding number of faults removed are either exponential, s- shaped or mix of two (Pham, 2006) is illustrated by plenty of SRGM's, in last two decades. Because of software augmentation at consumer's end the conventional software reliability growth models are unsuccessful to achieve the error growth. Consequently, a new up-graded version of software is announced in trade when software reaches a level where it acquires its operational reliability desired by the firm.

In order to maintain its rapport among its customers, renowned software firms like IBM, ADOBE, and WIPRO etc. are working tirelessly. Ergo, their R & D departments keep a close

International Journal of Mathematical, Engineering and Management Sciences
Vol. 2, No. 3, 172–184, 2017
https://dx.doi.org/10.33889/IJMEMS.2017.2.3-015

watch on the market strategies at consumers end and beget their presence in market by not only considering the errors from the present software but also by upgrading/add-ons i.e. by regularly adding some novel features in the present software, in the useful life phase. Introduction of new add-ons or features in present software are generated on the substratum of consumers requirement. While on each up-gradation, an increase in failure rate is acknowledged by software. Further failure rate decline steadily, on basis of errors found and corrected, after the up-gradation. Fig.1 given below, portrays the increment in failure rate on account of augmentation of novel functionalities in the software (Kapur et al., 2006a, 2006b, 2010a). The enhanced and present software may differentiate with respect to execution, interface and serviceability etc.

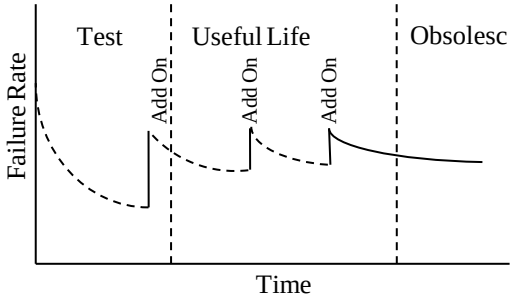

Fig.1. Failure rate curve due to feature enhancements for software systems

Instead of discrepancy in execution, interface and serviceability amid the enhanced and present software, the developers enhanced the software so as to ameliorate the software produced in spite of the worsening possibility of the up-graded version. In order to up-grade the present software, only chosen parts of the software system are altered whereas the others retain same to perform. Consequently, augments the error content which in result motivates the testing team to detect the errors in software. Secure enhancement not only reform the demeanors of the system but also sustain the market for firm; whereas vulnerable up gradation can create censorious faults in software. A multi release SRGM which states that multiplying errors in each generation not only based on all earlier releases but also, they are corrected with certitude was proposed by Kapur et al. (2006a). The proposed model is grounded on the presumptions that total error correction process of new release not only involved the reported errors from 1st release and onwards but also on the errors spawned as a result of amalgamation of novel features to the present software. Lastly, the knowledge acquired by the testing team while testing is incorporated in model in terms of learning factor and most importantly the errors are differentiated on the substratum of severity i.e. simple, hard and complex.

2. Assumptions

(i) The error identification / removal process are designed as non-homogenous process (NHPP).
(ii) The count of errors identified at any time is commensurate to the residual errors in the software.
(iii) Original count of errors is finite.
(iv) Errors identification process is designed as a stochastic process with a continuous state space.
(v) While testing the count of errors (simple, hard & complex) decline steadily.
(vi) Due to leftover errors in the system, software confronts the failure while execution.

International Journal of Mathematical, Engineering and Management Sciences
Vol. 2, No. 3, 172–184, 2017
https://dx.doi.org/10.33889/IJMEMS.2017.2.3-015

(vii) On account of effect required to correct errors, they are differentiated as simple, hard and complex.

(viii) Errors are debugged with certainty as no new errors are introduced while error isolation process.

3. Notations

$M(t)$	An arbitrary variable which is a count of errors identified while testing time t.
$Em(t)$	The mean value function or the predictable count of errors identified or corrected by time t.
$f(t)$	Possibility density function.
$F(T)$	Possibility distribution function.
$F_{ij}(t)$	Possibility distribution function for ith release and jth type of errors (i=1to4).
a	Original error content in the software.
$b(t)$	Error identification rate function which is time dependent.
$b_{ij}(t)$	Error identification rate for type (j=1to3) release in each release (i=1to 4).
σ_i	Positive constant which indicates the intensity of irregular fluctuation.
$\gamma(t)$	Standard Gaussian white noise.
β_i	Logistic learning constant for ith release (i=1 to 4).
t_i	Time for ith release (i= 1to 4).
p_i	Fragment of new simple errors induced in ith release corrected by new simple error correction rate.
p_i'	Fragment of new hard errors induced in ith release corrected by new hard error correction rate.
$(1-p_i-p_i')$	Fraction of new complex fault introduced in ith release removed by new complex fault removal rate.
λ_{ik}	Fraction of previous kth release simple fault removed by new ith release simple fault removal rate.
λ_{ik}'	Fraction of previous kth release simple fault removed by new ith release hard fault removal rate.
$(1-\lambda_{ik}-\lambda_{ik}')$	Fraction of previous kth release simple fault by new ith release complex fault removal rate.

Fraction of previous kth release hard fault removed by new ith release hard fault removal rate.

$(1-q_{ik})$	Fraction of previous kth release hard fault removed by new ith release complex fault removal rate.

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

International Journal of Mathematical, Engineering and Management Sciences
Vol. 2, No. 3, 172–184, 2017
https://dx.doi.org/10.33889/IJMEMS.2017.2.3-015

RMSPE Root Mean Square Prediction Error
FDR Fault Detection Rate

5. SDE Based Modeling of Up-Gradations for Each Release

An SRGM is created for four consecutive releases, portraying the software failure occurrence phenomenon while testing or operational phase by implementing stochastic and statistical theories. The mathematical equations designed portray the behavior of errors corrected while testing. It is presumed that novel features are amalgamated to the software while up-gradation for the first time and hence new code is written which in turn create new errors and are thus identified while testing. The count of errors eradicated while detection/correction process as compared to original error contents is abundantly petite which is a result of enlarge software system, consequently, portrays the randomness in error detection/correction procedure. We implement stochastic model of it^o type with continuous state space, so as to portrays the fluctuating nature of error detection process. Ample of SDE based SRGM's of *it^o* type have been designed by Yamada et al. (2003) such as, exponential, delayed S-shaped and inflection S-shaped stochastic differential equation models. A generalized SRGM on the substratum of SDE of *it^o* type which includes three distinct errors i.e. simple, hard and complex is proponed by Kapur et al. (2006b).

The number of errors detected in the software system up to testing time t is depicted by an arbitrary variable, called $\{m(t), t \geq 0\}$, where m(t) takes continuous real value. Software fault detection procedure is regarded as discrete state space by NHPP model while testing. The synonymous differential equation is given by (Kapur et al., 2010a; Kapur et al., 2010b; Kapur et al., 2006b; Pham and Zhang, 2003).

$$\frac{dm(t)}{dt} = \frac{f(t)}{1-F(t)}(a - m(t)) \tag{1}$$

Due to some random environment effects, incomplete rate is known, so that we have:

$$\text{r(t)} = \frac{f(t)}{1-F(t)} + \text{"}noise\text{"} \tag{2}$$

Where, r(t) is the time-based error identification/removal rate.

Let $\gamma(t)$ be a standard Gaussian white noise and σ be a positive constant depicting a intensity of the irregular fluctuations. So, the above equation can be penned as:

$$\frac{dm(t)}{dt} = \left[\frac{f(t)}{1-F(t)} + \sigma\gamma(t)\right](a - m(t)) \tag{3}$$

The overhead mathematical equation can be unfolded to the following, *ito* type stochastic differential equation:

International Journal of Mathematical, Engineering and Management Sciences
Vol. 2, No. 3, 172–184, 2017
https://dx.doi.org/10.33889/IJMEMS.2017.2.3-015

$$dm(t) = \left[\frac{f(t)}{1-F(t)} - \frac{\sigma^2}{2}\right](a-m(t))dt + \sigma(a-m(t))dW(t) \tag{4}$$

An integration of the white noise $\gamma(t)$ with respect to time t. is formally defined as one-dimensional Wiener process, $W(t)$ .Where, wiener process $W(t)$, is a Gaussian process which consists following properties:

$\Pr[W(0)=0]=1$
$E[W(t)]=0$
$E[W(t)W(t)]=\min[t,t]$

We get, $m(t)$, on implementing initial condition $m(0)=0$; as follows:

$$m(t) = a[1-(1-(F(t)))e^{-\sigma W(t)}] \tag{5}$$

Normal distribution is followed by Brownian motion or Wiener process, so, the density function (Singh et al., 2011) of w (t) is portrayed as:

$$f(w(t)) = \frac{1}{\sqrt{2\pi t}}\exp\left\{-\frac{(w(t))^2}{2t}\right\} \tag{6}$$

As a result the mean number of identified errors is (Singh et al., 2011) given as:

$$m^*(t) = E(m(t)) = a[1-(1-(F(t)))e^{\frac{\sigma^2 t}{2}}] \tag{7}$$

This paper involves the concept of three distinct detection rate in which simple errors are eradicated by exponential rate whereas hard and complex errors are detected/eradicated as two phase and three phase error eradication method, respectively. Also, three types of error identification/correction procedure incorporate the concept of learning factor which is the consequences of knowledge gained by testing team (Pham, 2006).

Simple errors are designed as:

$$\frac{dm(t)}{dt} = b(t)(a-m(t)) \tag{8}$$

$$b(t) = \frac{b}{(1+\beta e^{-bt})} \tag{9}$$

Hard errors are designed as two stage process (Singh et al., 2010):

International Journal of Mathematical, Engineering and Management Sciences
Vol. 2, No. 3, 172–184, 2017
https://dx.doi.org/10.33889/IJMEMS.2017.2.3-015

$$\frac{d(m_f(t))}{dt} = b(a - m(t) \tag{10}$$

$$\frac{d(m_r(t))}{dt} = b(t)(m_f(t) - m_r(t)), \tag{11}$$

$$b(t) = b \big/ (1 + \beta e^{-bt}) \tag{12}$$

Complex errors are designed as a three-stage process:

$$\frac{d(m_f(t))}{dt} = b(a - m(t) \tag{13}$$

$$\frac{d(m_i(t))}{dt} = b(m_f(t) - m_i(t)) \tag{14}$$

$$\frac{d(m_r(t))}{dt} = b(t)(m_f(t) - m_r(t)) \tag{15}$$

$$b(t) = b \big/ (1 + \beta e^{-bt}) \tag{16}$$

By applying original condition, m (0) =0 while solving overhead equations, we get m (t) as:

$$m(t) = \begin{cases} a\{(1-e^{-bt})/(1+\beta e^{-bt})\} & \ldots \text{ Simple errors} \\ a\{(1-(1+bt)e^{-bt})/(1+\beta e^{-bt})\} & \ldots \text{ Hard errors} \\ a\{(1-(1+bt+(b^2t^2/2))e^{-bt})/(1+\beta e^{-bt})\} & \ldots \text{ Complex errors} \end{cases} \tag{17}$$

Hence by utilizing the concept of $m(t) = aF(t)$ Kapur et al. (1995), $F(t)$, can be penned as:

$$F(t) = \begin{cases} \{(1-e^{-bt})/(1+\beta e^{-bt})\} & \ldots \text{Simple errors} \\ \{(1-(1+bt)e^{-bt})/(1+\beta e^{-bt})\} & \ldots \text{Hard errors} \\ \{(1-(1+bt+(b^2t^2/2))e^{-bt})/(1+\beta e^{-bt})\} & \ldots \text{Complex errors} \end{cases} \tag{18}$$

Using equation (7), (17) and (18), F (t) can be penned as:

$$F(t) = \begin{cases} \{(1-((1+\beta)/(1+\beta e^{-bt}))e^{(-bt+\frac{\sigma^2 t}{2})}\} & \ldots\ldots \text{Simple errors} \\ \{(1-((1+\beta+bt)/(1+\beta e^{-bt}))e^{(-bt+\frac{\sigma^2 t}{2})})\} & \ldots\ldots \text{Hard errors} \\ \{(1-((1+\beta+bt+\frac{b^2t^2}{2})/(1+\beta e^{-bt}))e^{(-bt+\frac{\sigma^2 t}{2})})\} & \ldots\ldots \text{Complex errors} \end{cases} \tag{19}$$

International Journal of Mathematical, Engineering and Management Sciences
Vol. 2, No. 3, 172–184, 2017
https://dx.doi.org/10.33889/IJMEMS.2017.2.3-015

6. Modeling of Four Releases

6.1 Release-1

In release 1, simple errors are extracted exponentially though hard faults by Yamada's identification/correction error rate and complex faults by Erlang method, incorporating the learning function in all three-error detection/correction rate.

$$M_1(t) = p_1 a_1 F_{11}(t) + p_1' a_1 F_{12}(t) + (1 - p_1 - p_1') a_1 F_{13}(t) \qquad 0 < t < t_1 \tag{20}$$

$$F_{1j}(t) = \begin{cases} \{1 - \{((1+\beta_1)/(1+\beta_1 e^{-b_{11}t}))e^{(-b_{11}t + \frac{\sigma_1^2 t}{2})}\}\} & \ldots \text{Simple errors} \\ \{1 - \{((1+\beta_1 + b_{12}t)/(1+\beta_1 e^{(-b_{12}t)}))e^{(-b_{12}t + \frac{\sigma_1^2 t}{2})}\}\} & \ldots \text{Hard errors} \\ \{(1 - \{((1+\beta_1 + b_{13}t + \frac{b_{13}^2 t^2}{2})/(1+\beta_1 e^{-b_{13}t}))e^{(-b_{13}t + \frac{\sigma_1^2 t}{2})}\}\} & \ldots \text{Complex errors} \end{cases} \tag{21}$$

6.2 Release-2

Amalgamation of novel features in the present software activates the alteration of the existing code which further increases the error content. While testing of newly formed code , the team not only eradicates certain errors (simple, hard ,complex) of previous release which are lying dormant and are not detected/ eradicated in previous release software version but also considers dependency and effect of subsumed new features with existing system. And the overall aloft procedure is accomplished before releasing the up-graded version of software in market. While, testing in this release, left over simple errors $p_1 a_1 (1 - F_{11}(t_1))$, left over hard errors $p_1 a_1 (1 - F_{12}(t_1))$ and left over complex errors $(1 - p_1 - p_1') a_1 (1 - F_{13}(t_1))$ of the first iteration interacts with novel simple, hard and complex detection /correction rate. A fragment λ_{21} of remaining simple errors from first version interacts with novel simple rate and a fraction λ_{21}' of remaining simple errors from first version interacts with novel hard rate whereas the remaining fraction $(1 - \lambda_{21} - \lambda_{21}')$ of errors from first version interacts novel detection/correction rate. Similarly, for the remaining hard errors $p_1 a_1 (1 - F_{12}(t_1))$ of the first iteration interacts with novel detection/ correction rate. A fragment of $q_1 p_1 a_1 (1 - F_{12}(t_1))$ interacts with novel hard rate and remaining fraction $(1 - q_1) p_1 a_1 (1 - F_{12}(t_1))$ with novel complex rate. Similarly, for the remaining complex errors from first iteration $(1 - p_1 - p_1') a_1 (1 - F_{13}(t_1))$ which is interacted with novel complex identification/ correction rate. Further, errors are created on account of amelioration of the features, a section of these errors are also eradicated while the testing with novel exposure rate i.e. $F_{21}(t - t_1)$ for simple errors and $F_{22}(t - t_1)$ for hard errors and $F_{23}(t - t_1)$ for complex errors. Variation in error detection rate is not only by virtue of alteration in time and testing strategies but also on account of metamorphosis of complexity in errors which is the result of amalgamation of novel functionalities. The resulting equation (Tickoo et al., 2015) can be designed as:

International Journal of Mathematical, Engineering and Management Sciences
Vol. 2, No. 3, 172–184, 2017
https://dx.doi.org/10.33889/IJMEMS.2017.2.3-015

$$M_2(t) = p_2 a_2 F_{21}(t) + p_2' a_2 F_{22}(t-t_1) + (1-p_2-p_2') a_2 F_{23}(t-t_1) + \lambda_{21} p_1 a_1 (1-F_{11}(t_1)) F_{21}(t-t_1)$$
$$+\lambda_{21}' p_1 a_1 (1-F_{11}(t_1) F_{22}(t-t_1) + (1-\lambda_{21}-\lambda_{21}') p_1 a_1 (1-F_{11}(t_1)) F_{23}(t-t_1)$$
$$+q_{21} p_1' a_1 (1-F_{12}(t_1)) F_{22}(t-t_1) + (1-q_{21}') p_1' a_1 (1-F_{12}(t_1)) F_{23}(t-t_1)$$
$$+(1-p_1-p_1') a_1 (1-F_{13}(t_1)) F_{23}(t-t_1)$$
$$t_1 < t < t_2 \qquad (22)$$

$$F_{ij}(t) = \begin{cases} \{1-\{((1+\beta_2)/(1+\beta_2 e^{-b_{21}t}))e^{(-b_{21}t+\frac{\sigma_2^2 t}{2})}\}\} & \ldots \text{Simple faults} \\ \{1-\{((1+\beta_2+b_{22}t)/(1+\beta_2 e^{-b_{22}t}))e^{(-b_{22}t+\frac{\sigma_2^2 t}{2})}\}\} & \ldots \text{Hard faults} \\ \{1-\{((1+\beta_2+b_{23}t+\frac{b_{23}^2 t^2}{2})/(1+\beta_2 e^{-b_{23}t}))e^{(-b_{23}t+\frac{\sigma_2^2 t}{2})}\}\} & \ldots \text{Complex fault} \end{cases} \qquad (23)$$

6.3 Release-3

Likewise, for release 3, we not only consider novel errors created in third release and but also the remnant errors (simple, hard and complex) from first and second release. 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. The proposed model for release -3 (Tickoo et al., 2015) can be designed as:

$$M_3(t) = p_3 a_3 F_{31}(t-t_2) + p_3' a_3 F_{32}(t-t_2) + (1-p_3-p_3') a_3 F_{33}(t-t_2)$$
$$+\lambda_{32} p_2 a_2 (1-F_{21}(t_2-t_1)) F_{31}(t-t_2) + \lambda_{32}' p_2 a_2 (1-F_{21}(t_2)) F_{32}(t-t_2)$$
$$+(1-\lambda_{32}-\lambda_{32}') p_2 a_2 (1-F_{21}(t_2-t_1)) F_{33}(t-t_2) +$$
$$+q_{32} p_2' a_2 (1-F_{22}(t_2-t_1)) F_{32}(t-t_2) + (1-q_{32}') p_2' a_2 (1-F_{22}(t_2-t_1)) F_{33}(t-t_2)$$
$$+(1-p_2-p_2') a_2 (1-F_{23}(t_2-t_1)) F_{33}(t-t_2) + \lambda_{31} p_1 a_1 (1-F_{11}(t_1)$$
$$(1-(\lambda_{21} F_{21}(t_2-t_1) + \lambda_{21}' F_{22}(t_2-t_1) + (1-\lambda_{21}-\lambda_{21}') F_{23}(t_2-t_1))) F_{31}(t-t_2)$$
$$+\lambda_{31}' p_1 a_1 (1-F_{11}(t_1))(1-(\lambda_{21} F_{21}(t_2-t_1) + \lambda_{21}' F_{22}(t_2-t_1) + (1-\lambda_{21}-\lambda_{21}') F_{23}(t_2-t_1))) F_{32}(t-t_2)$$
$$+(1-\lambda_{31}-\lambda_{31}') p_1 a_1 (1-F_{11}(t_1))(1-(\lambda_{21} F_{21}(t_2-t_1) + \lambda_{21}' F_{22}(t_2-t_1) + (1-\lambda_{21}-\lambda_{21}') F_{23}(t_2-t_1)))$$
$$F_{33}(t-t_2) + q_{31} p_1 a_1 (1-F_{12}(t_1)(1-(q_{21} F_{22}(t_2-t_1) + (1-q_{21}') F_{23}(t_2-t_1))) F_{32}(t-t_2)$$
$$+(1-q_{31}) p_1 a_1 (1-F_{12}(t_1)(1-(q_{21} F_{22}(t_2-t_1) + (1-q_{21}') p_1 a_1 F_{23}(t_2-t_1))) F_{33}(t-t_2)$$
$$+(1-p_1-p_1')(1-F_{13}(t_1))(1-F_{23}(t_2-t_1) F_{33}(t-t_2)$$
$$t_2 < t < t_3 \qquad (24)$$

$$F_{ij}(t) = \begin{cases} \{1-\{((1+\beta_3)/(1+\beta_3 e^{-b_{31}t}))e^{(-b_{31}t+\frac{\sigma_3^2 t}{2})}\}\} & \ldots \text{Simple errors} \\ \{1-\{((1+\beta_2+b_{22}t)/(1+\beta_2 e^{-b_{22}t}))e^{(-b_{22}t+\frac{\sigma_2^2 t}{2})}\}\} & \ldots \text{Hard errors} \\ \{1-\{((1+\beta_3+b_{33}t+\frac{b_{33}^2 t^2}{2})/(1+\beta_3 e^{-b_{33}t}))e^{(-b_{33}t+\frac{\sigma_3^2 t}{2})}\}\} & \ldots \text{Complex errors} \end{cases} \qquad (25)$$

International Journal of Mathematical, Engineering and Management Sciences
Vol. 2, No. 3, 172–184, 2017
https://dx.doi.org/10.33889/IJMEMS.2017.2.3-015

6.4 Release -4

Amalgamation of new functionalities is a continuous process which persists till that particular software is amid consumers. While testing and integration of code, this phenomenon helps in more and more errors detection/ eradication which firstly improves the quality and secondly ameliorate the reliability of software product under consideration. The new features are supplemented in the software for the third time and the resulting model for fourth release (Tickoo et al., 2015) is as follows:

$$
\begin{aligned}
&M_4(t) = p_4a_4F_{41}(t-t_3) + p_4'a_4F_{42}(t-t_3) + (1-p_4-p_4')a_4F_{43}(t-t_3) + \lambda_{43}p_3a_3\\
&(1-F_{31}(t_3-t_2))F_{41}(t-t_3) + \lambda_{43}'p_3a_3(1-F_{31}(t_3-t_2)F_{42}(t-t_3) + (1-\lambda_{43}-\lambda_{43}')\\
&p_3a_3(1-F_{31}(t_3-t_2))F_{43}(t-t_3) + q_{43}p_3'a_3(1-F_{32}(t_3-t_2))F_{42}(t-t_3) + (1-q_{43})\\
&p_3'a_3(1-F_{32}(t_3-t_2))F_{43}(t-t_3) + (1-p_3-p_3')a_3(1-F_{33}(t_3-t_2))F_{43}(t-t_3)\\
&+\lambda_{42}p_2a_2(1-F_{21}(t_2-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_{42}'p_2a_2(1-F_{21}(t_2-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_{42}(t-t_3) + (1-\lambda_{42}-\lambda_{42}')p_2a_2(1-F_{21}(t_2-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_{43}(t-t_3)\\
&+q_{42}p_2'a_2(1-F_{22}(t_2-t_1))(1-(q_{32}F_{32}(t_3-t_2) + (1-q_{32})F_{33}(t_3-t_2)))F_{42}(t-t_3)\\
&+(1-q_{42})p_2'a_2(1-F_{22}(t_2-t_1))(1-(q_{32}F_{32}(t_3-t_2) + (1-q_{32})F_{33}(t_3-t_2)))F_{43}(t-t_3)\\
&+(1-p_2-p_2')a_2(1-F_{23}(t_2-t_1))(1-F_{33}(t_3-t_2))F_{43}(t-t_3) + \lambda_{41}p_1a_1(1-F_{11}(t_1)\\
&(1-(\lambda_{21}F_{21}(t_2-t_1) + \lambda_{21}'F_{22}(t_2-t_1) + (1-\lambda_{21}-\lambda_{21}')F_{23}(t_2-t_1)))(1-(\lambda_{31}F_{31}(t_3-t_2)\\
&+\lambda_{31}'F_{32}(t_3-t_2) + (1-\lambda_{31}-\lambda_{31}')F_{33}(t_3-t_2)))F_{41}(t-t_3) + \lambda_{41}'p_1a_1(1-F_{11}(t_1))(1-(\lambda_{21}F\ (t_2-t_1)\\
&+\lambda_{21}'F_{22}(t_2-t_1) + (1-\lambda_{21}-\lambda_{21}')F_{23}(t_2-t_1)))(1-(\lambda_{31}F_{31}(t_3-t_2) + \lambda_{31}'F_{32}(t_3-t_2)\\
&+(1-\lambda_{31}-\lambda_{31}')F_{33}t_3-t_2)))F_{42}(t-t_3) + (1-\lambda_{41}-\lambda_{41}')p_1a_1(1-F_{11}(t_1)(1-(\lambda_{21}F_{21}(t_2-t_1)\\
&+\lambda_{21}'F_{22}(t_2-t_1) + (1-\lambda_{21}-\lambda_{21}')F_{23}(t_2-t_1)))(1-(\lambda_{31}F_{31}(t_3-t_2) + \lambda_{31}'F_{32}(t_3-t_2)\\
&+(1-\lambda_{31}-\lambda_{31}')F_{33}(t_3-t_2)) + q_{41}p_1'a_1(1-F_{11}(t_1))(1-(q_{21}F_{22}(t_2-t_1) + (1-q_{21}F_{23}(t_2-t_1)))\\
&(1-(q_{31}F_{32}(t_3-t_2) + (1-q_{31})F_{33}(t_3-t_2)F_{42}(t-t_3) + (1-q_{41})p_1'a_1(1-F_{11}(t_1))(1-(q_{21}F_{22}(t_2-t_1)\\
&+(1-q_{21}F_{23}(t_2-t_1)))(1-(q_{31}F_{32}(t_3-t_2) + (1-q_{31})F_{33}(t_3-t_2)F_{43}(t-t_3) + (1-p_1-p_1')(1-F_{13}(t_1))\\
&(1-F_{23}(t_2-t_1)(1-F_{33}(t_3-t_2))F_{43}(t-t_3)\\
&t_3 < t < t_4
\end{aligned}
\tag{26}
$$

$$
F_{ij}(t) = \begin{cases}
\{1-\{((1+\beta_4)/(1+\beta_4e^{-b_{41}t}))e^{\left(-b_{41}t+\frac{\sigma_4^2t}{2}\right)}\}\} & \ldots \text{Simple faults}\\
\{1-\{((1+\beta_4+b_{42}t)/(1+\beta_4e^{-b_{42}t}))e^{\left(-b_{42}t+\frac{\sigma_4^2t}{2}\right)}\}\} & \ldots \text{Hard faults}\\
\{1-\{((1+\beta_4+b_{43}t+\frac{b_{43}^2t^2}{2})/(1+\beta_4e^{-b_{43}t}))e^{\left(-b_{43}t+\frac{\sigma_4^2t}{2}\right)}\}\} & \ldots \text{Complex fault}
\end{cases}
\tag{27}
$$

International Journal of Mathematical, Engineering and Management Sciences
Vol. 2, No. 3, 172–184, 2017
https://dx.doi.org/10.33889/IJMEMS.2017.2.3-015

7. Model Validation, Data Set and Data Analysis

In order to illustrate the software reliability and to authenticate the proposed model, it is tested on tendem data for four releases. Further, for estimation of parameters, the non-linear least square technique is applied on proposed model. Table-1 and Table-2 portrays the estimated value of parameters and comparison criteria for four software releases, respectively. The performance analysis of proposed design is analyzed by the four common criteria i.e. Bias, Variation, Root Mean Square Prediction Error (RMPSE), Mean Square Error (MSE), on the substratum of data provided in Table1 as follow:

	I	II	III	IV
M.S.E	5.0955	3.295	0.532	0.6433
Bias	0.12	-0.024	0.158	.0222
Variation	2.130	1.865	0.744	0.824
R2	.994	.998	0.998	.997
RMSPE	2.130	3.786	0.760	0.824

Table 1. Comparison criteria

i=1to4	**1**	**2**	**3**	**4**
a_i	101.089	122.446	47.082	44.047
b_{i1}	11.832	14.509	7.471	1.034
b_{i2}	1.163	0.305	2.111	.272
b_{i3}	0.467	1.094	1.209	.419
p_i	0.184	0.107	.186	.131
p_i'	0.245	0.843	.150	.662
β_i	68.939	4.657	495.348	4.638
σ_i	0.002	0.073	.199	.090
λ_{i1}	-	0.14	0.110	.110
λ_{i1}'	-	0.12	0.120	.120
q_{i1}	-	0.121	0.300	0.300
λ_{i2}	-	-	0.120	0.110
λ_{i2}'	-	-	0.130	0.130
q_{i2}	-	-	0.494	0.300
λ_{i3}	-	-	-	0.110
λ_{i3}'	-	-	-	0.120
q_{i3}	-	-	-	0.180

Table 2. Parameter estimate

International Journal of Mathematical, Engineering and Management Sciences
Vol. 2, No. 3, 172–184, 2017
https://dx.doi.org/10.33889/IJMEMS.2017.2.3-015

Fig 2, 3, 4, 5 depicts the graphical representation of the actual and predicted values goodness of fit curve for the release 1

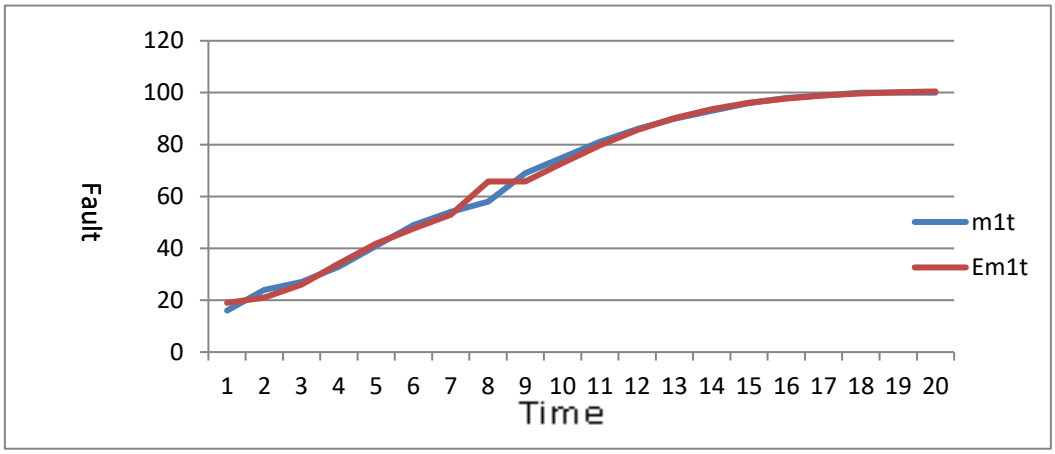

Fig. 2. Goodness of fit curve for release-1

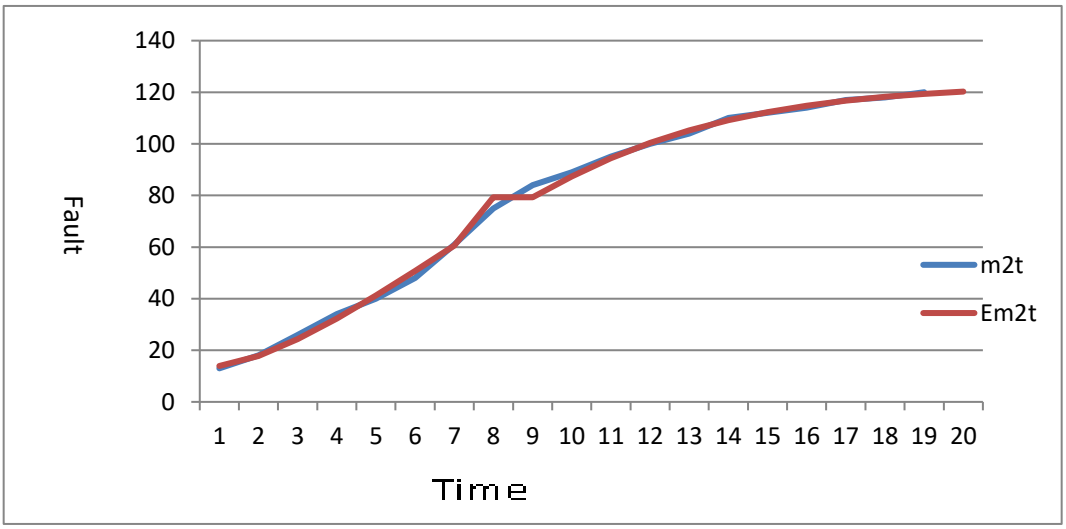

Fig. 3. Goodness of fit curve for the release 2

International Journal of Mathematical, Engineering and Management Sciences
Vol. 2, No. 3, 172–184, 2017
https://dx.doi.org/10.33889/IJMEMS.2017.2.3-015

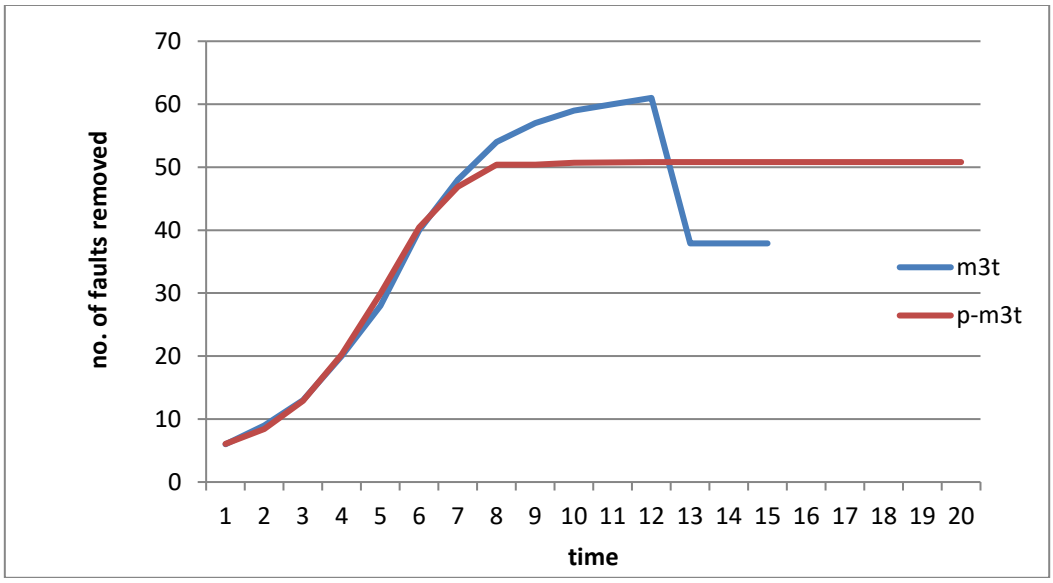

Fig. 4. Goodness of fit curve for the release 3

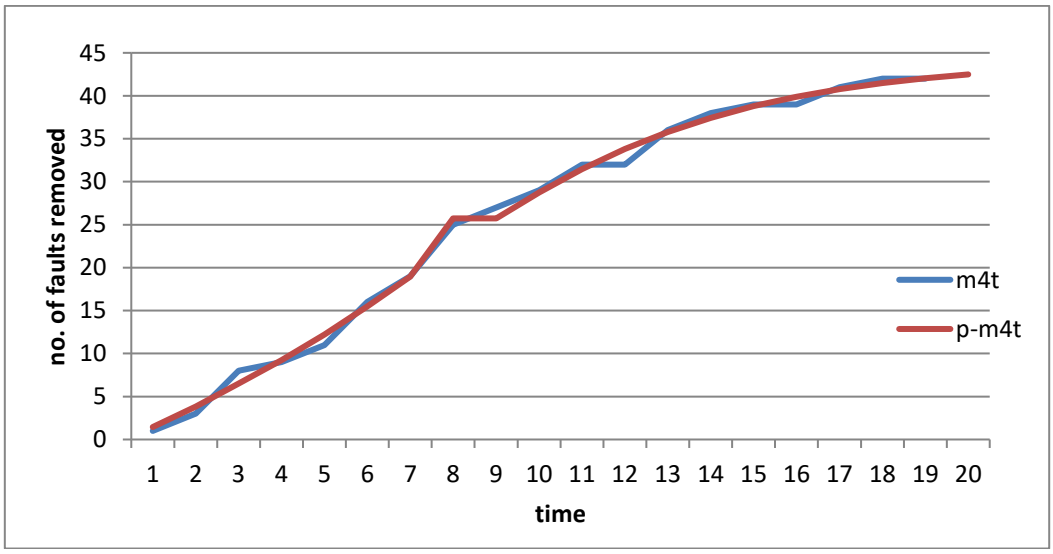

Fig. 5. Goodness of fit curve for the release 4

8. Conclusion

This paper suggested a noise based multi up-gradation model with fault severity of three types and incorporating the logistic learning function. Since, each software is hindered by bugs, so, software testing is imperative, which is an on-going process. Amid testing of newly created code, there is a possibility of detection/ correction of errors of previously released software which further enhances the reliability of software product under consideration. The suggested multi up gradation model is estimated on a four-release data set. In future, we can implement the proposed

model on n- types of errors. This proposal of models has, in fact, paved way for future research in other interdisciplinary fields of software reliability.

References

Goel, A. L., & Okumoto, K. (1979). Time-dependent error-detection rate model for software reliability and other performance measures. *IEEE Transactions on Reliability*, *28*(3), 206-211.

Kapur, P. K., Anand. S., Yadavalli, V. S. S., & Beichelt. F. (2006b). A generalized software growth model using SDE. In *International Conference on Quality, Reliability and Infocom Technology, ICQRIT*, (pp. 451-461).

Kapur, P. K., Basirzadeh, M., Inoue, S., & Yamada, S. (2010b). Stochastic differential equation based SRGM for errors of different severity with testing-effort. *International Journal of Reliability, Quality and Safety Engineering*, *17*(03), 179-197.

Kapur, P. K., Gupta, A., & Jha, P. C. (2006a). On modeling failure phenomenon of multiple release of a software in operational phase for product and project type software: a theoretical approach. In *Proceedings of International Conference on Quality, Reliability and Infocom Technology*, 55-164.

Kapur, P. K., Tandon, A., & Kaur, G. (2010a, December). Multi up-gradation software reliability model. In *Reliability, Safety and Hazard (ICRESH)*, 2010 2nd International Conference on (pp. 468-474). IEEE.

Kapur, P. K., Younes, S., & Agarwala, S. (1995). Generalised Erlang model with n types of faults. *ASOR Bulletin*, *14*(1), 5-11.

Pham, H. (2006). *System software reliability*. Springer Science & Business Media.

Pham, H., & Zhang, X. (2003). NHPP software reliability and cost models with testing coverage. *European Journal of Operational Research*, *145*(2), 443-454.

Singh O., Kapur P. K., & Anand, A. (2011). A stochastic formulation of successive software releases with faults severity, In *International Conference on Industrial Engineering and Engineering Management (IEEM)*, (pp. 136-140), IEEE.

Singh, O., Kapur, P. K., & Singh, J. (2010). Multi up-gradation software reliability growth model with learning effect and severity of faults using SDE. In *International Conference on Development and Applications of Statistics in Emerging Areas of Science and Technology, ICDASEAST.*

Tickoo, A., Kapur, P. K., & Khatri, S. K. (2015, February). Developing software reliability growth model for multi up gradations with faults of different severity and related release time problem. In *Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE), 2015 International Conference on* (pp. 376-382). IEEE.

Yamada, S., Nishigaki, A., & Kimura, M. (2003). A stochastic differential equation model for software reliability assessment and its goodness of fit. *International Journal of Reliability and Applications*, *4*(1), 1-11.