Mathematical Modeling of Software Bug Complexity

Vir Bahadur Singh[1], Meera Sharma [2], Sujata Khatri[3] ,Om Sharan Srivastava[4]
[1] Delhi College of Arts & Commerce, University of Delhi, Delhi India
[2] Swami Shraddhanand College, University of Delhi, India
[3] DDU College, University of Delhi, Delhi, India
[4] Delhi University Computer Centre University of Delhi, Delhi India

***Abstract*:** During testing of software, most of the bugs lying dormant in the software gets uncovered once the test cases are executed. Different bugs may take different amounts of effort and expertise for their removal. To understand the complexity of bugs from a developer's perspective, researchers have developed different mathematical models. Software consists of two types of bugs, dependent and independent. Dependent bugs are those whose removal depends upon the removal of some other bugs on which it is dependent. Dependency of bugs also makes the bug complex and bugs will take more time during fixing. Different debugging time lags functions have been taken to model different complexity of bugs. The aim of this paper is to study the bugs of different complexity. The complexity of bugs has been also modeled using dependency concept. Testing effort dependent bug complexity model using fault dependency has been also discussed. We also feel that that more complex bug will take more time and less complex bug will take less time during fixing. During removal of bugs, the removal team gets more familiar with the code during the fixing. The learning effect during testing has been incorporated using logistic removal rate. The models are validated based on different comparison criteria namely MSE, R^2 , Bias, Variation and Root mean squared error.

***Keywords/Index Terms*:** Non-homogeneous Poisson process, bug complexity, bugs types.

1. INTRODUCTION

During last four decades software reliability growth models have been used to measure the reliability growth and testing progress of software. (Musa 1987) defined software reliability as probability of failure free operation of the software in a fixed environment for a fixed interval of time. The fitting of software reliability growth models also depends upon the nature of failure data which is either of exponential or S-shaped nature or mixed of the two. On the same line, software reliability growth models that measure reliability growth of the software belongs to either of the two categories - Exponential and S-shaped. In software simple bugs are detected and removed in an exponential fashion. In an exponential growth curve, it is assumed that the error removal intensity is linearly related to the remaining number of software errors. The causes of S-shapedness are many and have been discussed by (Yamada, Ohba and Osaki 1984, Ohba 1984, Bittanti, Bolzern, Pedrotti and Scattolini 1988) and others. Software team which deals the removal process also learn and their skill grow. In reality, it has been observed that any software system may contain different types of bugs. Some bug affects the functionality of software and users gives different levels of severity which ranges from cosmetics, minor , major and critical. These bugs may take different amounts of time for their removal. When a developer wants to fix the bugs, the bugs may take less time or more time and it only depends upon the complexity of bugs. We define the complexity of bugs in terms of time taken during removal. Bugs are detected , isolated then removed or fixed. Some bugs take more time while others takes less time. This is a question why a bug takes more time or less time. And the answer is it depends how the bug is complex means what is their dependency, environmental impact and link to another module or functions. We need to treat different complexity of bugs with different strategy. A study in bugs complexity and their categorization is done by many researchers [(Obha 1984, (Yamada, Osaki and Narithisa 1985), (Kimura, Yamada and Osaki 1992) . Different types of growth models have been developed, namely, exponential, hyper exponential , exponential S-shaped model to capture different types of bugs present in the software.

Of late different researchers worked in this area and touched the depth concept of bug complexity. (Kapur, Younes and Agarwala 1995) developed a generalized Erlang SRGM to classify the bugs in the software system as simple, hard and complex with the assumption that the time delay between the failure observation and its removal represent the complexity of bugs. Implicit categorization of faults based on the time of detection of the fault has been discussed by (Kapur, Bardhan and Kumar 2000). Now, it is clear from literature survey that an SRGM should explicitly define the different types of faults due to the fact that any type of fault can be detected at any point of testing time. Thus, it is clear that modeling the bug complexity will help in resource management and provides an ease to project managers during testing or in the operational phase. In this paper a study has been conducted on mathematical modeling of software bug complexity which categories bugs into different categories depending upon the time they take

in fixing. Bug complexity has been defined by considering the time they take means failure detection and removal, the learning of the testing team, bug dependency and testing effort with bug dependency. Various debugging time lags functions have been taken for different types of bugs

It has been assumed that simple faults can be independently removed (termed as leading faults) and debugging time lag assumed to be negligible. Hard faults take more time in fixing due to the fact that it is assumed that they are dependent faults, whose removal is dependent on the removal of some leading faults. Hard faults cannot be immediately removed but lag the fault detection process by a debugging time lag / delay effect factor $\varphi(t)$. Complex faults takes more time in removal as they are more dependent on other bugs and also assumed to be dependent faults, whose removal is also dependent on the removal of some leading faults. Complex faults need more debugging time lag than hard bugs by a debugging time lag / delay effect factor $\varphi(t)$. But, in the case of complex faults debugging time lag is more in comparison to the hard faults.

2. Basic Assumption

An SRGM based on NHPP can be formulated as a Poisson process:

$$\Pr\{N(t)=n\}=\frac{(m(t))^n}{n!}\exp(-m(t)), \quad n=0,1,2 \quad (1)$$

And $$m(t)=\int_0^t \lambda(x)dx$$

Here, the expected no. of bugs is $N(t)$, and mean value function is $m(t)$ The intensity function λ(*x*) (or the mean value function *m*(*t*)) is the basic building block of all the NHPP models existing in the software reliability engineering literature.

1. The non homogeneous poison process has been used to model the bug detection /removal phenomenon.
2. Remaining bugs lying dormant in the software cause failures.
3. n types of bugs existing in the software and each type of bug is modeled by a different growth curve.
4. Each time when a failure is observed, an immediate (delayed) effort is taken to decide the cause of the failure and remove the corresponding bug.
5. No bug generation has been considered in the paper.
6. The bug removed in $(t,t+\Delta t)$ is proportional to the expected number of bugs remaining to be removed.

3. Software Reliability Growth Modeling

In this section we study software reliability growth model, which determines the types of bug and their proportion present in the software based on their complexity. We have also discussed the logistic removal rate for different types of bugs present in the software system. Dependency based models have been also discussed.

3.1. Generalized Erlang Software Reliability Growth Model[Kapur et al. 1995]

This model is a generalized model and provides the proportion of different types of bugs lying in the software Kapur et al.[9] . It is also assumed that different types of bugs exists in the software and may take different amount of time and follows different growth curves.

$$m(t)=\sum_{i=1}^{n} a_i\left[1-\exp(-b_i t)\left[\sum_{j=0}^{i-1}\frac{(b_i t)^j}{j!}\right]\right] \quad (2)$$

$m(t)$:mean value function of the expected number of detected/removed bugs the time interval $[0,t]$

i : Type of bug

p_i : proportion of type i bugs

$a_i\,(=ap_i)$: initial content of type i bug

$m_{if}(t)$: mean numbers of failure caused by i bugs in time t.

$m_{iis}(t)$: mean number of type i bugs isolated in time t

$m_i(t)$: mean number of type i bugs removed in time t.

b_1 $d_2(t), d_2(t)$: bug removal rate per bug for type 1, type 2, and type 3.

β_i : constant (for i =2 to n type of bugs)

j : the number of stages required to remove the bug after its failure observation/bug detection (j is dependent upon the type of bug)

GE-n : model with n type of bugs.

3.2. Modeling Complexity of Bugs by Considering **Learning [Singh V.B.2008]**

Different types of bugs are depicted by different types of curve. Here assumption is removal growth of type 1 bug which is simple in nature follows exponential curve. For other bugs, which are more, sever in nature, we incorporate logistic learning during removal phenomenon and these bugs are depicted by different types of S-shaped curves. In the beginning, we assume that only three types of bugs exist in software type 1, type 3 and type 3 (simple, hard and complex namely) and later, we extend our modeling to n types of bug.

Assuming a_1, a_2 and a_3 to be simple, hard and complex bugs in a software system ($a_1+a_2+a_3=a$), the simple bug removal process is modeled by the following

$$\frac{dm_1(t)}{dt} = b_1\left(a_1-m_1(t)\right) \quad (3)$$

$m_1(t)$ is the number of simple bugs removed.Solving equation (3) with the initial condition $m_1(0)=0$, we get:

$$m_1(t)=a_1\left(1-\exp\left(-b_1t\right)\right) \quad (4)$$

The hard bugs removal process is modeled as a two-stage process,

$$\frac{dm_{2f}(t)}{dt} = b_2\ (a_2 - m_{2f}(t) \quad (5)$$

$$\frac{dm_2(t)}{dt} = \frac{b_2}{1+\beta_2\exp\left(-b_2t\right)}\left(m_{2f}(t)-m_2(t)\right) \quad (6)$$

Here we assume that learning of removal team grows as testing progresses and follows logistic removal rate.

Solving equation (5) and (6) with the initial condition $m_{2f}(0)=0, m_2(0)=0$, we get:

$$m_2(t)=a_2\frac{\left(1-\left(1+b_2t\right)\exp\left(-b_2t\right)\right)}{\left(1+\beta_2\exp\left(-b_2t\right)\right)} \quad (7)$$

Here $m_{2f}(t)$ denotes the number of failures observed in time *t* whereas $m_2(t)$ represents the number of bugs removed in time *t*.

The complex bug removal process is modeled as a three-stage process,

$$\frac{dm_{3f}(t)}{dt} = b_3\ (a_3 - m_{3f}(t) \quad (8)$$

$$\frac{dm_{3is}(t)}{dt} = b_3\ (m_{3f}(t)-m_{3is}(t) \quad (9)$$

$$\frac{dm_3(t)}{dt} = \frac{b_3}{1+\beta_3\exp\left(-b_3t\right)}(m_{3is}(t)-m_3(t) \quad (10)$$

In a practical scenario, the removal rate follows logistic as learning of the removal team grows as testing progresses and Solving equation (8), (9) and (10) with the initial condition $m_{3f}(0)=0, m_{3is}(0)=0$ *and* $m_3(0)=0$, we get:

$$m_3(t)=a_3\frac{\left(1-\left(1+b_3t+\frac{b_3{}^2t^2}{2}\right)\exp\left(-b_3t\right)\right)}{\left(1+\beta_3\exp\left(-b_3t\right)\right)} \quad (11)$$

Here, $m_2(t)$ and $m_3(t)$ are expressed by delayed S-shaped and 3-stage Erlang growth curves with logistic removal rates. The removal rates for simple , hard and complex bugs are given as b_1 ,

$d_2 = b_2\left[\frac{1}{\left(1+\beta_2\exp\left(-b_2t\right)\right)}-\frac{1}{\left(1+\beta_2+b_2t\right)}\right]$ and

$d_3 = b_3\left[\frac{1}{\left(1+\beta_3\exp\left(-b_3t\right)\right)}-\frac{\left(1+b_3t\right)}{\left(1+\beta_3+b_3t+b^2{}_3t^2\right)}\right]$

respectively.

It is seen that b_2 and b_3 equal to b_1 in long run. The removal rates for three types of bugs become b_1

$d_2 = b_1\left[\frac{1}{\left(1+\beta_2\exp\left(-b_1t\right)\right)}-\frac{1}{\left(1+\beta_2+b_1t\right)}\right]$,and

$d_3 = b_1\left[\frac{1}{\left(1+\beta_3\exp\left(-b_1t\right)\right)}-\frac{\left(1+b_1t\right)}{\left(1+\beta_3+b_1t+b^2{}_1t^2\right)}\right]$ respectively.

We also note that

$$b_1 > b_1\left[\frac{1}{\left(1+\beta_2\exp\left(-b_1t\right)\right)}-\frac{1}{\left(1+\beta_2+b_1t\right)}\right] > b_1\left[\frac{1}{\left(1+\beta_3\exp\left(-b_1t\right)\right)}-\frac{\left(1+b_1t\right)}{\left(1+\beta_3+b_1t+b^2{}_1t^2\right)}\right]$$

, Which is in accordance with the severity of bugs.

The mean value function of the proposed SRGM is

$$m(t)=m_1(t)+m_2(t)+m_3(t)$$

Where $m_1(t)$ is the mean value function of the simple bugs removed in time $[0,t]$, $m_2(t)$ is the mean value function of the hard bugs removed in time $[0,t]$ and $m_3(t)$ is the mean value function of the complex bugs removed in time $[0,t]$.

$$m(t)=a_1\left(1-\exp\left(-b_1t\right)\right)+a_2\frac{\left(1-\left(1+b_2t\right)\exp\left(-b_2t\right)\right)}{\left(1+\beta_2\exp\left(-b_2t\right)\right)}+a_3\frac{\left(1-\left(1+b_3t+\frac{b_3^2t^2}{2}\right)\exp\left(-b_3t\right)\right)}{\left(1+\beta_3\exp\left(-b_3t\right)\right)} \quad (12)$$

Assuming $b_1=b_2=b_3=b,$ we have

$$m(t)=a_1\left(1-\exp\left(-bt\right)\right)+a_2\frac{\left(1-\left(1+bt\right)\exp\left(-bt\right)\right)}{\left(1+\beta_2\exp\left(-bt\right)\right)}+a_3\frac{\left(1-\left(1+bt+\frac{b^2t^2}{2}\right)\exp\left(-bt\right)\right)}{\left(1+\beta_3\exp\left(-bt\right)\right)} \quad (13)$$

Assuming

$$a_1=ap_1,\ a_2=ap_2\ and\ a_3=a\left(1-p_1-p_2\right)$$

$$m(t)=ap_1\left(1-\exp\left(-bt\right)\right)+ap_2\frac{\left(1-\left(1+bt\right)\exp\left(-bt\right)\right)}{\left(1+\beta_2\exp\left(-bt\right)\right)}+a\left(1-p_1-p_2\right)\frac{\left(1-\left(1+bt+\frac{b^2t^2}{2}\right)\exp\left(-bt\right)\right)}{\left(1+\beta_3\exp\left(-bt\right)\right)} \quad (14)$$

The model described above can be generalized to n different types of bugs depending upon their severity.

$$m(t)=\sum_{i=1}^{n}m_i(t)=a_1\left(1-\exp\left(-b_1t\right)\right)+\sum_{i=2}^{n}\frac{a_i}{\left(1+\beta_i\exp\left(-b_it\right)\right)}\left[1-\exp\left(-b_it\right)\left[\sum_{j=0}^{i-1}\frac{\left(b_it\right)^j}{j!}\right]\right] \quad (15)$$

Model described in above equation determines the type of bugs present in a software with a logistic removal rate and is abbreviated as GE-n (Logistic). If β_i =0 , above model reduces to equation (2).

Assuming $b_1=b_2=b_3=\ldots=b_n=b$. We also assume that the value of β remaining same for different types of bug from estimation view. We have

$$m(t)=a_1\left(1-\exp\left(-b_1t\right)\right)+\frac{1}{\left(1+\beta\exp\left(-bt\right)\right)}\sum_{i=2}^{n}a_i\left[1-\exp\left(-bt\right)\left[\sum_{j=0}^{i-1}\frac{\left(bt\right)^j}{j!}\right]\right] \quad (16)$$

3.3. Modeling Complexity of Bugs by Considering Dependency [Singh V.B.2008]

By considering different types of bugs lying dormant in the software, we can write

$$a = a_1 + a_2 + a_3 \qquad (17)$$

Where a_1, a_2 and a_3 are the initial contents of simple, hard and complex faults respectively.

Let $m(t)$ represents the mean number of bugs removed in time $[t, t+\Delta t]$. The value of $m(t)$ can be written as the superposition of three NHPP to incorporate the removal of simple, hard and complex bugs.

$$m(t) = m_1(t) + m_2(t) + m_3(t) \qquad (18)$$

Where $m_1(t)$, $m_2(t)$ and $m_3(t)$ are the mean value function of the simple, hard and complex faults removed in time $[0, t]$.

Modeling Simple Faults

Simple bugs are considered as independent bugs and the following differential equation can be written to deal simple bugs:

$$\frac{dm_1(t)}{dt} = b_1 \times \left[a_1 - m_1(t)\right] \qquad (19)$$

Solving equation (19) under the boundary condition i.e. at $t = 0, m_1(0) = 0$, we have

$$m_1(t) = a_1\left(1 - \exp(-b_1 t)\right) \qquad (20)$$

Equation (20) models the simple fault removal phenomenon..

Modeling Hard Faults

In case of hard bugs, the debugging time lag can be more and is expressed in the following differential equation

$$\frac{dm_2(t)}{dt} = b_2 \times \left[a_2 - m_2(t)\right] \times \frac{m_1\left(t - \phi(t)\right)}{a_1} \qquad (21)$$

Here, we define debugging time lag $\phi(t) = \frac{1}{b_1}\log(1 + b_1 t)$

Solving equation (21) with boundary condition $t = 0, m_2(0) = 0$, we obtain $m_2(t)$ as

$$m_2(t) = a_2\left(1 - \exp\left[\frac{2b_2}{b_1}\left(1 - \exp[-b_1 t]\right) - tb_2\left(1 + \exp[-b_1 t]\right)\right]\right) \qquad (22)$$

Modeling Complex Faults

In case of complex bugs the time lag will be more than the hard bug because it needs detection, isolation and removal. It is expressed in the following equation

$$\frac{dm_3(t)}{dt} = b_3 \times \left[a_3 - m_3(t)\right] \times \frac{m_1\left(t - \phi(t)\right)}{a_1} \qquad (23)$$

For complex faults debugging time lag is more than the hard faults. We define the debugging time lag as follows:

$$\phi(t) = \left[\frac{1}{b_1}\log\left(1 + b_1 t + \frac{b_1^2 t^2}{2}\right)\right]$$

Solving equation (23) with boundary condition $t = 0, m_3(0) = 0$, we obtain $m_3(t)$ as

$$m_3(t) = a_3\left(1 - \exp\left[\begin{array}{l}\frac{3b_3}{b_1}\left(1 - \left(1 + b_1 t\right)\exp\left[-b_1 t\right]\right) \\ -tb_3\left(1 - \left(1 - \frac{b_1 t}{2}\right)\exp\left[-b_1 t\right]\right)\end{array}\right]\right) \qquad (24)$$

From equation (18), we get

$$m(t) = a_1\left(1 - \exp(-b_1 t)\right) + a_2\left(1 - \exp\left[\frac{2b_2}{b_1}\left(1 - \exp[-b_1 t]\right) - tb_2\left(1 + \exp[-b_1 t]\right)\right]\right) + a_3\left(1 - \exp\left[\begin{array}{l}\frac{3b_3}{b_1}\left(1 - \left(1 + b_1 t\right)\exp\left[-b_1 t\right]\right) \\ -tb_3\left(1 - \left(1 - \frac{b_1 t}{2}\right)\exp\left[-b_1 t\right]\right)\end{array}\right]\right) \qquad (25)$$

Where $a_1 = ap_1$, $a_2 = ap_2$ and $a_3 = ap_3$ where $p_3 = (1 - p_1 - p_2)$

3.4. Modeling Complexity of Bugs by Considering Testing Effort and Bug Dependency [Singh V.B.2008]

Testing effort plays an important role during testing of software. In this section , we have discussed the modeling of bug complexity by considering the bug dependency and testing effort. The following equation expresses the removal of different types of bugs namely simple, hard and complex bugs.

Modeling Simple Faults:

The simple bugs which are considered as independent can be expressed in the following equation.

$$\frac{dm_1(t)}{dt} \times \frac{1}{w(t)} = b_1 \times \left[a_1 - m_1(t)\right] \qquad (26)$$

Solving equation (26) under the boundary condition i.e. at $t=0, m_1(0)=0, W(t=0)=W(0)$ we have

$$m_1(t)=a_1\left(1-\exp\left(-b_1W^*(t)\right)\right) \qquad (27)$$

here $W^*(t)=W(t)-W(0)$

Modeling Hard Faults:

For hard faults that are dependent and can be removed upon the removal of some leading faults with a debugging time $\Delta W(t)$, we have the following differential equation:

$$\frac{dm_2(t)}{dt}\times\frac{1}{w(t)}=b_2\times\left[a_2-m_2(t)\right]\times\frac{m_1\left(W(t)-\Delta W(t)\right)}{a_1} \qquad (28)$$

here, we define debugging time lag

$$\Delta W(t)=\frac{1}{b_1}\log\left(1+b_1W^*(t)\right)$$

Solving equation (28) with boundary condition $t=0, m_2(0)=0, W(t=0)=W(0)$ we obtain $m_2(t)$ as

$$m_2(t)=a_2\left(1-\exp\left[\frac{2b_2}{b_1}\left(1-\exp\left[-b_1W^*(t)\right]\right)-W^*(t)b_2\left(1+\exp\left[-b_1W^*(t)\right]\right)\right]\right) \qquad (29)$$

here, $w^*(t)=W(t)-W(0)$

Modeling Complex Faults:

The complex faults which are very difficult to detect and remove is based on the assumption (8) that these faults are dependent and can be removed upon the removal of some leading faults with a debugging time lag $\Delta W(t)$, we have the following differential equation:

$$\frac{dm_3(t)}{dt}\times\frac{1}{w(t)}=b_3\times\left[a_3-m_3(t)\right]\times\frac{m_1\left(W(t)-\Delta W(t)\right)}{a_1} \qquad (30)$$

For complex faults debugging time lag is more than the hard faults. We define the debugging time lag as follows:

$$\Delta W(t)=\left[\frac{1}{b_1}\log\left(1+b_1W^*(t)+\frac{b_1^2W^*(t)^2}{2}\right)\right]$$

Solving equation (30) with boundary condition $t=0, m_3(0)=0, W(t=0)=W(o)$ we obtain $m_3(t)$ as

$$m_3(t)=a_3\left(1-\exp\left[\begin{array}{l}\frac{3b_3}{b_1}\left(1-\left(1+b_1W^*(t)\right)\exp\left[-b_1W^*(t)\right]\right)\\ -W^*(t)b_3\left(1-\left(1-\frac{b_1W^*(t)}{2}\right)\exp\left[-b_1W^*(t)\right]\right)\end{array}\right]\right) \qquad (31)$$

From equation (18), we get

$$m(t)=a_1\left(1-\exp\left(-b_1W^*(t)\right)\right)+a_2\left(1-\exp\left[\begin{array}{l}\frac{2b_2}{b_1}\left(1-\exp\left[-b_1W^*(t)\right]\right)-\\ W^*(t)b_2\left(1+\exp\left[-b_1W^*(t)\right]\right)\end{array}\right]\right)$$
$$+a_3\left(1-\exp\left[\begin{array}{l}\frac{3b_3}{b_1}\left(1-\left(1+b_1W^*(t)\right)\exp\left[-b_1W^*(t)\right]\right)\\ -W^*(t)b_3\left(1-\left(1-\frac{b_1W^*(t)}{2}\right)\exp\left[-b_1W^*(t)\right]\right)\end{array}\right]\right) \qquad (31)$$

4. Results and Discussion

The models have been validated using real data sets and compared on the basis of different comparison criteria.

Data Set – I: In this data set, over the course of 20 weeks, 10,000 CPU hours were consumed, and 100 software faults were removed . It is cited from (Wood 1996) from a subset of software products releases at the Tandem Computers Company.

Data Set – II: The data are cited from (Misra 1983). Over the course of 38 weeks of testing a real time system 231 faults were removed.

All the tables have been shown in the appendix.

Table-1(a-b): shows the estimated parameter results of the existing models (Kapur Younes and Agarwala 1995) for Data Set–I.GE-2, GE-3, GE-4 and GE-5 shows that only two types of faults are lying in the software and majority of them are of the nth type. GE-6 estimates the presence of three types of faults and majority of them are nth type i.e.58 %.(more complex faults). Logistic removal rate also show that two types of faults are there and the majority of them are of the nth type. However, GE-6 with logistic removal rate estimates almost same value of a parameter as GE-6 (without logistic removal rate). Here it gives β =. 12, which shows the highly exponential nature of the curves. It is observed that the SRGM with more types of faults provide lower MSE, Bias, Variation and RMSPE. SRGM with Logistic removal rate also provide lower MSE, Bias, Variation and RMSPE with more types of faults. However, for GE-6 model with logistic removal rate gives the same R^2, MSE, Bias, Variation and RMSPE as GE-6(without logistic removal rate).

Table-2(a-b): shows the estimated parameters and comparison criteria of the model i.e. equation (25) and generalized Erlang model (equation 2) for data set I.

Table-3(a-b): shows the estimated parameters and comparison criteria of the model i.e. equation (25) and generalized Erlang model (equation 2). This is the result for data set II.

Table-4(a-b) shows the estimated parameters of the

model i.e. equation (31) for various testing effort functions. Moreover the value of other comparison criteria like bias, variation and root mean square prediction error are also described in this table for data set-I and II.

.**5. Conclusion**: In this paper we have studied how the complexity of software bugs can be mathematical modeled. The paper also categories software bugs into n categories depending upon their removal time. Bug complexity has been defined by considering time taken during their removal, dependency of bugs and testing effort and correspondingly mathematical models have been developed to quantify the proportion of bugs in the software. The model has been successfully tested on several data sets obtained under different environments ranging from exponential to S-shaped or mix of the two. It is shown that the inbuilt model flexibility takes care of different environment. Categorizing the bugs into different types where each type is modeled by a different growth curve helps in making the model structure flexible and thus capturing wider class of growth curves. It is observed that introduction of more bug type's increases the flexibility of the model. Introduction of new parameter $p_i = 1,2,3$ i.e. the proportion of the minor fault, major fault and complex bugs can help us to improve testing effectiveness. The values of initial fault contents a_1, a_2, a_3 can be calculated using $a_i = ap_i; i = 1,2,3$. Actually, if a programmer can act with the knowledge of the probability distribution of simple faults, hard faults and complex faults in mind, much time and effort can be saved, and programmers will have more time to refine the software based on customer's needs or the company's reliability requirements.

ACKNOWLEDGMENTS

We are thankful to Prof. P.K. Kapur, AIB, Amity University for their support and encouragement.
First author acknowledges with thanks the research grants received from Department of Science and Technology (DST), Govt. of India with Grant No. SR/S4/MS: 600/09.

REFERENCES

Antkiewicz A.P. Wood (1996) "Predicting Software Reliability", *IEEE Computer*, pp.69-77.

Brooks W.D. and Motley, RW (1980) "Analysis of discrete software reliability models-Technical Report (RADC-TR-80-84)", *Rome Air Development Center*, New York.

Bittanti S, Bolzern P, Pedrotti E, Scattolini R(1988). "A flexible modelling approach for Software reliability growth" *Software Reliability Modelling and Identification (Ed.) G. Goos and J. Harmanis*, Springer Verlag, Berlin, 101-140.

Goel, AL and Okumoto K. (1979) "Time dependent error detection rate model for software reliability and other performance measures" *IEEE Transactions on Reliability* Vol. R-28 (3) pp.206-211.

Goswami D.N., Khatri Sunil K.and Kapur Reecha(2008) "Discrete Software Reliability Growth Modeling for Errors of Different Severity incorporating Change-Point Concept" *International Journal of Automation and Computing* Vol. 4(4) pp.396-405.

Kapur P.K., Younes S. and Agarwala S. (1995) 'Generalized Erlang Software Reliability Growth Model with n types of bugs", *ASOR Bulletin*,14,5-11

Kapur P.K., Bardhan A.K., and Kumar S. (2000) "On Categorization of Errors in a Software", *Int. Journal of Management and System*, 16(1), 37-38.

Kapur P.K. Kumar Archana ,Yadav Kalpana and Khatri Sunil(2007) "Software Reliability Growth Modelling for Errors of Different Severity using Change Point" *International Journal of Quality ,Reliability and Safety Engineering* ,Vol.14,No.4, pp 311-326.

Kapur P.K. Kumar Archana Singh V.B. and Nailana F.K.(2007) " On Modeling Software Reliability Growth Phenomanon for Errors of Different Severity" *In the Proceedings of National Conference on Computing for Nation Development*, Bhartiya Vidyapith's Institute of Computer Applications and Management, New Delhi, pp.279-284, held during 23rd-24 th February.

Kapur, P.K., Kumar Archana,Yadav, Kalpana and Kumar Jyotish**.(2007),** "Incorporating Errors of Different Severity and Change- Point in Software Reliability Growth Modeling". Published in "*Quality, Reliability and Infocom Technology*", Eds : P.K. Kapur and A. K. Verma, Macmillan India Ltd., New Delhi.

Kimura M., Yamada S. and Osaki S. **(1992)** "Software Reliability Assessment for an Exponential S-shaped Reliability Growth Phenomenon" *Computers and Mathematics with Application, 24*, pp.71-78.

Musa J.D., et.al. (1987), "*Software reliability: Measurement Prediction, Applications*" Mc Graw Hill, New York.

Ohba M.(1984) " Software Reliability Analysis Models" *IBM Journal of Research and Development* ,Vol.28,No.4,pp.428-443.

Yamada S., Osaki S and Narithisa H. (1985) "A Software Reliability Growth Model with Two Types of Errors" *RAIRO*; 19, pp. 87-104.

Singh V.B (2008). Ph.D. Thesis "A Study on Software Reliability Growth modeling Using Change Point and Fault Dependency" University of Delhi, India

Yamada S., Ohba M. and Osaki S. (1984) "S-shaped Software Reliability Growth Models and their Application" *IEEE Tran.*

P.K.Kapur, V.B.Singh, and BO Yang(2007) "Software Reliability Growth Model for Determining Fault Types" in the proceedings of 3rd *International Conference on Reliability and Safety Engineering* (Eds. R.B.Misra, V.N.A. System Journal 1983; 22(3): pp 262-270.

Naikan, S.K.Chaturvedi, and N.K.Goyal), Udaipur, 2007, pp. 334-349.

P.K.Kapur, V.B.Singh, and BasirZadeh Mashaallh(2008), "Considering Errors of Different Severity in Software Reliability Growth Modeling using Fault Dependency and Various Debugging Time Lag Functions*"* in the proceedings of Advances in Performance and safety of complex systems (Eds. A.K.Verma, P.K.Kapur and S.G. Ghadge) MacMillan India Ltd, pp. 839-849.

Misra PN. "Software reliability analysis" IB

Appendix

Table 1(a): For Data Set-I

Models (equation 16)	Parameter Estimates								
	a	b	P_1	P_2	P_3	P_4	P_5	P_6	β
GE-2	115	.1692	.4580	0.5420	-	-	-	-	-
GE-3	108	.2839	.3706	.0000	0.6294	-	-	-	-
GE-4	104	.4033	.3431	.0000	.0000	0.6569			
GE-5	102	.5223	.3334	.0000	.0000	.0000	0.6666	-	-
GE-6	101	.6123	.3167	.0000	.0000	.1001	.0000	0.5832	-
GE-2(Logistic)	101	.4104	.3825	0.6175	-	-	-	-	33
GE-3(Logistic)	101	.4093	.3932	.0000	0.6068	-	-	-	20
GE-4(Logistic)	102	.4469	.3777	.0000	.0000	0.6223	-	-	11
GE-5(Logistic)	102	.5259	.3434	.0000	.0000	.0000	0.6566		2
GE-6(Logistic)	101	.6117	.3170	.0000	.0000	.1018	.0000	0.5812	.12

Table 1(b): For Data Set-I

Models Models(equation 16)	Comparison Criteria				
	R^2	MSE	Bias	Variation	RMSPE
GE-2	.98798	9.7684	-0.3890	10.1268	10.1343
GE-3	.99200	6.5075	-0.2938	6.7591	6.7655
GE-4	.99596	3.2817	-0.1760	3.4218	3.4263
GE-5	.99804	1.5908	-0.0663	1.6699	1.6712
GE-6	.99832	1.36352	-0.0265	1.4345	1.4347
GE-2(Logistic)	.99781	1.78217	-0.0814	1.8689	1.8707
GE-3(Logistic)	.99790	1.70375	-0.0702	1.7882	1.7896
GE-4(Logistic)	.99794	1.67833	-0.0517	1.7638	1.7646
GE-5(Logistic)	.99813	1.51636	-0.0428	1.5942	1.5948
GE-6(Logistic)	.99832	1.36345	-0.0265	1.4344	1.4347

Table-2(a): For Data Set-I

Models	**Parameter Results**						
	a	b1	b2	b3	P1	P2	P3
Equation 2	562	.0215	.0121	.0407	. 6407	.3420	.0173
Equation 25	503	.0220	.0664	.6845	. 6407	.3420	.0173

Table-2(b): For Data Set-I

Models	**Comparison Criteria**				
	R^2	**MSE**	**Bias**	**Variation**	**RMSPE**
Equation 2	.99469	19.7059	-0.7567	4.4328	4.4970
Equation 25	.99644	13.2276	0.1208	3.6837	3.6857

Table-3(a): For Data Set-II

Models	Parameter Results						
	a	b1	b2	b3	p1	p2	p3
Equation 2	142	.3244	.0015	.2839	.2627	.2334	0.5039
Equation 25	103	.1289	.6789	.4005	.2501	.5744	0.1755

Table 3(b): For Data Set-II

Models	Comparison Criteria				
	R^2	MSE	Bias	Variation	RMSPE
Equation 2	.99249	6.1092	-0.2310	2.5247	2.5353
Equation 25	.99854	1.1908	-0.0005	1.1195	1.1195

Table 4(a): For Data Set-I

Testing Effort Functions	Parameter Results of Model (equation 31)						
	A	b_1	b_2	b_3	p_1	p_1	p_3
Exponential	102	.0001	.0020	.0004	.0733	.6746	0.2521
Rayleigh	122	.0001	.0005	.0052	.4807	.3591	0.1602
Weibull	111	.0001	.0013	.0018	.4256	.4148	0.1596
Logistic	114	.0003	.0004	.0026	.5044	.3615	0.1341

Table 4(b): For Data Set-II

Testing Effort Functions	Comparison criteria				
	R^2	MSE	Bias	Variation	RMSPE
Exponential	.99848	1.2324	-0.007	1.1389	1.13901
Rayleigh	.99866	1.0861	0.0001	1.0692	1.0692
Weibull	.99830	1.3845	0.0005	1.2072	1.2072
Logistic	.99868	1.0728	0.001	1.0626	1.0626