International Journal of Automation and Computing 04(4), October 2007, 359-368
DOI: 10.1007/s11633-007-0359-y

Considering the Fault Dependency Concept with Debugging Time Lag in Software Reliability Growth Modeling Using a Power Function of Testing Time

V. B. Singh[1,*] Kalpana Yadav[2] Reecha Kapur[3] V. S. S. Yadavalli[4]

[1] Delhi College of Arts and Commerce, University of Delhi, Delhi 110 023, India
[2] Indira Gandhi Institute of Technology, Guru Gobind Singh Indraprastha University, Delhi 110 006, India
[3] Department of Mathematics and Computer Application, Bundelkhand University, Jhansi 284 128, India
[4] Department of Industrial and Systems Engineering, University of Pretoria, Pretoria 0002, South Africa

Abstract: Since the early 1970s tremendous growth has been seen in the research of software reliability growth modeling. In general, software reliability growth models (SRGMs) are applicable to the late stages of testing in software development and they can provide useful information about how to improve the reliability of software products. A number of SRGMs have been proposed in the literature to represent time-dependent fault identification / removal phenomenon; still new models are being proposed that could fit a greater number of reliability growth curves. Often, it is assumed that detected faults are immediately corrected when mathematical models are developed. This assumption may not be realistic in practice because the time to remove a detected fault depends on the complexity of the fault, the skill and experience of the personnel, the size of the debugging team, the technique, and so on. Thus, the detected fault need not be immediately removed, and it may lag the fault detection process by a delay effect factor. In this paper, we first review how different software reliability growth models have been developed, where fault detection process is dependent not only on the number of residual fault content but also on the testing time, and see how these models can be reinterpreted as the delayed fault detection model by using a delay effect factor. Based on the power function of the testing time concept, we propose four new SRGMs that assume the presence of two types of faults in the software: leading and dependent faults. Leading faults are those that can be removed upon a failure being observed. However, dependent faults are masked by leading faults and can only be removed after the corresponding leading fault has been removed with a debugging time lag. These models have been tested on real software error data to show its goodness of fit, predictive validity and applicability.

Keywords: Non-homogeneous Poisson process, fault dependency, leading fault, dependent fault, delay effect.

1 Introduction

The role of software is expanding rapidly in modern society. Hence, quality, reliability, and customer satisfaction become the main goals for software engineers and very important considerations for software development organizations. Testing is the major quality control used during software development. It not only uncovers errors introduced during coding, but also removes errors introduced during the previous phases. To improve the software quality, software reliability engineering plays an important role throughout the entire software development life cycle (SDLC).

Software reliability is the probability that the software will provide failure-free operations in a fixed environment for a fixed interval of time[1]. The future failure behavior of a software system is predicted by studying and modeling its past failure behavior. How to enhance the reliability of the software systems and reduce the cost to an acceptable level becomes the main focus of the software industry[2]. A software reliability growth model (SRGMs) provides a mathematical relationship between the number of faults removed and the testing time (CPU time or calendar time) for the purpose. Many SRGMs have been developed and widely used for the past several decades that assess software reliability during testing[3−8] with the assumption that the number of faults detected at any time instant is proportional to the remaining number of faults in the software.

In Section 4.1 we will review a class of software reliability growth models[9, 10] based on non-homogeneous Poisson process (NHPP), where fault detection process is dependent not only on the number of residual fault content but also on testing time. In many software reliability growth models, it has been assumed that the detected faults are immediately removed, which is not in real practice. The time to remove a detected fault depends on the complexity of the fault, the skill and experience of the debugging team, the available manpower, the software development environment, and so on. Thus, the time delay between the fault detection and correction processes should not be negligible. Besides, during testing, mutually independent faults can be directly detected and removed, but mutually dependent faults can be removed iff the leading faults were removed. In Section 4.2 we discuss the fault detection and correction process by using a delay effect factor.

There is a class of NHPP models that assume that faults present in the software are of different types. From our studies, we found that many researchers categorize the faults with respect to the level of difficulty and time taken for

Manuscript received September 27, 2006; revised June 20, 2007
*Corresponding author. E-mail address: singh_vb@rediffmail.com

removal[7, 11]; others do so by distinguishing the means of fault identification[4, 6, 12]. Ohba[12] proposed an inflection S-shaped model to describe the software failure occurrence phenomenon with mutual dependency of faults. He visualized that an exponential SRGMs was sometimes insufficient and inaccurate to analyze actual software failure data for reliability assessment. In fact Ohba categorize the faults as independent and dependent faults. Dependent faults can be removed only after some faults (independent faults) lying on that path are detected. Kapur and Garg[4] assumed that more faults could be removed during the checking of code for identification of the cause of a failure. It is a fact that different categories of faults exist in a software. In this paper, we develop several SRGMs by modifying the above assumption. We assume that two types of faults exist in a software, i.e., leading faults and dependent faults. Leading faults are those that cause failure, and dependent faults are detected upon identification of leading faults. The model is developed as a two-stage process. This type of modeling was done first by Kapur and Younes[6]. Following their work, Kapur *et al.*[13] incorporated time-dependent lag function into the second stage, i.e., modeling of a dependent fault detection process. Huang *et al.*[14] also proposed several SRGMs that incorporate different time-dependent lag functions during modeling of a dependent fault detection process.

The structure of this paper is organized as follows. Section 2 gives the description of the model assumptions. In Section 3.1 we review a class of NHPP models proposed by Kapur *et al.*[9, 10] by considering the concept of power function of testing time, and a fault detection and correction process by using delay effect factor explained in Section 3.2. In Section 3.3 we propose four models, which incorporate the fault dependency concept with various debugging time lag using a power function of testing time. We describe data sets cited in the literature, comparison criteria, description of tables, parameter results and goodness of fit curves in Section 4. Finally, Section 5 concludes this paper and indicates future research prospects.

Notation

$m(t)$: Value function of the expected number of detected/removed faults in the time interval [0, t].

$m_f(t)$: Value function of the expected number of observed/detected faults in the time interval [0, t].

$m_r(t)$: Value function of the expected number of removed faults in the time interval [0, t].

$m_1(t)$: Value function of the expected number of leading faults.

$m_2(t)$: Value function of the expected number of dependent faults.

a, k : Constants, represents initial fault content and power of time, respectively.

β : Constant.

$e(t)$: Expected number of instructions executed in the time interval [0, t].

a_1 : Total number of independent faults.

a_2 : Total number of dependent faults.

b : Fault detection rate of independent faults.

c : Faults detection rate of dependent faults.

P : Proportion of leading faults.

Δt : Delay effect factor.

2 Basic assumptions

If we define the expected number of faults, $N(t)$, whose mean value function is known as $m(t)$, then an SRGMs based on NHPP can be formulated as a Poisson process:

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

where $m(t)=\int_0^t \lambda(x)\mathrm{d}x$.

The intensity function $\lambda(x)$ (or the mean value function $m(t)$) is the basic building block of all the NHPP models existing in the software reliability engineering literature.

The proposed model is based on the following assumptions.

Assumption 1. Software is subject to failures at random times caused by errors remaining in the software.

Assumption 2. The fault removal process follows an NHPP.

Assumption 3. Identified faults are removed perfectly. No additional faults are introduced during the removal process.

Assumption 4. Number of faults removed is a function of number of instructions executed.

Assumption 5. The number of instructions executed is a power function of testing time.

Assumption 6. All detected faults can be categorized as either leading faults or dependent faults. The total number of faults is finite.

Assumption 7. The mean number of leading faults detected in the time interval [t, $t+\Delta t$] is proportional to the mean number of remaining leading faults in the system.

Assumption 8. The mean number of dependent faults detected in the time interval [t, $t+\Delta t$] is proportional to the mean number of remaining dependent faults in the system and to the ratio of leading faults removed at time t to the total number of leading faults.

Assumption 9. The detected dependent faults may not be immediately removed, and it lags the fault detection process by a delay effect factor Δt.

3 Software reliability growth modeling

In this section, we review software reliability growth models developed with a different approach where the fault detection process is dependent not only on the number of residual fault content but also on testing time. We also describe the fault detection and removal process and see how these models can be alternatively derived using a delay effect factor. Finally, in this section, we propose the fault dependency concept with various debugging time lag in software reliability growth modeling using a power function of testing time.

3.1 A general NHPP model

In the literature, many software reliability growth models have been developed with the assumption that the fault detection process depends only on residual fault contents. However, in the following, we will review some software reliability growth models which have been developed with the assumption that the fault detection process is dependent not only on the number of residual fault content but also on the testing time.

3.1.1 A general NHPP model

During testing, instructions are executed on the software and the output is matched with the expected results. If there is any discrepancy a failure is said to have occurred. Effort is made to identify and remove the cause of the failure.

On the basis of Assumptions 4 and 5, the failure / removal phenomenon can be described with respect to time as follows:

$$\frac{\mathrm{d}m_r(t)}{\mathrm{d}t} = \frac{\mathrm{d}m_r(t)}{\mathrm{d}e(t)}\frac{\mathrm{d}e(t)}{\mathrm{d}t}. \tag{2}$$

Each component of the right hand side of the above expression is described in (3) and (4) respectively.

The rate at which failures occur depends upon the number of faults remaining in the software. Based on this assumption the differential equation for fault removal can be written as

$$\frac{\mathrm{d}m_r(t)}{\mathrm{d}e(t)} = k_1\left(a - m_r(t)\right). \tag{3}$$

k_1 is the rate at which residual faults cause failure. It is a constant as each one of the faults has an equal probability of causing failure.

The second component of (2) relates the number of instructions executed with testing time. In [9, 10], this component has been taken as a power function of testing time.

$$\frac{\mathrm{d}e(t)}{\mathrm{d}t} = k_2 t^k \tag{4}$$

here, $k \geq 0$. The form given in (4) is more general and flexible in nature, and it represents different types of relationships between e and t depending on different values of k.

Substituting (3) and (4) in (2) we get

$$\frac{\mathrm{d}m_r(t)}{\mathrm{d}t} = k_1 k_2\left(a - m_r(t)\right)t^k. \tag{5}$$

Solving (5) with initial condition $m_r(0) = 0$ we get

$$m_r(t) = a\left(1 - \exp\left(-\frac{b}{k+1}t^{(k+1)}\right)\right) \tag{6}$$

here, $b = k_1 k_2$. If $k = 0$ the above model reduces to a G-O model[3].

We may also write the above-described SRGMs in the following form

$$\frac{\mathrm{d}m_r(t)}{\mathrm{d}t} = b(t)\left(a - m_r(t)\right)$$

where $b(t) = bt^k$. Then by solving for $m_r(t)$ we get (6).

In the above discussion, it has been assumed that the detected faults are immediately removed. The mean value function of detected faults as in (6) can be taken as a failure observation / fault detection phenomenon, i.e.,

$$m_f(t) = a\left(1 - \exp\left(-\frac{b}{k+1}t^{(k+1)}\right)\right). \tag{7}$$

3.1.2 Yamada delayed s-shaped model using power function of testing time proposed by Kapur[10]

In this model, a realistic assumption has been made that actual removal of a fault is done after a failure is observed and corresponding fault is isolated. Hence, the removal is done in two steps. In the first step, the failure identification team isolates a failure. In the second step, another team removes the fault causing that failure. Yamada *et al.*[8] has developed an s-shaped SRGMs taking into account the time lag between the two stages, i.e., failure and removal by defining the failure observation and fault removal rates as a constant b. In this model, the rate of failure observation and their corresponding removal has been taken as a power function of testing time, i.e., $b(t) = bt^k$.

Let us define:

$$\frac{\mathrm{d}m_f(t)}{\mathrm{d}t} = bt^k(a - m_f(t))$$

and

$$\frac{\mathrm{d}m_r(t)}{\mathrm{d}t} = b\,t^k(m_f(t) - m_r(t))$$

where $m_f(t)$ denotes the number of failures observed in time t whereas $m_r(t)$ represents the number of faults removed in time t.

Solving the above two equations with the initial condition $m_f(0) = m_r(0) = 0$, we get

$$m_r(t) = a\left(1 - \left(1 + b\frac{t^{k+1}}{k+1}\right)\exp\left(-b\frac{t^{k+1}}{k+1}\right)\right). \tag{8}$$

Here, if $k = 0$, then it reduces to a Yamada S-shaped model[8].

3.1.3 K-3 stage-model using power function of testing time proposed by Kapur *et al.*[10]

This model was proposed by Kapur *et al.*[10] and in this model it has been assumed that software testing and debugging consists of three different processes: failure observation, fault isolation and fault removal. The time lag between the failure observation and fault isolation / removal represents the severity of the fault. The harder the fault, the longer is the time lag, and rate has been taken as a power function of testing time.

Let us define:

$$\frac{\mathrm{d}m_f(t)}{\mathrm{d}t} = bt^k(a - m_f(t))$$

$$\frac{\mathrm{d}m_i(t)}{\mathrm{d}t} = bt^k(m_f(t) - m_i(t))$$

$$\frac{\mathrm{d}m_r(t)}{\mathrm{d}t} = bt^k(m_i(t) - m_r(t)).$$

Solving the above three equations with the initial condition $m_f(0) = m_i(0) = m_r(0) = 0$, we get

$$m_r(t) = a\left(1 - \left(1 + b\frac{t^{k+1}}{k+1} + \frac{b^2}{2}\left(\frac{t^{k+1}}{k+1}\right)^2\right)\right. \\ \left.\exp\left(-b\frac{t^{k+1}}{k+1}\right)\right). \tag{9}$$

Now if $k = 0$, then it reduces to a Kapur 3-stage model[11].

3.1.4 Kapur–Garg model using power function of testing time proposed by Kapur[10]

This model was proposed by Kapur *et al.*[10] and it has been assumed that the rate at which failures occur depends not only upon the number of faults remaining in the software but also on the proportion of faults already detected. Based on this assumption the differential equation for fault identification / removal can be written as

$$\frac{\mathrm{d}m_r(t)}{\mathrm{d}e(t)} = (k_1 + k_2\frac{m_r(t)}{a})(a - m_r(t)) \tag{10}$$

where k_1 is the rate at which residual faults cause failure. It is a constant as each one of these faults has an equal probability of causing failure. k_2 is the rate at which additional faults are identified without their causing any failure.

Let the second component of (2) be defined as a power function of testing time:

$$\frac{\mathrm{d}e(t)}{\mathrm{d}t} = k_3 t^k. \tag{11}$$

Substituting (10) and (11) in (2) we have

$$\frac{\mathrm{d}m_r(t)}{\mathrm{d}t} = k_3 t^k\left(k_1 + k_2\frac{m_r(t)}{a}\right)(a - m_r(t)).$$

It is a first order differential equation. Solving it with the initial condition $m_r(0)= 0$ we get

$$m_r(t) = a\left(\frac{1 - \exp\left(-b\frac{t^{k+1}}{k+1}\right)}{1 + \beta\exp\left(-b\frac{t^{k+1}}{k+1}\right)}\right) \tag{12}$$

where $b = k_3(k_1 + k_2)$ and $\beta = (k_2/k_1)$.

If we have $k = 0$, then it is the same as the Kapur-Garg model[4].

3.2 Modeling of fault detection and removal process

In Section 3.1.1, we have discussed a model with the assumption that the detected faults are immediately removed and the mean value function is given by (6). But from a practical point of view, the assumption that detected faults are immediately removed may not be suitable in an actual software environment. However, it lags the fault detection process by a delay effect factor Δt. We will discuss in this section how existing software reliability growth models described in Sections 3.1.2–3.1.4 can be reinterpreted as the delayed fault detection models.

From the modified assumption and using (7), the mean value function is given by

$$m_r(t) = m_f(t - \Delta t) \tag{13}$$

$$m_r(t) = a\left(1 - \exp\left(-\frac{b}{k+1}(t - \Delta t)^{(k+1)}\right)\right). \tag{14}$$

3.2.1 Goel–Okumoto model using power function of testing time proposed by Kapur[9, 10]

This model has been reviewed in Section 3.1.1 and can be reinterpreted as the delayed fault detection model. If we consider that there is no lag between fault detection and their corresponding removal, i.e., $\Delta t = 0$ in (14), we have

$$m_r(t) = a\left(1 - \exp\left(-\frac{b}{k+1}t^{(k+1)}\right)\right). \tag{15}$$

3.2.2 Yamada delayed s-shaped model using the power function of testing time proposed by Kapur[10]

This model has been reviewed in Section 3.1.2 and can be reinterpreted as the delayed fault detection model. If we consider that there is a time lag between fault detection and its corresponding removal, i.e.,

$$\Delta t = t - \left[t^{k+1} - \frac{k+1}{b}\log\left(1 + \frac{bt^{k+1}}{k+1}\right)\right]^{\frac{1}{k+1}}$$

in (14), we have

$$m_r(t) = a\left(1 - \left(1 + b\frac{t^{k+1}}{k+1}\right)\exp\left(-b\frac{t^{k+1}}{k+1}\right)\right). \tag{16}$$

3.2.3 K-3 stage-model using power function of testing time proposed by Kapur[10]

This model has been reviewed in Section 3.1.3 and can be reinterpreted as the delayed fault detection model. If we consider that there is time lag between fault detection and its corresponding removal, i.e.,

$$\Delta t = t - \left[t^{k+1} - \frac{k+1}{b}\log\left(1 + \frac{bt^{k+1}}{k+1} + \frac{b^2}{2}\left(\frac{t^{k+1}}{k+1}\right)^2\right)\right]^{\frac{1}{k+1}}$$

in (14), we have

$$m_r(t) = a\left(1 - \left(1 + b\frac{t^{k+1}}{k+1} + \frac{b^2}{2}\left(\frac{t^{k+1}}{k+1}\right)^2\right)\right. \\ \left.\exp\left(-b\frac{t^{k+1}}{k+1}\right)\right). \tag{17}$$

3.2.4 Kapur–Garg model using the power function of testing time proposed by Kapur[10]

This model has been reviewed in Section 3.1.4 and can be reinterpreted as the delayed fault detection model. If we consider that there is time lag between fault detection and its corresponding removal, i.e.,

$$\Delta t = t - \left[-\frac{1}{b}\log\left(\frac{(1+\beta)\exp\left(-\frac{bt^{k+1}}{k+1}\right)}{1 + \beta\exp\left(-\frac{bt^{k+1}}{k+1}\right)}\right)^{k+1}\right]^{\frac{1}{k+1}}$$

in (14), we have

$$m_r(t) = a\left(\frac{1-\exp\left(-\dfrac{bt^{k+1}}{k+1}\right)}{1+\beta\exp\left(-\dfrac{bt^{k+1}}{k+1}\right)}\right). \tag{18}$$

3.3 Considering fault dependency and debugging time lag in software reliability modeling using a power function of testing time

Here, we assume that two types of faults exist in a software, i.e., leading faults and dependent faults. In general, mutually independent faults are not involved with other faults, and can be directly detected and removed. Considering mutual dependent faults, the statements associated with the observed faults can produce an expected computation if the leading faults are also removed. Therefore, we have to analyze the dependencies between faults, i.e., data dependency and control dependency. For example, let us consider two statements A and B, if a variable x is defined, and used in the statement A and B, and x is not redefined prior to B, then a data dependency exists between them. On the other hand, if the execution of statement A prevents the execution of statement B, a control dependency exists between the statements A and B. In the following, we are explaining the concept of fault dependency by using a sequential statement, as shown in Table 1.

Table 1 Concept of fault dependency

Correct program		Faulty program		Remarks
Line	Statement	(Code)		(Code)
10	S-1	$x = p + 1$	$x = p - 1$	misusing operator
⋮		⋮	⋮	
20	S-2	$y = q\%2$	$y = q\%5$	wrong operand
⋮		⋮	⋮	
30	S-3	$z = 200\%x$	$z = 200/x$	misusing operator
⋮		⋮	⋮	
40	S-4	$z1 = x\%y$	$z1 = y\%x$	reverse order

It is clear from Table 1 that the left hand side is a correct program and the right hand side is a faulty program. The state after execution of line 30 (S-3) will not be corrected unless the misused operators in statement S-1 and S-3 are both corrected. Similarly, the definition fault of variable x in statement S-1 will also spread to statement S-4 (line 40). So it is clear that the removal of the leading fault (misusing operator) in statement S-1 is critical to the perfect removal of faults in statements S-3 and S-4. Similarly, the fault dependency concept can be illustrated for repetitive statement and conditional statement.

In this section we attempt to develop an SRGMs incorporating dependency of the errors using a power function of testing time. From Assumption 6, we have the following equation:

$$a = a_1 + a_2 \tag{19}$$

where a_1 and a_2 are the number of leading and dependent faults respectively.

The proposed model is the mean value function of an NHPP, i.e., Assumption 2. Let $m(t)$ represent the mean number of errors removed in time $[t, t+\Delta t]$. The removal of leading and dependent errors is also assumed to follow an NHPP. Thus, $m(t)$ can be written as the superposition of two NHPPs:

$$m(t) = m_1(t) + m_2(t) \tag{20}$$

where $m_1(t)$ is the mean value function of the expected number of leading faults detected in time $[0, t]$ and $m_2(t)$ is the mean value function of the expected number of dependent faults detected in time $[0, t]$.

Consequently, if the number of detected leading faults is proportional to the number of remaining leading faults, i.e., Assumption 7, then we obtain the following differential equation

$$\frac{\mathrm{d}m_1(t)}{dt} = bt^k\left[a_1 - m_1(t)\right], \quad a_1 > 0, \quad 0 < b \le 1. \tag{21}$$

In the above equation, we used a power function of testing time, which has been discussed in Section 3.1.

Solving (21) under the boundary condition, i.e., at $t = 0$, $m_1(0) = 0$, we have

$$m_1(t) = a_1\left(1-\exp\left(-\frac{bt^{k+1}}{k+1}\right)\right). \tag{22}$$

The other forms of removal phenomenon of detected faults can be found in Section 3.1.

From Assumption 8 and by using the concept of power function of testing time, which has been discussed in Section 3.1, we have the following differential equation:

$$\frac{\mathrm{d}m_{2(t)}}{\mathrm{d}t} = ct^k\left[a_2 - m_2(t)\right]\frac{m_1(t-\Delta t)}{a_1}. \tag{23}$$

In earlier literature Kapur and Younes[6] assumed that the mean number of dependent faults detected in the time interval $[t, t+\Delta t]$ is proportional to the mean number of remaining dependent faults in the system and the ratio of leading faults removed at time t to the total number of faults.

Kapur *et al.*[13] modified the above assumption by taking the ratio of leading faults removed at time t to the total number of leading faults. Later, Huang *et. al.*[14] incorporated the assumption given by Kapur and Younes[9] in their modeling. It is important to note that the dependent faults can be removed only when the leading fault is perfectly removed with a debugging time lag Δt.

Here we let

$$a_1 = pa \quad \text{and} \quad a_2 = (1-p)a, \quad 0 \le p \le 1. \tag{24}$$

Using different removal equations discussed in Section 3.1., solving (23) with boundary condition $t = 0$, $m_2(0) = 0$, and using equation (20), we obtain $m(t)$ as can be seen in different cases discussed below:

Case 1. (refer to Sections 3.1.1 and 3.2.1)
If $\Delta t = 0$, (20) becomes

$$m\left(t\right) = a\left(1 - p\exp\left[-\frac{bt^{k+1}}{k+1}\right] - (1-p) \right. \\ \left. \exp\left[-\frac{c}{b}\left(1 - \exp\left[-\frac{bt^{k+1}}{k+1}\right]\right)\frac{t^{k+1}}{k+1}c\right]\right). \tag{25}$$

If we take $k = 0$, $\Delta t = 0$ and incorporate the assumption described by Kapur and Younes[6] during the modeling of dependent faults, we get the error dependency model[6].

Case 2. (refer to Sections 3.1.2 and 3.2.2)
If

$$\Delta t = t - \left[t^{k+1} - \frac{k+1}{b}\log\left(1 + \frac{bt^{k+1}}{k+1}\right)\right]^{\frac{1}{k+1}}$$

(20) becomes

$$m\left(t\right) = a\left(1 - p\left(1 + \frac{bt^{k+1}}{k+1}\right)\exp\left[-\frac{bt^{k+1}}{k+1}\right] - \right. \\ (1-p)\exp\left[\frac{2c}{b}\left(1 - \exp\left[-\frac{bt^{k+1}}{k+1}\right]\right) - \right. \\ \left.\left. \frac{t^{k+1}}{k+1}c\left(1 + \exp\left[-\frac{bt^{k+1}}{k+1}\right]\right)\right]\right). \tag{26}$$

If we take $k = 0$ and incorporate the assumption described by Kapur and Younes[6] during the modeling of dependent faults, the above model reduces to Huang *et al.* model 1[14].

Case 3. (refer to Sections 3.1.3 and 3.2.3)
If

$$\Delta t = t - \left[t^{k+1} - \frac{k+1}{b}\log\left(1 + \frac{bt^{k+1}}{k+1} + \frac{b^2}{2}\left(\frac{t^{k+1}}{k+1}\right)^2\right)\right]^{\frac{1}{k+1}}$$

(20) becomes

$$m\left(t\right) = a\left(1 - p\left(1 + \frac{bt^{k+1}}{k+1} + \frac{b^2}{2}\left(\frac{t^{k+1}}{k+1}\right)^2\right)\exp\left[-\frac{bt^{k+1}}{k+1}\right] - \right. \\ \left. (1-p)\exp\left[\begin{array}{l} \frac{3c}{b}\left(1 - \left(1 + \frac{bt^{k+1}}{k+1}\right)\exp\left[-\frac{bt^{k+1}}{k+1}\right]\right) \\ -\frac{t^{k+1}}{k+1}c\left(1 - \left(1 - \frac{bt^{k+1}}{2(k+1)}\right)\exp\left[-\frac{bt^{k+1}}{k+1}\right]\right) \end{array}\right]\right). \tag{27}$$

Case 4. (refer to Section 3.1.4, 3.2.4)
If

$$\Delta t = t - \left[-\frac{1}{b}\log\left(\frac{(1+\beta)\exp\left(-\frac{bt^{k+1}}{k+1}\right)}{1 + \beta\exp\left(-\frac{bt^{k+1}}{k+1}\right)}\right)^{k+1}\right]^{\frac{1}{k+1}}$$

(20) becomes

$$m\left(t\right) = a\left(1 - p\frac{(1+\beta)\exp\left(-\frac{bt^{k+1}}{k+1}\right)}{1 + \beta\exp\left[-\frac{bt^{k+1}}{k+1}\right]} - (1-p) \right. \\ \left. \exp\left(-\frac{ct^{k+1}}{k+1}\right)\left(\frac{1+\beta}{1 + \beta\exp\left(-\frac{bt^{k+1}}{k+1}\right)}\right)^{\frac{c(1+\beta)}{b\beta}}\right). \tag{28}$$

If we take $k = 0$ and incorporate the assumption described by Kapur and Younes[6] during the modeling of dependent faults, the above model reduces to Huang *et al.* model 2[14].

A summary of NHPP models is shown in Table 2.

Table 2. A summary of NHPP models

Model	Mean value function
G-O model[3]	$m\left(t\right) = a\left(1 - \exp\left(-bt\right)\right)$
Yamada-delayed s-shaped model[8]	$m(t) = a\ \left(1 - \left(1 + bt\right)\exp\left(-bt\right)\right)$
Kapur-3-stage model[11]	$m_r(t) = a\ \left(1 - \left(1 + bt + \frac{b^2t^2}{2}\right)\exp\left(-bt\right)\right)$
K-G model[4]	$m\left(t\right) = a\left(\frac{1-\exp(-bt)}{1+\beta\exp(-bt)}\right)$
Error dependency model[6]	$m\left(t\right) = a\left(1 - p\exp\left[-bt\right] - (1-p)\cdot \exp\left[\frac{pc}{b}\left(1 - \exp\left[-bt\right]\right) - pct\right]\right)$
Huang *et. al.* model 1[14]	$a\left(1 - p\left(1 + rt\right)\exp\left[-rt\right] - (1-p)\cdot \exp\left[\frac{2p\theta}{r}\left(1 - \exp\left[-rt\right]\right) - tp\theta\left(1 + \exp\left[-rt\right]\right)\right]\right)$
Huang *et. al.* model 2[14]	$a\left(1 - p\frac{(1+\varphi)\exp[-rt]}{1+\varphi\exp[-rt]} - (1-p)\cdot \exp\left[-pt\theta\right]\left(\frac{1+\varphi}{1+\varphi\exp[-rt]}\right)^{\frac{p\theta(1+\varphi)}{r\varphi}}\right)$

4 Parameter estimation and comparison criteria

To verify the proposed models that incorporate the concept of fault dependency and various debugging time lag using a power function of testing time, we estimated the unknown parameters by using the statistical package for social sciences (SPSS) software tool based on non-linear regression technique.

4.1 Description of data sets

Data set I. The data is cited from Brooks and Motley[15]. The fault data set is for a radar system of size 124 KLOC (kilo lines of code) tested for 35 months in which 1 301 faults were identified.

Data set II. The data is obtained from Musa *et al.*[16]. The software is a real-time command and control system, which was tested for 92 days (21 weeks). The delivered object instructions were 21 700 involving 9 programmers; 136 faults were removed during testing.

4.2 Comparison criteria

The performance of an SRGMs is judged by its ability to fit the past software fault and to predict satisfactorily the future behavior of the software fault removal process. Therefore, we use the following comparison criteria.

4.2.1 The mean square fitting error (MSE)[5]

The model under comparison is used to simulate the fault data. The difference between the estimated values, $m(t_i)$ and the observed values y_i is measured by MSE as follows.

$$\text{MSE} = \frac{\sum\limits_{i=1}^{k}\left(m(t_i) - y_i\right)^2}{k}$$

where k is the number of observations. The lower the value of MSE indicates, the less the fitting error is, thus the better the goodness of fit is.

4.2.2 Coefficient of determination (R^2)[5]

This goodness of fit measure can be used to investigate whether a significant trend exists in the observed failure intensity. We define this coefficient as the ratio of the sum of squares (SS) resulting from the trend model to that from the constant model subtracted from 1.

$$R^2 = 1 - \frac{\text{Corrected SS}}{\text{Residual SS}}.$$

R^2 measures the percentage of the total variation about the mean accounted for the fitted curve. It ranges in value from 0 to 1. Small values indicate that the model does not fit the data well. The larger R^2 is, the better the model explains the variation in the data.

4.2.3 Bias

The difference between the observation and prediction of number of failures at any instant of time i is known as PE_i (prediction error). The average of PE is known as Bias. The lower the value of Bias, the better the goodness of fit is.

4.2.4 Variation

The standard deviation of prediction error is known as Variation.

$$\text{Variation} = \sqrt{\frac{1}{N-1}\sum(\text{PE}_i - \text{Bias})^2}.$$

The lower the value of Variation is, the better the goodness of fit is.

4.2.5 Root mean square prediction error (RMSPE)

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

$$\text{RMSPE} = \sqrt{(\text{Bias}^2 + \text{Variation}^2)}$$

4.2.6 Predictive validity criterion[5]

The number of faults removed by time t_k can be predicted by the SRGMs and compared to the reported fault removal, i.e., y_k. The difference between the predicted value $\hat{m}(t_k)$ and the reported value measures the fault in prediction. The ratio $[(\hat{m}(t_k) - y_k)/y_k]$ is called the relative prediction error (RPE). If the RPE is negative (positive) the SRGMs is said to underestimate (overestimate) the fault removal process. Portions of the failure data are sequentially chosen to calculate the RPE. Values close to zero for RPE indicate more accurate prediction, thus more confidence in the model. Value is acceptable if it is within ± 10.

Table 3 (a) shows the estimated parameters of the proposed models and various existing models. It is noted that for proposed models, i.e., (25) - (28) independent faults are approximately 52%, 97%, 30%, and 90%, and dependent faults are approximately 48%, 3%, 70%, and 10%, respectively for data set I.

Table 3 (b) shows the values of different comparison criteria for the proposed models and various existing models. As can be seen, the proposed models almost provide the lowest MSE, and the highest R^2 when compared with other existing models. Moreover, the value of other comparison criteria like bias, variation and root mean square prediction error are also described in this table. The mean value function (MVF) of the proposed models provides a good fit to data set I.

Table 3 (c) shows the value of relative predictive error (RPE) of the proposed models and various existing models. As can be seen, the value of the relative predictive error is much less at different truncation points compared with other existing models for data set I.

Table 4 (a) shows the estimated parameters of the proposed models and various existing models. It is noted that for the proposed models, i.e., (25) - (28) independent faults are 43%, 98%, 98%, and 85% approximately, and dependent faults are 57%, 2%, 2%, and 15% approximately, respectively for data set II. As seen from the table, the G-O model, Yamada delayed S-shaped model and K-3 stage model do not provide good estimates for the value of parameter a.

Table 4 (b) shows the values of different comparison criteria for proposed models and various existing models. As can be seen, the proposed models almost provide the lowest MSE, and the highest R^2 when compared with other existing models. Moreover, the value of other comparison criteria like bias, variation and root mean square prediction error are also described in this table. The MVF of the proposed models provides a good fit to data set II.

Table 4 (c) shows the value of the relative predictive error of the proposed models and various existing models. As seen, the value of relative predictive error is much less at different truncation points compared with other existing models for data set II.

4.2.7 Parameter results

Several models are included in Tables 3 and 4. They are G-O model[3], Yamada-delayed s-shaped model[8], Kapur-3-stage model[11], K-G model[4], error dependency model[6], Huang *et al.* model 1[14], and Huang *et. al.* model 2[14]. In Tables 3 and 4, "-" denotes "not applicable".

Table 3. (a-c) Data set I
(a) Parameter estimates

Model	a	b	c	p	k	β
G-O model	10589	.004	-	-	-	-
Yamada-delayed s-shaped model	1689	.090	-	-	-	-
Kapur-3-stage model	1449	.166	-	-	-	-
K-G model	1331	.201	-	-	-	20.16
Error dependency model	1559	.0438	.913	.144	-	-
Huang *et al.* model 1	1414	.094	.819	.314	-	-
Huang *et al.* model 2	1349	.129	.257	.920	-	1.455
Proposed model 1 (Equation 25)	1325	.013	.021	.525	.775	-
proposed model 2 (Equation 26)	1311	.029	.006	.975	.648	-
proposed model 3 (Equation 27)	1319	.093	.017	.309	.973	-
Proposed Model 4 (Equation 28)	1312	.082	.062	.909	.362	1.747

(b) Comparison criteria

Model	R^2	MSE	Bias	Variation	RMSPE
G-O model	.95810	8923.676	18.712	93.945	95.791
Yamada-delayed s-shaped model	.98726	2713.172	5.88	52.511	52.839
Kapur-3-stage model	.99423	1228.158	-3.442	35.384	35.552
K-G model	.99904	203.816	-2.147	14.320	14.480
Error dependency model	.99013	2101	6.676	46.011	46.494
Huang *et al.* model 1	.99642	762.83	-5.812	27.395	28.005
Huang *et al.* model 2	.99920	170.303	1.404	13.164	13.238
Proposed model 1 (Equation 25)	.99835	352.261	-3.95	18.616	19.031
Proposed model 2 (Equation 26)	.99813	397.9859	0.046	20.241	20.241
Proposed model 3 (Equation 27)	.99779	471.255	0.049	22.025	22.025
Proposed model 4 (Equation 28)	.99967	70.228	0.248	8.499	8.503

(c) Relative predictive error (RPE)

Model	100%	90%	80%	70%
G-O model	.119	.171	.198	.195
Yamada-delayed s-shaped model	.067	.112	.206	.326
Kapur-3-stage model	.035	.057	.095	.138
K-G model	.004	.009	.021	.059
Error dependency model	.054	.081	-.057	-.087
Huang *et al.* model 1	.043	.067	.098	-.113
Huang *et al.* model 2	.007	.013	.027	.064
Proposed model 1 (Equation 25)	.004	.012	.036	-.001
Proposed model 2 (Equation 26)	.003	.009	.034	.128
Proposed model 3 (Equation 27)	.005	.034	.048	.156
Proposed model 4 (Equation 28)	.003	.003	-.014	-.012

Table 4 (a-c) Data Set II

(a) Parameter estimates

Model	a	b	c	p	k	β
G-O model	63883	.00008	-	-	-	-
Yamada-delayed s-shaped model	503551	.001	-	-	-	-
Kapur-3-stage model	2032	.046	-	-	-	-
K-G Model	150	.417	-	-	-	582.9
Error dependency model	218	.045	.282	.253	-	-
Huang *et al.* model 1	196	.098	.554	.304	-	
Huang *et al.* model 2	261	.063	.964	.959	1.186	-
Proposed model 1 (Equation 25)	144	.0003	.0005	.435	2.454	-
Proposed model 2 (Equation 26)	145	.001	.022	.988	2.156	-
Proposed model 3 (Equation 27)	147	.008	.001	.988	1.478	-
Proposed model 4 (Equation 28)	145	.014	.022	.856	1.107	1.432

(b) Comparison criteria

Model	R^2	MSE	Bias	Variation	RMSPE
G-O model	.74141	613.01	8.361	23.880	25.302
Yamada-delayed s-shaped model	.95580	104.79	3.852	9.719	10.454
Kapur-3-stage model	.97965	48.227	0.958	7.048	7.113
K-G model	.99716	6.723	-0.525	2.602	2.654
Error dependency model	.80085	472.09	2.189	22.151	22.259
Huang *et al.* model 1	.91721	196.28	4.672	13.534	14.318
Huang *et al.* model 2	.99480	12.317	0.2438	3.588	3.596
Proposed model 1 (Equation 25)	.99708	6.934	-0.667	2.610	2.694
Proposed model 2 (Equation 26)	.99727	6.479	0.024	2.608	2.608
Proposed model 3 (Equation 27)	.99698	7.162	0.003	2.742	2.742
Proposed model 4 (Equation 28)	.99760	5.683	-0.152	2.438	2.443

(c) Relative predictive error (RPE)

Model	100%	90%	80%	70%
G-O model	-.233	-.314	-.445	-.575
Yamada-delayed s-shaped model	.026	.026	-.071	-.192
Kapur-3-stage model	.019	-.021	.033	.087
K-G model	.016	.081	.095	-.037
Error dependency model	.049	.057	-.181	-.234
Huang *et al.* model 1	.027	.028	-.041	-.217
Huang *et al.* model 2	.025	.031	-.126	-.142
Proposed model 1 (Equation 25)	.009	.066	-.084	.096
Proposed model 2 (Equation 26)	.002	.011	-.056	.086
Proposed model 3 (Equation 27)	.003	.021	.048	.102
Proposed model 4 (Equation 28)	.001	.008	-.042	-.068

4.3 Goodness of fit curves for the proposed SRGMs

Figs. 1 and 2 show the goodness of fit curves of the proposed models.

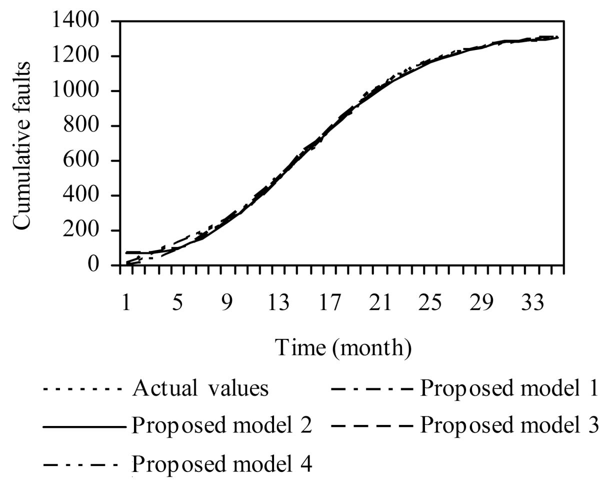

Fig. 1 Mean value function of the proposed models for data set I

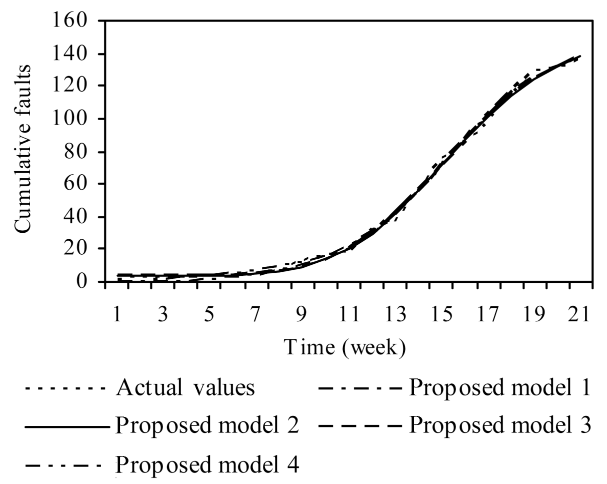

Fig. 2 Mean value function of the proposed models for data set II

5 Conclusions

In this paper, we first review an approach where the fault detection process is dependent not only on residual fault content but also on testing time. Based on this concept, we proposed several software reliability growth models, which incorporate the fault dependency concept and various debugging time lag using a power function of testing time. We have provided a simple but useful approach to measure and assess software reliability during testing. It has significant potential in predicting software reliability during the testing phase. Experimental results show that the proposed framework to incorporate the fault dependency concept and various debugging time lag using a power function of testing time for an SRGMs has a fairly accurate goodness of fit. It is assumed during the present modeling that there is a time lag between detection and subsequent removal. The various time lags assumed give rise to an s-shaped curve for an independent fault detection process, which can be relaxed, and it may be assumed that the time lag will be zero for independent faults and the total fault detection phenomenon is given by a more flexible model. Depending upon the values of the parameters, it may reduce to either exponential or s-shaped or a mix of the two unlike the one described in this paper where the fault detection process is given by s-shaped. This comparison will be brought in future research work. Consequently, the development and production of software can be improved significantly.

References

[1] J. D. Musa, A. Iannino, K. Okumoto. *Software Reliability: Measurement Prediction, Application,* McGraw Hill, New York, 1987.

[2] H. Pham. *Software reliability*, Springer, Sigapore, 2000.

[3] A. L. Goel, K. Okumoto. Time Dependent Error Detection Rate Model for Software Reliability and Other Performance Measures. *IEEE Transactions on Reliability*, vol. 28, no. 3, pp. 206–211, 1979.

[4] P. K. Kapur, R. B Garg. A Software Reliability Growth Model for an Error Removal Phenomenon. *Software Engineering Journal*, vol. 7, no. 4, pp. 291–294, 1992.

[5] P. K. Kapur, R. B. Garg, S. Kumar. *Contributions to Hardware and Software Reliability*, World Scientific, Singapore, 1999.

[6] P. K. Kapur, S. Younes. Software Reliability Growth Model with Error Dependency. *Microelectronics and Reliability*, vol. 35, no. 2, pp. 273–278, 1995.

[7] P. K. Kapur, A. K. Bardhan, S. Kumar. On Categorization of Errors in a Software. *Intrenational Journal of Management and System*, vol. 16, no. 1, pp. 37–38, 2000.

[8] S. Yamda, M. Ohba, S. Osaki. S-shaped Reliability Growth Modeling for Software Error Detection. *IEEE Transactions on Reliability*, vol. 32, no. 5, pp. 475–484, 1983.

[9] P. K. Kapur, V. B. Singh, S. Anand, V. S. S. Yadavalli. Software Reliability Growth Model with Change-point and Effort Control Using a Power Function of Testing Time. *International Journal of Production Research*, [online], Available: http://www.informaworld.com, November 17, 2006.

[10] P. K. Kapur, V. S. S. Yadavalli, A. Gupta. Software Reliability Growth Modeling Using Power Function of Testing Time. *International Journal of Operations and Quantitative Management*, vol. 12, no. 2, pp. 127–140, 2006.

[11] P. K. Kapur, S. Younes, S. Agarwala. Generalized Erlang Software Reliability Growth Model. *ASOR Bulletin*, vol. 14, no. 1, pp. 5–11, 1995.

[12] M. Ohbha. Inflection S-shaped Software Reliability Growth Model. *Stochastic Models in Reliability Theory*, S. Osaki, Y. Hotoyama (eds.), Springer Verlag, Berlin, pp. 144–162, 1984.

[13] P. K. Kapur, A. K. Bardhan, O. Shatnawi. Software Reliability Growth Model with Fault Dependency Using Lag Function. In *Proceedings of International Conference on Quality, Reliability and Control*, IIT Mumbai, vol. 53, pp. 1–7, 2001.

[14] C. Y. Huang, C. T. Lin. Software Reliability Analysis by Considering Fault Dependency and Debugging Time Lag. *IEEE Transactions on Reliability*, vol. 55, No. 3, pp. 436–450, 2006.

[15] W. D. Brooks, R. W. Motley. Analysis of Discrete Software Reliability Models. Technical Report RADC-TR-80-84, Rome Air Development Center, New York, 1980.

[16] J. D. Musa. *Software Reliability Data, Data and Analysis Center for Software*, [online], Available: http://www.dacs.dtic.mil, May 1980.

V. B. Singh received the M.C.A. degree from M.M.M. Engineering College, Gorakhpur, U.P., India. He is a lecturer in the Department of Computer Science at Delhi College of Arts and Commerce, University of Delhi, Delhi, India. Presently, he is a Ph.D. candidate at the University of Delhi, Delhi. He has published nine research papers.

His research interests include software testing and software reliability engineering.

Kalpana Yadav received the M.Tech. in computer science and engineering from Guru Jambheshwar University, Hissar, India. She is a lecturer in the Department of Computer Science at Indira Gandhi Institute of Technology, Guru Gobind Singh Indraprastha University, Delhi, India. Presently, she is a Ph.D. candidate at Jiwaji University, Gwalior. She has published nine research papers.

Her research interests include software testing and software reliability engineering.

Reecha Kapur is a research scholar at the Department of Mathematics and Computer Application, Bundelkhand University, Jhansi, India. She did her Post Graduation in mathematics from Bundelkhand University, Jhansi, India. She has published three research papers.

Her research interests include imperfect debugging models in software reliability and its effect on software testing cost.

V. S. S. Yadavalli received his Ph.D. degree from the Indian Institute of Technology in 1982. He is a professor of Industrial & Systems Engineering at the University of Pretoria. He has published over 90 research papers on reliability theory, queueing theory, inventory theory, software reliability, manpower planning, econometric modeling in ISI accredited journals like *IEEE Transactions on Reliability*, *Microelectronics and Reliability*, *Stochastic Analysis and Applications*, *International Journal of Systems Science*, *Asia Pacific Journal of Operational Research*, *Applied Mathematics & Computation*, *South African Computer Journal*, *South African Journal of Industrial Engineering*, *International Journal of Computers & Industrial Engineering*, *etc.* He is in the editorial board of *Asia Pacific Journal of Opertional Research*, *Management Dynamics*, *South African Journal of Industrial Engineering*. He has been recently listed in *Marquis Who's Who* (23rd edition).

His research interests include reliability theory, queueing theory, inventory theory, software reliability, manpower planning, and econometric modeling.