Quantitative Software Reliability Modeling from Testing to Operation

Chin-Yu Huang[1], Sy-Yen Kuo[1], Michael R. Lyu[2], and Jung-Hua Lo[1]

[1]*Department of Electrical Engineering*
National Taiwan University
Taipei, Taiwan
sykuo@cc.ee.ntu.edu.tw

[2]*Computer Science & Engineering Department*
The Chinese University of Hong Kong
Shatin, Hong Kong
lyu@cse.cuhk.edu.hk

Abstract

In this paper, we first describe how several existing software reliability growth models based on Non-homogeneous Poisson Processes (NHPPs) can be derived based on a unified theory for NHPP models. Under this general framework, we can verify existing NHPP models and derive new NHPP models. The approach covers a number of known models under different conditions. Based on these approaches, we show a method of estimating and computing software reliability growth during operational phase. We can use this method to describe the transitions from testing phase to operational phase. That is, we propose a method of predicting the fault detection rate to reflect changes in the user's operational environments. The proposed method offers a quantitative analysis on software failure behavior in field operation and provides useful feedback information to the development process.

1. Introduction

In recent years, software permeates industrial equipment and consumer products. Software reliability may be the most important quality attribute of application software since it quantifies software failures during the software development process. Since software reliability represents a customer-oriented view of software quality, it relates to practical operation rather than design of program. Therefore, it is dynamic rather than static. The aim and objective of software reliability engineers are to increase the probability that a completed program will work as intended by the customers. Hence, measuring and computing the reliability of a software system are very important. Software reliability measurements can be used for planning and controlling testing resources during development. They can also give us a confidence about the correctness of the completed software.

Research efforts in software reliability engineering have been conducted over the past two decades and many software reliability growth models (SRGMs) have been proposed [1-3]. They are used to evaluate software development status and software engineering technology quantitatively. Basically, SRGMs can help us in estimating the number of initial faults and understanding the effect of faults on software operation. In practice, the software reliability modeling techniques help us in predicting the reliability of software systems, the overall quality, and the optimal software release time [1].

From our previous studies in [4], several existing SRGMs can be unified under a general formulation. A unified theory is very useful for the study of general models without making many assumptions [5-7]. In this paper, we first review the unification of SRGMs based on Non-Homogeneous Poisson Processes (NHPPs). Then we show how these existing SRGMs could be derived by applying the concept of three well-known means: *weighted arithmetic mean*, *weighted geometric mean*, and *weighted harmonic mean* [8]. Some recently proposed SRGMs can also be derived by the same way. Based on this framework, we further present a new general NHPP model incorporating the concept of power transformation into the model unification. From the unified approach, we can derive not only existing NHPP models but also new NHPP models. Furthermore, we discuss some important mathematical relationships and use them to estimate software reliability from testing to operation. We propose an innovative approach to describe the transitions from the testing phase to the operational phase. This approach allows us to understand the software failure behavior during operation. That is, the method can offer a quantitative analysis of failure distribution in the field operation and can also feedback some information to the development process. Consequently, it has a significant potential in predicting and controlling software reliability during operation.

In Section 2, we discuss the unification of SRGMs based on NHPPs. Furthermore, we present a new general NHPP model with the power transformation. Section 3 discusses the parameter estimation and mathematical properties. In Section 4, we estimate and compute the software operational reliability, using the unified theory. Finally, the conclusions are given in Section 5.

2. Non-homogeneous poisson process models

2.1 Reviews of weighted arithmetic, weighted geometric, and weighted harmonic means

Here we first review three well-known means: *arithmetic*, *geometric*, and *harmonic means* [4-5, 7-8]. Let $x \geq 0$ and $y \geq 0$, the *arithmetic mean z* of x and y is defined as

$$z = \frac{1}{2}x + \frac{1}{2}y.$$

More generally, the *weighted arithmetic mean z* of x and y with weights w and $1-w$ is defined as

$$z = wx+(1-w)y, \quad 0<w<1.$$

The *geometric mean z* of x and y is defined as

$$z = \sqrt{xy}.$$

That is,

$$\ln z = \frac{1}{2}\ln x + \frac{1}{2}\ln y.$$

Similarly, the *weighted geometric mean z* of x and y with weights w and $1-w$ is defined as

$$\ln z = w\ln x + (1-w)\ln y, \quad 0<w<1.$$

Finally, the *harmonic mean z* of x and y is defined as

$$\frac{1}{z} = \frac{1}{2x} + \frac{1}{2y},$$

and the *weighted harmonic mean z* of x and y with weights w and $1-w$ is defined as

$$\frac{1}{z} = w\frac{1}{x} + (1-w)\frac{1}{y}, \quad 0<w<1.$$

Based on the concept of weighted mean, we will derive a general discrete NHPP model in the next subsection.

2. 2 A general discrete NHPP model

For the discrete Goel-Okumoto model [1], suppose that the expected number of errors detected per test run is proportional to the current error content of a software system, that is,

$$m(i+1)-m(i) = b(a-m(i)) \tag{1}$$

where $a=m(\infty)$ is the expected number of software errors to be eventually detected and b is the error detection rate per error which is a constant. Taking $w=1-b$, we have

$$m(i+1) = wm(i) + (1-w)a \tag{2}$$

This shows that $m(i+1)$ is expressed by the weighted arithmetic mean of $m(i)$ and a. Now, consider the case that $m(i+1)$ is equal to the weighted geometric mean of $m(i)$ and a with weights w and $1-w$, then

$$\frac{1}{m(i+1)} = w\frac{1}{m(i)} + (1-w)\frac{1}{a} \tag{3}$$

where $0<w<1$ and $a>0$.

Next, consider the case that $m(i+1)$ is equal to the weighted harmonic mean of $m(i)$ and a with weights w and $1-w$, then

$$\ln m(i+1) = w\ln m(i) + (1-w)\ln a \tag{4}$$

where $0<w<1$ and $a>0$.

More generally, let g be a real-valued and strictly monotonic function and $m(i+1)$ be equal to the quasi-arithmetic mean of $m(i)$ and a with weights w and 1-w, respectively, then

$$g(m(i+1)) = wg(m(i)) + (1-w)g(a) \tag{5}$$

where $0<w<1$ and $a>0$.

If w in Eq. (5) is not a constant for all i, and let $m(i+1)$ be equal to the quasi-arithmetic mean of $m(i)$ and a with weights $w(i)$ and $1-w(i)$, respectively, then Eq. (5) can be generalized as

$$g(m(i+1)) = w(i)g(m(i)) + (1-w(i))g(a) \tag{6}$$

where $0<w(i)<1$ and $a>0$.

Solving Eq. (6), we yield

$$\begin{aligned} g(m(i)) &= \prod_{j=1}^{i} w(j)g(m(0)) + (1-\prod_{j=1}^{i} w(j))g(a) \\ &= u_i g(m(0)) + (1-u_i)g(a) \end{aligned} \tag{7}$$

where $u_0 = 1$ and $u_i = \prod_{j=1}^{i} w(j)$ for $i \geq 1$.

Therefore,

$$m(i) = g^{-1}(u_i g(m(0)) + (1 - u_i)g(a)) \qquad (8)$$

is the general discrete NHPP model.

2.3 A general continuous software reliability growth model

In this subsection, we will discuss a general continuous NHPP model. Similar to the above discussion in the discrete case, let $m(t+\Delta t)$ be equal to the quasi-arithmetic mean of $m(t)$ and a with weights $w(t, \Delta t)$ and $1-w(t, \Delta t)$, then

$$g(m(t+\Delta t)) = w(t, \Delta t)\, g(m(t)) + (1-w(t, \Delta t))g(a) \quad (9)$$

where $0<w(t, \Delta t)<1$ and g is a real-valued, strictly monotonic, and differentiable function.

That is,

$$\frac{g(m(t+\Delta t)) - g(m(t))}{\Delta t} = \frac{1 - w(t, \Delta t)}{\Delta t}(g(a) - g(m(t))) \qquad (10)$$

where $0< w(t, \Delta t)<1$.

Suppose $(1-w(t, \Delta t))/\Delta t \rightarrow b(t)$ as $\Delta t \rightarrow 0$, we get the differential equation

$$\frac{\partial}{\partial t} g(m(t)) = b(t)(g(a) - g(m(t))) \qquad (11)$$

For $g(x) = x$ in Eq. (11) (i.e., considering the weighted arithmetic mean considered), then

$$\frac{\partial}{\partial t} g(m(t)) = b(t)(a - m(t)). \qquad (12)$$

Here, $b(t)$ is the error detection rate per error. Furthermore, if $b(t) = b$, then the Goel-Okumoto model can be derived from Eq. (12). The differential equations for $g(x) = \ln x$ and $g(x) = 1/x$ can also be derived from Eq. (11), respectively.

Theorem 1: Let $\frac{\partial}{\partial t} g(m(t)) = b(t)(g(a) - g(m(t)))$, where g is a real-valued, strictly monotonic, and differentiable function. We have [4]

$$m(t) = g^{-1}(g(a) + \{g(m(0)) - g(a)\} \exp[-B(t)]) \quad (13)$$

and $B(t) = \int_0^t b(u)du$.

Corollary 1: Based on the weighted arithmetic mean, take $g(x) = x$ in Eq. (13) and let $k = 1-m(0)/a$, then

1. $m(t) = a(1 - k\exp[-B(t)])$, $a>0$, $0 < k \leq 1$.
2. $\lambda(t) = akb(t)\exp[-B(t)]$
3. $R(t|s) = \exp[-ak(\exp[-B(s)]-\exp[-B(t+s)]]$
4. $d(t) = b(t)$

Corollary 2: Based on the weighted geometric mean, take $g(x) = \ln x$ in Eq. (13) and let $k = m(0)/a$, then

1. $m(t) = ak^{\exp[-B(t)]}$, $a>0$, $0<k<1$.
2. $\lambda(t) = -a(\ln k)b(t)\mathrm{ex}\pi[-B(t)]k^{\exp[-B(t)]}$
3. $R(t|s) = \exp[-a(k^{\exp[-B(t+s)]} - k^{\exp[-B(s)]}]$
4. $d(t) = b(t)m(t)(\ln\alpha - \ln m(t))/(\alpha - m(t))$
5. If $b(t)$ is non-decreasing in t, then $d(t)$ is non-decreasing in t.

Corollary 3: Based on the weighted harmonic mean, take $g(x) = 1/x$ in Eq. (13) and let $k = a/m(0) - 1$, then

1. $m(t) = \dfrac{a}{1 + k\exp[-B(t)]}$, $a>0$, $0 < k \leq 1$.
2. $\lambda(t) = \dfrac{akb(t)\exp[-B(t)]}{(1 + k\exp[-B(t)])^2}$
3. $R(t|s) = \exp[-a \times (\dfrac{1}{1 + k\exp[-B(t+s)]} - \dfrac{1}{1 + k\exp[-B(s)]})]$
4. $d(t) = b(t)m(t)/a$

We have already shown that several classical SRGMs based on NHPPs can be directly derived from Corollary 1, 2, or 3 in [4]. They are *Goel-Okumoto model*, *Gompertz growth curve*, *Logistic growth curve*, *Goel generalized NHPP model*, *Delayed S-shaped model*, *Inflected S-shaped model*, *Modified Duane model*, *Two types of software errors model*, and *Weibull-type testing-effort function model* [1-4]. In the following we will show that other new SRGMs which are proposed recently can also be derived from these corollaries.

- **Log-logistic software reliability growth model**

This model was proposed by Gokhale and Trivedi [9]. They tried to offer a decomposition of the mean value function of a finite failure NHPP model, which can capture the increasing/decreasing nature of the hazard function. That is, the increasing/decreasing behavior of the failure occurrence rate per fault can be captured by the hazard function of the log-logistic distribution.

Take $g(x) = 1/x$ and $b(t) = k/t$, then from Corollary 3,

$$m(t) = a \times \frac{ct^k}{1 + d(ct^k)}, a > 0, c > 0, d > 0, k > 0. \quad (14)$$

Let $c = \lambda^k$ and $d = 1$, we get the log-logistic form:

$$m(t) = a\frac{\lambda^k t^k}{1 + \lambda^k t^k} = a\frac{(\lambda t)^k}{1 + (\lambda t)^k} \quad (15)$$

■ SRGM with logistic testing-effort function

In the field of software reliability modeling, Musa first discussed the validity of execution time theory by taking data sets from real software systems, as testing effort can be faithfully represented by execution time [2-3]. However, most existing software reliability models do not take testing effort into consideration. Recently, we proposed a simple and new software reliability growth model with logistic testing-effort function [11-12]. This model attempts to account for the relationship among the calendar testing, the amount of testing-effort, and the number of software faults detected during testing. The testing-effort can be measured as the human power, the number of test cases, the number of CPU hours, ..., etc. The mean value function $m(t)$ can be described as follow:

$$m(t) = a(1 - \exp[-b(W(t) - W(0))])$$
$$= a(1 - \exp[-b\left\{\frac{N}{1 + A\exp[-\alpha t]} - \frac{N}{1 + A}\right\}]) \quad (16)$$

where a is the expected number of initial faults, b is the error detection rate per unit testing-effort at testing time t that satisfies $b>0$, N is the total amount of testing effort to be eventually consumed, α is the consumption rate of testing-effort expenditures, and A is a constant.

If we take $g(x) = x$ and $b(t) = \frac{bNA\alpha e^{-\alpha t}}{(1 + Ae^{-\alpha t})^2}$, from Corollary 1, we have

$$m(t) = a(1 - k\exp[-b\left\{\frac{N}{1 + A\exp[-\alpha t]} - \frac{N}{1 + A}\right\}]). \quad (17)$$

Furthermore, if $k=1$, then

$$m(t) = a(1 - \exp[-b\left\{\frac{N}{1 + A\exp[-\alpha t]} - \frac{N}{1 + A}\right\}])$$

In addition to the previously mentioned three known means, we propose a more general transformation which includes a parametric family of power transformations:

$$g(x) = \begin{cases} \frac{x^\alpha - 1}{\alpha}, & \alpha \neq 0 \\ \ln x, & \alpha = 0 \end{cases} \quad (18)$$

Note $g(x)$ is only one of many parametric families of transformations that can be applied for dada analysis.

Corollary 4: Based on the power transformation, if we take $g(x) = \begin{cases} \frac{x^\alpha - 1}{\alpha}, & \alpha \neq 0 \\ \ln x, & \alpha = 0 \end{cases}$ into Eq. (13) and $k = 1 - \left(\frac{m(0)}{a}\right)^\alpha$, then

1. $m(t) = a(1 - ke^{-B(t)})^{1/\alpha}, \alpha \neq 0$
2. $\lambda(t) = \frac{1}{\alpha}akB'(t)e^{-B(t)}(1 - ke^{-B(t)})^{\frac{1-\alpha}{\alpha}}, \alpha \neq 0$
3. $R(t \mid s) = \exp\{a[(1 - k^{-B(s)})^{1/\alpha} - (1 - ke^{-B(t+s)})^{1/\alpha}]\}, \alpha \neq 0$
4. $d(t) = \frac{\frac{1}{\alpha}kB'(t)e^{-B(t)}(1 - ke^{-B(t)})^{\frac{1-\alpha}{\alpha}}}{1 - (1 - k^{-B(t)})^{1/\alpha}}$,

where $k = 1 - \left(\frac{m(0)}{a}\right)^\alpha$

If $\alpha = 0$, then the result is the same as Corollary 2.

Proof:

(i) Since $g(x) = \frac{x^\alpha - 1}{\alpha}$, $\alpha \neq 0$, that is,

$$g^{-1}(y) = (\alpha y + 1)^{\frac{1}{\alpha}}.$$

From Eq. (13), we know:

$$m(t) = g^{-1}(g(a) + \{g(m(0)) - g(a)\}\exp[-B(t)]).$$

Therefore, through some simple calculations, we obtain

$$g(a) + (\frac{m(0)^\alpha - a^\alpha}{\alpha})\exp[-B(t)]$$

$$= \frac{a^\alpha - 1}{\alpha} + k' \times \exp[-B(t)]$$

where $k' = \frac{m(0)^\alpha - a^\alpha}{\alpha}$, $g(m(0)) = \frac{m(0)^\alpha - 1}{\alpha}$,

and $g(a) = \frac{a^\alpha - 1}{\alpha}$.

Consequently,

$$m(t) = g^{-1}(g(a) + \{g(m(0)) - g(a)\}\exp[-B(t)])$$

$$= (\alpha \times (\frac{a^\alpha - 1}{\alpha} + k' \times \exp[-B(t)]) + 1)^{\frac{1}{\alpha}}$$

$$= (a^\alpha + (m(0)^\alpha - a^\alpha) \times \exp[-B(t)])^{\frac{1}{\alpha}}$$

$$= (a^\alpha \times (1 - \frac{1}{a^\alpha} \times (a^\alpha - (m(0)^\alpha) \times \exp[-B(t)]))^{\frac{1}{\alpha}}$$

$$= (a^\alpha \times (1 - k \times \exp[-B(t)]))^{\frac{1}{\alpha}}$$

$$= a \times (1 - k \times \exp[-B(t)])^{\frac{1}{\alpha}},$$

where $k = 1 - \left(\frac{m(0)}{a}\right)^\alpha$

The proofs for (ii) to (v) are straightforward and omitted here.

Altogether, based on the Corollary 4, we can generate some new models with various $b(t)$:

(i) If $b(t) = b$, i.e. $B(t) = bt$, then the new mean value function (MVF) is

$$m(t) = a(1 - e^{-bt})^{1/\alpha} \quad, \alpha \neq 0. \qquad (19)$$

(ii) If $b(t) = bct^{c-1}$, *i.e.* $B(t) = bt^c$, then the new mean value function (MVF) is

$$m(t) = a(1 - ke^{-bt^c})^{1/\alpha}, \alpha \neq 0. \qquad (20)$$

(iii) If $b(t) = \frac{c}{b+t}$, *i.e.* $B(t) = c \ln \frac{b+t}{b}$, then the new mean value function (MVF) is

$$m(t) = a(1 - k(\frac{b+t}{b})^c)^{1/\alpha}, \alpha \neq 0. \qquad (21)$$

(iv) If $b(t) = \frac{b^2 t}{1+bt}$, *i.e.* $B(t) = bt - \ln(1+bt)$, then the new mean value function (MVF) is

$$m(t) = a(1 - (1+bt)ke^{-bt})^{1/\alpha}, \alpha \neq 0. \qquad (22)$$

(v) If $b(t) = \frac{b}{1+ce^{-bt}}$, *i.e.* $B(t) = bt + \ln \frac{1+ce^{-bt}}{1+c}$, then the new mean value function (MVF) is

$$m(t) = a(1 - \frac{k(1+c)e^{-bt}}{1+ce^{-bt}})^{1/\alpha} \quad, \alpha \neq 0. \qquad (23)$$

3. Parameter estimation and model properties

3.1 Maximum likelihood estimation and least squares estimation

Fitting a proposed model to actual failure data involves estimating the model's parameters from the real test data sets. *Maximum likelihood estimation* is one of the most popular estimation techniques. The maximum likelihood technique estimates parameters by solving a set of simultaneous equations and it is easy to derive confidence intervals [27].

Let $\{t_k, k=1, 2, \ldots\}$ denote the sequence of times between successive software failures. Then t_k is the time between $(k\text{-}1)^{\text{th}}$ and k^{th} failure. Let S_k denote the time to failure k, then

$$S_k = \sum_{i=1}^{k} t_k .$$

For a given sequence of software failure times $S = (S_1, S_2, \ldots, S_n)$, the joint density or the likelihood function of $S_1, S_2, \ldots, S_n$ can be written as

$$f_{S1,S2,\ldots,Sn,}(S1, S2, \ldots, Sn) = e^{-m(s_n)} \prod_{i=1}^{n} \lambda(s_i)$$

Thus the log likelihood in the case of power transformation of NHPP model can be written as:

$$L(a, \alpha, k|s) = -a(1 - ke^{-B(s_n)})^{1/\alpha} + n \log ak +$$
$$\sum_{i=1}^{n} \log B'(s_i) + \frac{1-\alpha}{\alpha} \sum_{i=1}^{n} \log(1 - ke^{-B(s_i)}) \qquad (24)$$

Maximizing the above equation with respect to a, k and α, we have:

- $\frac{\partial L}{\partial a} = 0$, that is

$$\frac{n}{a} = (1 - ke^{-B(s_n)})^{1/\alpha} \tag{25}$$

- $\frac{\partial L}{\partial \alpha} = 0$, that is

$$a\alpha(1 - ke^{-B(s_n)})^{\frac{1-\alpha}{\alpha}} + \sum_{i=1}^{n} \log(1 - ke^{-B(s_i)}) = 0 \tag{26}$$

- $\frac{\partial L}{\partial k} = 0$, that is

$$\frac{n}{k} = \frac{1-\alpha}{\alpha} \sum_{i=1}^{n} \frac{e^{-B(s_i)}}{1 - ke^{-B(s_i)}} \tag{27}$$

Solving these three non-linear equations simultaneously, we can obtain the point estimates of a, k and α. In addition, for *Least Squares Estimation*, generally the data set is given in the form:

$(t_0, m_0), (t_1, m_1), (t_2, m_2), (t_3, m_3), (t_4, m_4), \ldots\ldots, (t_n, m_n)$

where m_j is the total number of faults detected by time t_j.

For this method, the evaluation formula $S(a, r)$ is

$$S(a,r) = \sum_{k=1}^{n} [m_k - m(t_k)]^2 \tag{28}$$

where m_k is the cumulative number of detected faults in a given time interval $(0, t_k]$ and $m(t_k)$ is the estimated cumulative number of detected faults in the mean value function.

Differentiating S with respect to a and r, setting the partial derivatives to zero, and rearranging these terms, we can solve the nonlinear least square problem[1].

3.2 Mathematical properties

From the discussions in Section 2, we know that SRGMs can be used to estimate the number of residual faults and different models can obtain different mean value functions, i.e., the cumulative number of discovered faults. Here, we discuss some important mathematical properties about the proposed mean value function $m(t)$.

A continuous function $f(x)$ is called a ***concave*** curve if $\frac{d^2 f}{dx^2}$ <0, that is, $\frac{df}{dx}$ is decreasing in x, and then we have

[1] Generally, the DNCONF subroutine of IMSL MATH Library can be used to obtain the parameter estimates. However, there are still many mathematical software packages available in workstations or PCs which are easier to use to help us in estimating these parameters.

$$\lim_{h\to 0} \frac{f(x) - f(x-h)}{h} > \lim_{h\to 0} \frac{f(x+h) - f(x)}{h} \tag{29}$$

where $-\infty < x < \infty$ and $h>0$.

If we consider that a discrete function $f(x_i)$ is ***concave***, then Eq. (29) can be modified as

$$\frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} > \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} \tag{30}$$

where $x_{i-1}<x_i<x_{i+1}$.

Furthermore, assuming $x_i - x_{i-1} = x_{i+1} - x_i$ for $i \geq 1$, then Eq. (30) can be reduced to

$$f(x_{i+1})+f(x_{i-1})-2f(x_i)<0, \; i=1,2,\ldots \tag{31}$$

Similarly, a continuous function $f(x)$ is called a ***convex*** curve if $\frac{d^2 f}{dx^2}$ >0, that is, $\frac{df}{dx}$ is increasing in x, and then we have

$$\lim_{h\to 0} \frac{f(x) - f(x-h)}{h} < \lim_{h\to 0} \frac{f(x+h) - f(x)}{h} \tag{32}$$

If we consider that a discrete function $f(x_i)$ is ***convex***, then Eq. (15) can be modified to be

$$\frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} < \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} \tag{33}$$

Furthermore, assuming $x_i - x_{i-1} = x_{i+1} - x_i$ for $i \geq 1$, then Eq. (33) can be reduced to

$$f(x_{i+1}) + f(x_{i-1}) - 2f(x_i) > 0, \; i=1,2,\ldots \tag{34}$$

In addition, a continuous function $f(x)$ is called an ***S-shaped*** curve if there exists an X such that $\frac{d^2 f}{dx^2} \geq 0$ for all $x<X$ and $\frac{d^2 f}{dx^2}$ <0 for all $x \geq X$. Assuming that $x_i - x_{i-1} = x_{i+1} - x_i$ for $i \geq 1$ and by the same argument above, a discrete function $f(x_i)$ can represent an S-shaped curve if there exists a finite integer $I > 1$ such that

$$f(x_{i+1}) + f(x_{i-1}) - 2f(x_i) \geq 0, \text{ for all } i<I$$

and

$$f(x_{i+1}) + f(x_{i-1}) - 2f(x_i) < 0, \text{ for all } i \geq I.$$

The point where $\frac{d^2 f}{dx^2}$ =0 is called the ***inflection*** point of the curve. For example, using the mean value function (MVF) we proposed in Eq. (17), we know this mean value function is positive and monotonically increasing. Therefore, we obtain

$$\frac{d^2m(t)}{dt^2} = ar(\exp[-rW^*(t)]) \times (\frac{dw(t)}{dt} - r(w(t))^2) \quad (35)$$

and

$$\frac{d^2W(t)}{dt^2} = -\alpha^2 NA \times (\exp[\frac{\alpha t}{2}] + A\exp[\frac{-\alpha t}{2}])^{-3} \times$$
$$(\exp[\frac{\alpha t}{2}] - A\exp[\frac{-\alpha t}{2}]) \quad (36)$$

where $W^*(t) = W(t) - W(0)$ and $W(t) = \frac{N}{1 + Ae^{-\alpha t}}$.

The second derivatives of $W(t)$ and $m(t)$ can determine when the inflection point will occur.

Definition 1: Let $\phi(t) = \frac{d}{dt}w(t) - r(w(t))^2$ and we call $\phi(t)$ the inflectional factor of the mean value function $m(t)$ at time t.

That is, by calculating $\phi(t)$, we can know whether the mean value function $m(t)$ of Eq. (17) is a *concave* or *convex* curve in certain interval. Therefore, through some calculations, it is noted that $W(t)$ has only one inflection point at $t^* = \frac{\ln A}{\alpha}$, $m(t)$ has only one inflection point at

$$t_m^* = \frac{1}{\alpha}\ln\left[\frac{A}{2}(\sqrt{b^2N^2 + 4} - bN)\right] \quad (37)$$

and $t_m^* < t^*$.

Similarly, based on Corollary 4, we can generate a new mean value function in Eq. (19), i.e.,

$$m(t) = a(1 - ke^{-B(t)})^{1/\alpha}, \alpha \neq 0.$$

and use it to estimate software reliability.

If we let

$$D(t) = [\frac{d^2B(t)}{dt^2} - (\frac{dB(t)}{dt})^2] - ke^{-B(t)} \times$$
$$[\frac{d^2B(t)}{dt^2} - 2 \times (\frac{dB(t)}{dt})^2],$$

there will exist an η such that $D(\eta) = 0$. That is, we can treat η as the inflection point for this mean value function. In fact, the inflection point can be easily derived by using

$$\frac{d^2m(t)}{dt^2} = ake^{-B(t)}(1 - ke^{-B(t)})^{1/\alpha} \times D(t).$$

For example, if $B(t) = bt$ and $\alpha = 1$ (i.e. the G-O model), then the inflection point occurs when $t = \frac{\ln 2k}{b}$.

4. Software operational reliability estimation

In general, the performance of a software system strongly depends on its execution environment. That is, the reliability of a software system depends on how it is executed. In general, the software execution environment includes the operating system (OS), the hardware platform, the workload, and the operational profile [14]. In practice, the fault detection phenomenon in the operational phase is different from that in the testing phase [14-20]. However, this fact is not distinctly incorporated in many software reliability modeling efforts. Here, we attempt to describe the fault detection behavior using the unified theory of NHPP models during the operational phase. That is, similar to the fault detection process in the software-testing phase, we can formulate and simulate the fault detection process in the operational phase. For example, by re-arranging Eq. (17), the detection rate per remaining fault at testing time t can be described. This represents the *detectability* of a fault for the current fault content [1-3, 14-20]:

$$d(t) = \frac{dm(t)/dt}{a - m(t)} = b \times w(t) \quad (38)$$

where $w(t)$ is current testing-effort consumption at time t, b is the fault detection rate (FDR) per unit testing-effort

By using Eq. (38), Fig. 1 describes the relationship between the fault detection and the testing-effort expenditures for Ohba's data set [21]. It is noted that the software development process is in the testing phase before the 19th week. However, after the 19th week, anytime can be the optimal release time depending on the cost, reliability, and/or cost-reliability requirements. Thus the detectability of faults during software testing and operational phase may be assumed as [10]

$$b \times w(t) \geq b_{op} \times w_{op}(s) \quad (39)$$

where b_{op} is the fault detection rate per unit operation-effort, $w_{op}(s)$ is the operation-effort expenditures consumed at time s, and s is the time interval in the operational phase.

Note the values of b_{op} and $w_{op}(s)$ can be estimated and computed from the past experience before the operational phase begins. Suppose that if b_{op} is half of b, i.e.,

b_{op}=0.5 × b, then we find that the slope of fault detectability $d(t)$ decreases very slowly. The reason is that the remaining faults of a software system are relatively difficult to find. On the other hand, if this software is very popular in the market, there may be many "excellent testers" to test and debug it. Hence, the $d(t)$ will decrease more rapidly. However, $w_{op}(s)$ is usually assumed to be a constant [10]. Hence, fault detectability in the operational phase is dominated by b_{op}, and b_{op} is significantly affected by b. Consequently, we can conclude that b_{op} and b are the key parameters in modeling software reliability during software development process.

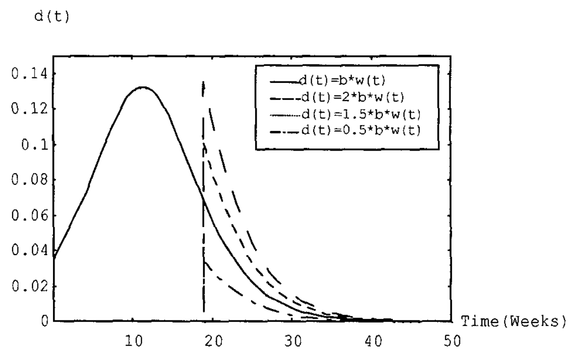


Figure 1: The detectability of faults during software testing and operational phase for Ohba's data set.

It has been widely conceived that the operational profile of a software system may be essentially different from the profile used during testing [14-20]. In any case, we wish to establish a simple link between testing and operation to model software reliability. Among various NHPP SRGMs, the two most important parameters are *the number of initial faults* and *the fault detection rate*. The number of initial faults is the number of faults in the software at the beginning of test and this value will not normally be changed except for imperfect debugging when switching from software testing to operation. We do not need to assume perfect debugging here since it is beyond the scope of this paper. In contrast, the fault detection rate is used to measure the effectiveness of detecting faults by test techniques and test cases, which is more likely to change. Therefore, we concentrate on the FDR transformation from the testing phase into the operational phase.

In order to describe the possible fault detection process, to derive the best-fitted mean value function, and to provide accurate estimates of needed parameters during operation, we now introduce a new approach to estimate software reliability during the operational phase.

Definition 2: Let $T : R \rightarrow R$ be a projection, which indicates the transformation of fault detection rate function from software testing to operation.

Based on the unified theory of general continuous NHPP models in the previous section, we can obtain several useful transformations of FDR function from software testing to operation. Under the assumption of using the same $g(x)$ (if the original FDR is a constant), then we can compute the best-fitted FDR function in order to reflect the user's operational environment. Based on Theorem 1 and Definition 2, let s denote the interval and b denote the FDR, we obtain the following scenarios for describing the software operational reliability transformation from testing to operation.

Case 1: If $T(s \mid b) = b$, that is, the FDR is unchanged when software life cycle proceeds to the operational phase, then we can get the estimated mean value function during software execution as

$$m(s) = a(1 - k\exp(-bs)), 0 < t_1 < s < \infty \qquad (40)$$

where t_1 is the time to end the testing.

The failure intensity during operation is

$$\lambda_0(s) = abk\exp(-bs). \qquad (41)$$

Case 2: If $T(s \mid b) = bcs^{c-1}$, then we can get the estimated mean value function during operation as

$$m(s) = a(1 - k\exp(-bs^c)),\ 0 < t_1 < s < \infty. \qquad (42)$$

The failure intensity function during operation is

$$\lambda_1(s) = abcks^{c-1}\exp(-bs^c). \qquad (43)$$

Through Eq. (41) and Eq. (43), we know

$$\frac{\lambda_1(s)}{\lambda_0(s)} = \frac{cs^{c-1}}{\exp(bs^c - bs)} \qquad (44)$$

Eq. (44) is an index that shows whether the FDR transformation is reasonable or not. Choosing c>1, we can get a proper transformation of FDR function from testing to operational phase, where $\lambda_1(s)$ decreases more rapidly than $\lambda_0(s)$, meaning that a transformation from $\lambda_0(s)$ (in testing) to $\lambda_1(s)$ (in operation).

Case 3: If $T(s \mid b) = \dfrac{b}{1 + c\exp(-bs)}$, then we can get the estimated mean value function during operation as

$$m(s) = a(1 - k\frac{(1+c)\exp(-bs)}{1 + c\exp(-bs)}), 0 < t_1 < s < \infty \,. \quad (45)$$

The failure intensity function during operation is

$$\lambda_2(s) = abk\exp(-bs)\frac{1+c}{1 + c\exp(-bs)} \quad (46)$$

Through Eq. (41) and Eq. (46), we know

$$\frac{\lambda_2(s)}{\lambda_0(s)} = \frac{1+c}{1 + c\exp(-bs)} \quad (47)$$

Since $0 \le \exp(-bs) \le 1$, Eq. (47) is bigger than 1, indicating a transformation from $\lambda_0(s)$ to $\lambda_2(s)$ could not be reasonable.

Case 4: If $T(s \mid b) = \dfrac{b^2 s}{1 + bs}$, then we can get the estimated mean value function during operation as

$$m(s) = a(1 - k(1 + bs)\exp(-bs)), 0 < t_1 < s < \infty \,. \quad (48)$$

Therefore, the failure intensity function during operation is

$$\lambda_3(s) = ab^2 ks\exp(-bs) \quad (49)$$

Similarly, through Eq. (41) and Eq. (49), we know

$$\frac{\lambda_3(s)}{\lambda_0(s)} = bs \quad (50)$$

Again, we note that the above transformation may not be reasonable for general case.

Figure 2 depicts the ratio of failure intensity functions from software testing to operational phase for the Ohba's data set [21]. From Fig. 2(a), we find that the transitions from case 1 to case 2 are reasonable as the ratio of $\lambda_1(s)/\lambda_0(s)$ show a decreasing trend. Note that our new models during operational phase are isomorphic to their original models during testing. Most published papers assumed that the operational mean value function has the same structure compared with the mean value function during testing [14-20]. The main differences among these schemes are in the estimates of key parameters. Basically existing models first compute the number of remaining faults at the end of software testing phase and then adjust the notations and re-compute the values of the parameters [19]. Our approaches, on the other hand, do not require these computations. We only need to get the possible FDR transition (projection) function and understand the nature attribute of the software product during testing and operation according to previous software development experiences or software releases.

In addition, Fig. 2(b) and Fig. 2(c) depict the ratio of failure intensity function transitions from software testing to operational phase when

$$\frac{\lambda_2(s)}{\lambda_0(s)} = \frac{2.25}{1 + 1.25\exp(-0.04272 \times s)} \quad (51)$$

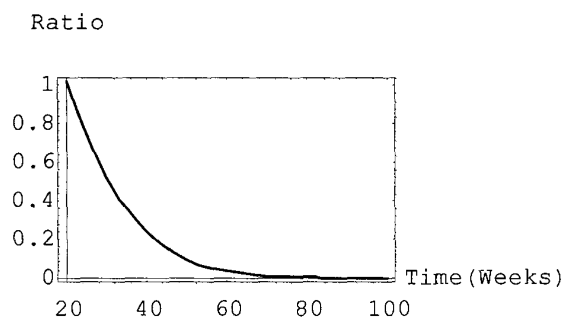
and

$$\frac{\lambda_3(t)}{\lambda_0(t)} = 0.0427223 \times s \,. \quad (52)$$

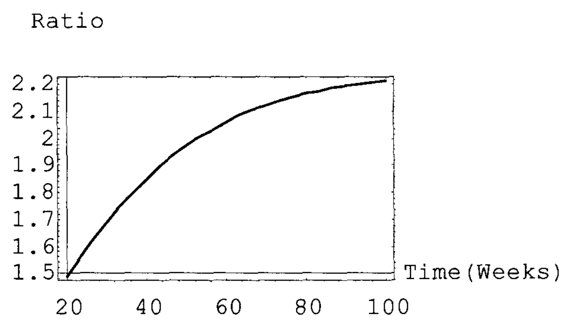
Through these two figures and Eq. (47) and Eq. (50), it is obvious that these two transformations are not reasonable for general case. Finally, so far we have illustrated our approach for $g(x)=x$ and four possible FDR transformations:

- $T(s \mid b) = b$
- $T(s \mid b) = bcs^{c-1}$,
- $T(s \mid b) = \dfrac{b}{1 + c\exp(-bs)}$
- $T(s \mid b) = \dfrac{b^2 s}{1 + bs}$

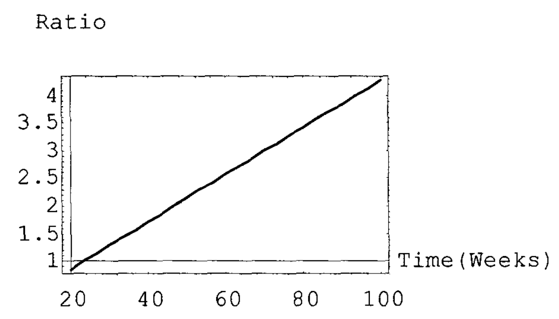
However, following the similar procedures and steps, different $g(x)$ with different FDR transformations can also be derived. Using these methods, software developers can obtain an early estimate of software failure behavior distribution. Besides, we can predict the FDR of discovering remaining fault contents in software during operation. Maintenance activities can thus be planned accordingly.



$$(a)\frac{\lambda_1(s)}{\lambda_0(s)} = \frac{1.25 \times s^{0.25}}{\exp(0.0427223 \times s^{1.25} - 0.0427223 \times s)}$$



$$(b)\frac{\lambda_2(s)}{\lambda_0(s)} = \frac{2.25}{1 + 1.25\exp(-0.0427223s)}$$



$$(c)\frac{\lambda_3(t)}{\lambda_0(t)} = 0.0427223 \times s$$

Figure 2: Ratio of failure intensity functions from software testing to operational phase for the Ohba's data set.

5. Conclusions

In this paper, we first reviewed how several existing software reliability growth models based on Non-homogeneous Poisson Processes could be derived by adopting the concept of weighted arithmetic, weighted geometric, and weighted harmonic means. Then we proposed a more general NHPP model using the power transformation. With the proposed unified theory, many existing NHPP SRGMs can be derived. Our goal is not to add one more model to the already large collection of SRGMS, but to emphasize new approaches for model development and classification. We showed that most existing NHPP SRGMs could be treated as special cases of our general NHPP model. We discussed parameter estimation and related mathematical properties of our new model. Besides, we also proposed a method of computing the operational mean value function and the software operational reliability. One major contribution of this paper is an integrated approach for a comprehensive reliability growth modeling effort during the testing phase and the operational phase. We established an easy and simple link to model possible FDR transformations between these phases. Our approaches of describing the working status of various software operational environments are flexible as we can model various environments ranging from an exponential NHPP to an S-shaped SRGM curve. Based on the integrated theoretical foundation, the technologies and approaches presented in this paper offer a unified and consistent software reliability evaluation scheme from testing to operation.

6. Acknowledgments

We would like to express our gratitude for the support of the National Science Council, Taiwan, R.O.C., under Grant NSC 87-TPC-E-002-017. The work described in this paper was partially supported by a grant from the Research Grant Council of the Hong Kong Special Administrative Region (Project No. CUHK4432/99E), and by a grant from France/Hong Kong Joint Research Scheme 1999/2000. Besides, the authors also thank several anonymous referees for their constructive reviews and comments.

References

[1] M. R. Lyu (1996). *Handbook of Software Reliability Engineering*. McGraw Hill.

[2] J. D. Musa, A. Iannino, and K. Okumoto (1987). *Software Reliability, Measurement, Prediction and Application.* McGraw Hill.

[3] J. D. Musa (1998). *Software Reliability Engineering: More*

Reliable Software, Faster Development and Testing. McGraw-Hill.

[4] R. H. Hou, S. Y. Kuo, and Y. P. Chang, "On a Unified Theory of Some Nonhomogenous Poisson Process Models for Software Reliability," *Proceedings of the 1998 International Conference on Software Engineering: Education & Practice* (SEEP'98), pp. 60-67, Jan. 1998, Dunedin, New Zealand.

[5] N. Langberg and N. D. Singpurwalla, "A Unification of Some Software Reliability Models", *SIAM J. Sci. Stat. Comput.*, Vol. 6, No.3, pp. 781-790, 1985.

[6] D. R. Miller, "Exponential Order Statistic Models of Software Reliability Growth," *IEEE Trans. Software Engineering*, Vol. 12, No.1, pp. 12-24, 1986.

[7] M. Trachtenberg, "A General Theory of Software-Reliability Modeling," *IEEE Trans. Reliability*, Vol. 39, No. 1, pp. 92-96, 1990.

[8] P. S. Bullen, D. S. Mitrinovic, and P. M. Vasic, *Means and Their Inequalities*, D. Reidel Publishing Company, Dordrecht, Holland, 1988.

[9] S. S. Gokhale and K. S. Trivedi, "Log-Logistic Software Reliability Growth Model," *Proceedings of the 3rd IEEE International High-Assurance Systems Engineering Symposium*, pp. 34 -41, Nov. 13-14, 1998, Washington, DC.

[10] H. Ohtera and S. Yamada, "Optimum Software-Release Time Considering an Error-Detection Phenomenon during Operation," *IEEE Trans. on Reliability*, Vol. R-39, pp. 596-599, 1990.

[11] C. Y. Huang, J. H. Lo and S. Y. Kuo, "A Pragmatic Study of Parametric Decomposition Models for Estimating Software Reliability Growth," *Proceedings of the 9th International Symposium on Software Reliability Engineering* (ISSRE'98), pp. 111-123, Nov. 4-7. 1998, Paderborn, Germany.

[12] C. Y. Huang, J. H. Lo, S. Y. Kuo, and M. R. Lyu, "Software Reliability Modeling and Cost Estimation Incorporating Testing-Effort and Efficiency," *Proceedings of the 10th International Symposium on Software Reliability Engineering* (ISSRE'99), pp. 62-72, Nov. 1-4, 1999, Boca Raton, FL, U.S.A.

[13] G. E. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*, San Francisco: Holden-Day, 1976.

[14] J. D. Musa, "Operational Profiles in Software Reliability Engineering, " *IEEE Software Magazine*, pp. 14-32, March 1993.

[15] J. D. Musa, "Sensitivity of Field Failure Intensity to Operational Profile Errors," *Proceedings of the 5th International Symposium on Software Reliability Engineering*, pp. 334-337, 6-9 Nov. 1994.

[16] J. D. Musa, "Adjusting Measured Field Failure Intensity for Operational Profile Variation," *Proceedings of the 5th International Symposium on Software Reliability Engineering*, pp. 330-333, 6-9 Nov. 1994.

[17] S. Keene and C. Lane, "Reliability Growth of Fielded Software," *Proceedings Annual Reliability and Maintainability Symposium*, pp. 360-365, Piscataway, N.J. 1993.

[18] G. Q. Kenney, "Estimating Defects in Commercial Software During Operational Use, "*IEEE Trans. on Reliability*, Vol. 42, No. 1, pp. 107-115, 1993.

[19] T. Philip, P. N. Marinos and K. S. Trivedi, "A Multiphase Software Reliability Model: From Testing to Operational Phase, *Technical Report*, TR-96-01, Center for Advanced Computing and Communication, Duke University, January 1996.

[20] Y. Chen, "Modeling Software Operational Reliability via Input Domain-Based Reliability Growth Model, " *Proceedings of the 28th International Symposium on Fault-Tolerant Computing* (FTCS), pp. 314-323, 1998.

[21] M. Ohba, " Software Reliability Analysis Models, "*IBM J. Res. Develop.*, Vol. 28, No. 4, pp. 428-443, July 1984.

[22] Y. Tohma, R. Jacoby, Y. Murata, and M. Yamamoto, "Hyper-Geometric Distribution Model to Estimate the Number of Residual Software Faults, " *Proc. COMPSAC-89*, Orlando, pp. 610-617, September 1989.

[23] G. Q Kenney, and M. A. Vouk, "Measuring Field Quality of Wide-Distribution Commercial Software," *Proceedings of the 3rd International Symposium on Software Reliability Engineering* (ISSRE'92), pp. 351-357, 1992.

[24] R. Chillarege, R. K. Iyer, J. C. Laprie, and J. D. Musa, "Field Failures and Reliability in Operation," *Proceedings of the 4th International Symposium on Software Reliability Engineering* (ISSRE'93), pp. 122-126, 1993.

[25] A. L. Goel, "Relating Operational Software Reliability and Workload: Results from an Experimental Study," *Proceedings Annual Reliability and Maintainability Symposium*, pp. 167-172, Las Vegas, Nevada USA, January 1996.

[26] A. L. Goel, and J. Sjogren, "Software Reliability Assessment During the Operational Phase," *Proc. of the 1st International Symposium on Software Reliability Engineering*, April 1990, Washington, D.C., U.S.A.

[27] K. Tokuno and S. Yamada, "Operational Software Availability: Modeling and Measurement," *Fourth Conferences of the Asian-Pacific Operational Research Societies within IFORS*, Nov. 30-Dec. 4, 1997.