International Scholarly Research Network
ISRN Software Engineering
Volume 2012, Article ID 413090, 12 pages
doi:10.5402/2012/413090

Research Article

Two-Dimensional Software Defect Models with Test Execution History

Tomotaka Ishii and Tadashi Dohi

Department of Information Engineering, Hiroshima University, 1-4-1 Kagamiyama, Higashi-Hiroshima 739–8527, Japan

Correspondence should be addressed to Tadashi Dohi, dohi@rel.hiroshima-u.ac.jp

Received 6 August 2012; Accepted 11 September 2012

Academic Editors: A. Lastovetsky and B. Yang

In general, the software-testing time may be measured by two kinds of time scales: calendar time and test execution time. In this paper, we develop two-dimensional software reliability models with two-time measures and incorporate both of them to assess the software reliability with higher accuracy. Since the resulting software defect models are based on the familiar nonhomogeneous Poisson processes with two time scales, which are the natural extensions of one-dimensional software defect models, it is possible to treat the time data both simultaneously and effectively. We investigate the dependence of test-execution time as a testing effort on the software reliability assessment and validate quantitatively the software defect models with two-time scales. We also consider an optimization problem when to stop the software testing in terms of two-time measurements.

1. Introduction

The reliable software plays a central role to develop the dependable and high assurance computer-based systems. Since the debugging cycle times of software are often reduced due to smaller release time requirements, the accurate estimation of software reliability tends to be more important day by day, especially, in the earlier testing phase. Software defect models (SDMs) are used to measure the quantitative software reliability and to control the software testing process effectively. Since the software reliability is defined as the probability that software failures caused by faults do not occur for a specified time period, the time evolution of failure occurrence process (fault detection process in software testing) should be modeled by any stochastic counting process. In fact, during the last three decades since the seminal contribution by Jelinski and Moranda [1], a huge number of SDMs have been extensively developed by many authors to help us in estimating the number of initial fault contents and understanding the effect of errors on software operation as well as in predicting the software reliability [2–5]. These are characterized by modeling the software intensity function which implies the instantaneous debugging rate of software faults in the testing phase and is equivalent to the transition rate of stochastic point process.

When one counts the number of software faults detected in the testing phase, the cumulative number of faults is described by a generalized pure birth process with time and/or state dependent transition rate [6, 7]. If the software intensity function depends on only the number of remaining faults, it is reduced to a time-homogeneous continuous-time Markov chain (CTMC). The Jelinski and Moranda SRM [1] is the simplest example of time-homogeneous CTMCs. On the other hand, Goel and Okumoto [8] develop an alternative CTMC model with time-dependent transition rate, based on the nonhomogeneous Poisson process (NHPP). Since the NHPP has a simple mathematical structure and its analytical treatment is quite easy, many SDMs belonging to this category are developed under different software debugging scenarios with deterministic intensity functions. Langberg and Singpurwalla [9] provided a bridge between Jelinski and Moranda SRM [1] and Goel and Okumoto SRM [8] from the Bayesian point of view. Miller [10] extended the Langberg and Singpurwalla's idea to more general cases with the concept of exponential order statistics. Gokhale and Trivedi [11] took account of the testing coverage and propose a different unification framework of the NHPP-based SRMs. Apart from the sophisticated probabilistic approaches, Huang et al. [12] explained the deterministic behavior of NHPP-based SDMs, namely, the mean value functions, by introducing several kinds of mean operations. Pham and Zhang [13] solved a generalized differential equation by which the mean value function is governed and proposed a unified NHPP-based SDM with many parameters.

The above modeling frameworks may unify the NHPP-based SDMs, but never refer to how to use them effectively in practice. For instance, the software testing time may be usually measured by two kinds of time scales: calendar time (hour, day, week, etc.) and test-execution time (CPU second, CPU minute, CPU hour, etc). Musa et al. [3] developed the calendar-time theory and gave a relationship between the calendar-time modeling and the test-execution time modeling in software reliability. Parr [14] used the Rayleigh curve to describe the software development effort (see [15, 16] for the other static approaches). An alternative approach based on the NHPP-based SDMs was proposed by Yamada et al. [17, 18], Huang and Kuo [19], Huang [20], Kuo et al. [21]. They defined the mean value functions of NHPPs as functions of test-execution time and developed some testing-efforts dependent SDMs. These SDMs are quite simple for practical use, but involve several technical problems in terms of parameter estimation and model validation, as we point out them latter. Chen et al. [22] took both execution time, and code coverage into consideration as factors that influence the cumulative number of software failure occurrences. Grottke [23] considered a number of factors influencing the cumulative number of failure occurrences, like calendar time, execution time and code coverage, by specifying consecutive relationships, within the framework of CTMC modeling.

In this paper, we summarize two-dimensional SDMs with two-time measures and incorporate both of them to assess the quantitative software reliability with higher accuracy. Ishii and Dohi [24] developed two-dimensional NHPP-based SDMs and estimate the model parameters with elementary least squares estimation. In the subsequent paper [25], the same authors considered the maximum likelihood estimation for the same SDMs and investigated the goodness-of-fit and the predictive performance with real software fault data. Ishii et al. [26] focused on the discrete-time models with the number of test cases spent for testing and proposed the similar two-dimensional SDMs. The modeling framework developed in [24–26] was widely used for obtaining analogous SDMs [27, 28]. Since the resulting SDMs are based on the familiar NHPPs with two time scales which are the natural extensions of one-dimensional SDMs, it is possible to treat the time data both simultaneously and effectively. In that sense, other two-dimensional SDMs are quite different from Grottke [23]. Also it is worth noting that the two-dimensional SDMs in [25, 26] should be distinguished from the earlier stochastic models developed in the reliability filed [29–33], because they are consistent to the existing one-dimensional NHPP-based SDMs in the literature and involve as special cases. The modeling approach employed here enables us to apply the maximum likelihood method for estimating the model parameters as well as to provide a useful tool to incorporate the testing-effort expenditure as one of the testing matrices. As pointed out by Musa et al. [3], the test-execution time is the best time scale to measure the net testing time and efforts. On the other hand, the software reliability should be defined on the calendar time based on the operational profile, and the testing control to release the software products will be also made on the calendar time. That is, it is essentially important to treat two kinds of time measures in the software reliability assessment. We investigate the dependence of test-execution time as a testing effort on the software reliability assessment and validate quantitatively our new SDMs with two-time scales. We also consider an optimization problem when to stop the software testing in terms of two-time measurements.

2. One-Dimensional Software Defect Modeling

2.1. NHPP-Based SDMs. We make the following assumptions.

(A-1) The software fault detected in testing phase is instantly fixed and removed.

(A-2) The number of initial fault contents in the program, $N(\geq 0)$, is given by a nonnegative discrete random variable having the probability mass function (p.m.f.) $g(n) = \Pr\{N = n\}$.

(A-3) The jth software fault ($j = 1, 2, 3, \ldots$) can be detected by independent and identically distributed (i.i.d.) random calendar time T_j, having the cumulative distribution function (c.d.f.) $F(t) = \Pr\{\mathrm{T}_j \leq t\}$.

We call the above c.d.f. *the fault-detection probability* in this paper. From the assumptions (A-1)–(A-3), it can be seen that the total number of software faults detected by an arbitrary calendar time t (day, week, month, etc.) is given by the univariate random variable $X(t)$ having the conditional binomial p.m.f.

$$\Pr\{X(t) = x \mid N = n\} = \binom{n}{x} F(t)^x \{1 - F(t)\}^{n-x}, \tag{1}$$

where

$$\begin{aligned} \mathrm{E}[X(t) \mid N = n] &= nF(t), \\ \mathrm{Var}[X(t) \mid N = n] &= nF(t)\{1 - F(t)\}. \end{aligned} \tag{2}$$

If $F(t)$ is the exponential distribution and $T_{(1)}, T_{(2)}, T_{(3)}, \ldots$ are the exponential order statistics sampled from $F(t)$, then the corresponding stochastic point process $\{X(t), t \geq 0\} = \max\{j \geq 0; t \geq T_{(j)}\}$ is equivalent to the well-known Jelinski and Moranda SDM [1].

In addition, suppose that the initial number of software faults N is unknown and obeys the Poisson distribution with mean ω (>0), that is,

$$g(n) = \frac{\omega^n}{n!} \exp\{-\omega\}. \tag{3}$$

Then the cumulative number of software faults detected by time t is given by

$$\begin{aligned} \Pr\{X(t) = x\} &= \sum_{n=0}^{\infty} \Pr\{X(t) = x \mid N = n\} g(n) \\ &= \frac{\{\omega F(t)\}^x}{x!} \exp\{-\omega F(t)\}. \end{aligned} \tag{4}$$

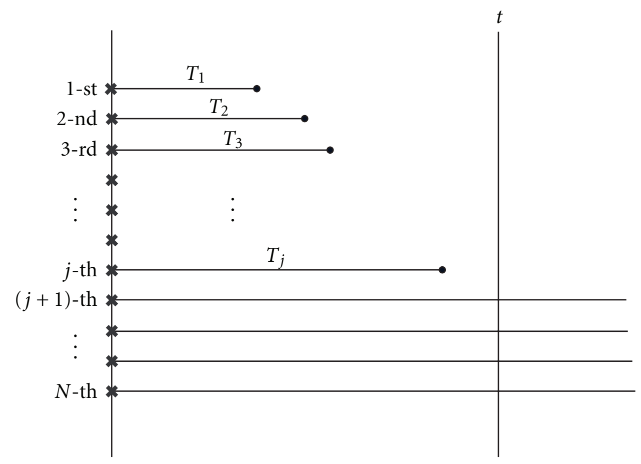


Figure 1: Configuration of one-dimensional software debugging.

This implies that the probabilistic law of $X(t)$ is governed by the well-known nonhomogeneous Poisson process (NHPP) with mean value function $\Lambda(t) = \omega F(t)$, and that $\mathrm{E}[X(t)] = \mathrm{Var}[X(t)] = \Lambda(t)$. By substituting several kinds of fault-detection probabilities into $F(t)$, it is possible to develop a number of NHPP-based SDMs. Goel and Okumoto [8] assumed the exponential distribution $F(t) = 1 - \exp(-at)$ with constant $a(> 0)$ and developed the exponential NHPP-based SDM with $\Lambda(t) = \omega\{1 - \exp(-at)\}$. Yamada et al. [34] and Goel [35] used the two-phase Erlang distribution $F(t) = 1 - (1 + at)\exp(-at)$ and the Weibull distribution $F(t) = 1 - \exp(-at^b)(b > 0)$ and proposed the S-shaped NHPP-based SDM and the generalized exponential NHPP-based SDM with $\Lambda(t) = \omega\{1-(1+at)\exp(-at)\}$ and $\Lambda(t) = \omega\{1 - \exp(-at^b)\}$, respectively. In Figure 1, we show the configuration of one-dimensional software debugging with the i.i.d. fault-detection times to obtain the NHPP-based SDMs.

On the other hand, when the cumulative test-execution time s (CPU time) is observed in the testing phase, it can be used as an alternative of the calendar time, that is, $\Lambda(s) = \omega F(s)$. Musa et al. [3] claimed to use the test-execution time effectively instead of the calendar time in software reliability modeling and pointed out that the execution-time modeling is superior to the calendar-time modeling. However, the execution-time modeling has a limitation to the real software reliability assessment and management. To overcome the problem, Musa et al. [3] developed the calendar-time modeling in order to transform the execution time to the calendar time and gave a relationship (deterministic formula) among test-execution time, calendar time, and resource usage parameters through a simple linear regression analysis. Unfortunately, since this is an intuitive model under a specific testing circumstance, it is not applicable to the general reliability assessment in practice. In other words, the calendar-time modeling by Musa et al. [3] involves some theoretical problems and has seldom been used in the actual software reliability management.

2.2. Testing-Effort Dependent SDMs. It is common to assume that the cumulative test-execution time is regarded as a function of calendar time, that is, $s = W(t)$. Yamada et al. [17, 18] assumed the fault-detection probability $F(t) = 1 - \exp\{-rW(t)\}$ and proposed the testing-effort dependent NHPP-based SDMs, where r (>0) is a constant. More specifically, define the function

$$W(t) = \int_0^t w(x)dx, \qquad W(0) = 0, \tag{5}$$

where $w(t)$ is called *the testing-effort function* and means the testing-effort expenditure spent per unit calendar time. From (5), the function $W(t)$ can be considered as the cumulative testing-effort expenditure consumed up to time t. Yamada et al. [17, 18] further assumed the Weibull testing-effort function

$$w_W(t) = \alpha\beta m t^{m-1} e^{-\beta t^m} \tag{6}$$

with arbitrary parameters α (>0), β (>0), and m (>0). Huang and Kuo [19] and Kuo et al. [21] proposed the logistic testing-effort function

$$w_L(t) = \frac{\alpha}{1 + \beta e^{-mt}}. \tag{7}$$

Let t_j (=j), s_j, and x_j denote the calendar time, the test-execution time, and the cumulative number of software faults detected by the jth calendar week, respectively. Given n fault data (t_j, s_j, x_j) $(j = 1, 2, \ldots, n)$, the parameters (α, β, m) or (α, m) in (6) and (7) are estimated by means of the regression analysis with the data (t_j, s_j), and the remaining parameters (ω, r) are done by the method of maximum likelihood with the data (t_j, x_j). Since this two-stage method is a combination of the least squares method and the maximum likelihood method, the resulting estimates have no significant meaning in terms of statistical theory. That is, the maximum likelihood estimates of (ω, r) with fixed (α, β, m) do not possess the invariant property and/or asymptotical efficiency. In addition, since the testing resource is finite and $\lim_{t\to\infty} W(t) < 1$, it is evident that $\lim_{t\to\infty} F(t) = 1 - \exp(-r) < 1$ and that the fault detection probability is *defective* [36]. This implies that the initial fault contents are given by the Poisson distributed random variable with mean $\omega\{1 - \exp(-r)\}$ and is inconsistent with the model assumptions in II-A. In the testing-effort dependent modeling, the fundamental idea is that the test-execution time $s = W(t)$ is given by a function of the calendar time t. However, if the execution time s can be observed directly, substituting s into $W(t)$ formally leads to $\Lambda(s) = \omega\{1 - \exp(-rs)\}$ which is equivalent to the mean value function of the simple exponential NHPP-based SDM with test-execution time s. Consequently, if one uses only the execution time s, it corresponds to the execution-time modeling based on the exponential NHPP and is independent of the calendar-time data. In this way, the classical testing-effort dependent SDMs [17–19, 21] give no satisfactory modeling framework to deal with two-time measures consistently in the past literature.

3. Two-Dimensional SDMs

3.1. Bivariate Fault-Detection Time Distributions. To cope with two-time scales (t, s) simultaneously, we extend the above univariate binomial modeling framework to the bivariate one. Let T and S be the calendar time (week) and the test-execution time (CPU hr), respectively. Suppose that (T, S) is a continuous bivariate random variable having the bivariate c.d.f. [37, 38]

$$\Pr\{T \le t, S \le s\} = F(t, s), \tag{8}$$

where $F(\infty, \infty) = 1$ and $F(0, s) = F(t, 0) = F(0, 0) = 0$. For convenience, letting

$$F(t, \infty) = F_T(t), \qquad F(\infty, s) = F_S(s) \tag{9}$$

denote the marginal distributions, it is worth noting in the bivariate probability distribution [37] that

$$\Pr\{T > t, S > s\} = 1 - F_T(t) - F_S(s) + F(t, s). \tag{10}$$

Also, it can be easily checked that $\Pr\{T > t, S > s\} \ne 1 - F(t, s)$ and that the bivariate c.d.f. cannot be determined uniquely only by its marginal distributions, that is, the bivariate c.d.f. with the same marginal distributions can be defined infinitely (see [39–43]).

In this paper, we focus on two simplest cases where (T, S) obeys the bivariate exponential distribution by Marshall and Olkin (Marshall-Olkin distribution) [42, 43] and the bivariate Weibull distribution by Lu and Bhattacharyya [44], because the objective here is not to develop a number of SRMs with different fault-detection probabilities.

(i) Marshall-Olkin distribution:

$$F_{\mathrm{MO}}(t, s) = 1 - e^{-(a+c)t} - e^{-(b+c)s} + e^{-at-bs-c\cdot\max(t,s)}. \tag{11}$$

(ii) Bivariate Weibull distribution:

$$F_{\mathrm{LB}}(t, s) = 1 - e^{-(t/\theta_1)^{\beta_1}} - e^{-(s/\theta_2)^{\beta_2}} + e^{-\{(t/\theta_1)^{\beta_1/\delta} + (s/\theta_2)^{\beta_2/\delta}\}^{\delta}}, \tag{12}$$

where a (>0), b (>0), $c \in [0, 1]$, θ_1 (>0), θ_2 (>0), β_1 (>0), β_2 (>0), and $\delta \in (0, 1]$ are arbitrary parameters. Figure 2 depicts the configuration of two-dimensional software debugging with two-time measures.

For the initial fault contents N, let $\{X(t, s), t \ge 0, s \ge 0\}$ be the i.i.d. bivariate random variable and the number of software faults detected by time (t, s). Since each software fault-detection time distribution is given by $F(t, s)$, from the assumptions (A-1)–(A-3), it is immediate to see that

$$\Pr\{X(t, s) = x \mid N = n\} = \binom{n}{x} F(t, s)^x \times \{1 - F(t, s)\}^{n-x},$$
$$\Pr\{X(t, s) = x\} = \frac{\{\omega F(t, s)\}^x}{x!} \exp\{-\omega F(t, s)\}. \tag{13}$$

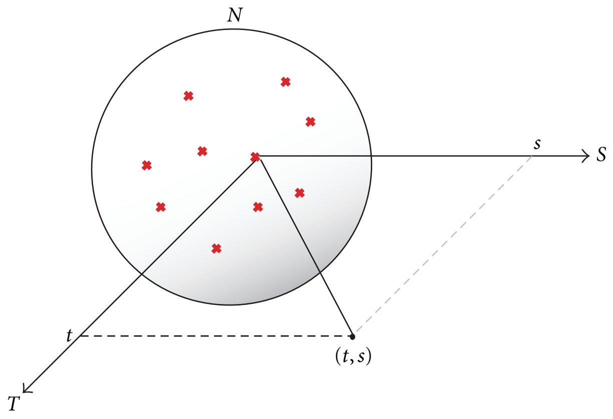


Figure 2: Configuration of two-dimensional software debugging.

This is a two-dimensional NHPP with two-time scales t and s. In this paper, we call this type of NHPP *the two-dimensional SDM* with mean value function $\Lambda(t, s) = \mathrm{E}[X(t, s)] = \mathrm{Var}[X(t, s)] = \omega F(t, s)$.

It is reasonable to assume that the test-execution time is proportional to the number of test cases. In the two-dimensional SDM, if either T or S takes an extremely large value, that is, a number of test cases are consumed during an extremely short period or very a few test cases are tried for an extremely long period, then the fault-detection probability becomes small. This satisfies our intuition for the relationship between the testing time and the number of test cases. It can be found that $c = 0$ and $F(t, s) = F_T(t)F_S(s)$ if T and S are statistically independent from each other. This means that the cumulative number of test cases has no correlation with the length of testing period measured by calendar time and seems to be quite unreasonable. In fact, we expect in modeling that the functions $\mathrm{E}[S \mid T = t]$ and $\mathrm{E}[T \mid S = s]$ are both increasing functions of t and s, respectively. This is the main reason to use the Marshall-Olkin distribution [42, 43], because the other bivariate exponential distributions with explicit form of c.d.f. such as the Gumbel distribution [40] have the negative correlation [27, 28]. In the Marshall-Olkin distribution, we have

$$\begin{aligned} \mathrm{E}[S \mid T = t] &= \int_0^\infty \Pr\{S > s \mid T = t\} ds \\ &= \frac{1}{b} - \frac{c(a+b+c)e^{-bt}}{b(a+c)(b+c)}, \end{aligned} \tag{14}$$

$$\begin{aligned} \mathrm{E}[T \mid S = s] &= \int_0^\infty \Pr\{T > t \mid S = s\} ds \\ &= \frac{1}{a} - \frac{c(a+b+c)e^{-as}}{a(a+c)(b+c)}. \end{aligned} \tag{15}$$

Also, it holds in this case that $\Pr\{T = S\} = c/(a+b+c) > 0$ and that the software fault can be detected with positive probability at time $t = T = s$.

On the other hand, for the bivariate Weibull distribution in (12), we have

$$\begin{aligned}
\Pr\{S > s \mid T = t\} &= \frac{\partial F_{\mathrm{BW}}(t,s)/\partial t}{\partial^2 F_{\mathrm{BW}}(t,s)/\partial t \partial s} \\
&= \frac{1}{\theta_1}\left\{ e^{(t/\theta_1)_1^{\beta} - [(t/\theta_1)^{\beta_1/\delta} + (s/\theta_2)^{\beta_2/\delta}]^{\delta}} \times \beta_1 \left(\frac{t}{\theta_1}\right)^{-1+(\beta_1/\delta)} \right. \\
&\qquad \left. \times \left[\left(\frac{t}{\theta_1}\right)^{\beta_1/\delta} + \left(\frac{s}{\theta_2}\right)^{\beta_2/\delta} \right]^{-1+\delta} \right\}.
\end{aligned} \tag{16}$$

Unfortunately, the explicit forms of $\mathrm{E}[S \mid T = t]$ and $\mathrm{E}[T \mid S = s]$ for the bivariate Weibull distribution are not available, but can be numerically calculated. These regression functions may be used to predict the cumulative test-execution time in future at an arbitrary calendar time, after estimating model parameters in $\Lambda(t,s)$.

3.2. Maximum Likelihood Estimation. In the seminal paper [24], we apply the least squares estimation to determine the model parameters. However, the advantage of the two-dimensional SDMs is to make it easy to estimate the model parameters by means of the maximum likelihood method. Let $x_j\,(j = 1,2,\ldots,n)$ denote the cumulative number of software faults detected by the jth calendar week. Given n fault data $(t_j, s_j, x_j)\,(j = 1,2,\ldots,n)$, the problem is to estimate the model parameter θ, where $\Lambda(t,s;\theta) = \omega F(t,s;\pi)$, $\theta = (\omega,\pi)$, $\pi = (a,b,c)$, and $(\theta_1,\theta_2,\beta_1,\beta_2,\delta)$ for the Marshall-Olkin distribution and the bivariate Weibull distribution, respectively. Then, the likelihood function is given by

$$\begin{aligned}
L(\theta) &= \Pr\{X(t_1,s_1) = x_1, X(t_2,s_2) = x_2, \ldots, X(t_k,s_k) = x_k\} \\
&= \prod_{j=1}^{k} \exp\left\{ -\left[\Lambda\left(t_j,s_j;\theta\right) - \Lambda\left(t_{j-1},s_{j-1};\theta\right) \right] \right\} \\
&\quad \times \frac{\left\{ \Lambda\left(t_j,s_j;\theta\right) - \Lambda\left(t_{j-1},s_{j-1};\theta\right) \right\}^{(x_j - x_{j-1})}}{\left(x_j - x_{j-1}\right)!} \\
&= \exp\{-\Lambda(t_k,s_k;\theta)\} \\
&\quad \times \prod_{j=1}^{k} \frac{\left\{ \Lambda\left(t_j,s_j;\theta\right) - \Lambda\left(t_{j-1},s_{j-1};\theta\right) \right\}^{(x_j - x_{j-1})}}{\left(x_j - x_{j-1}\right)!}.
\end{aligned} \tag{17}$$

Taking the logarithm of both sides of (17) yields the following logarithmic likelihood function:

$$\begin{aligned}
\ln L(\theta) &= \sum_{j=1}^{k} \left(x_j - x_{j-1}\right) \\
&\quad \times \ln\left\{ \Lambda\left(t_j,s_j;\theta\right) - \Lambda\left(t_{j-1},s_{j-1};\theta\right) \right\} \\
&\quad - \Lambda(t_k,s_k;\theta) - \sum_{j=1}^{k} \ln\left\{ \left(x_j - x_{j-1}\right)! \right\},
\end{aligned} \tag{18}$$

where $(t_0, s_0, x_0) = (0,0,0)$.

Since the logarithmic function is monotonically increasing, the maximum likelihood estimate of the parameter θ maximizes the logarithmic likelihood function $\ln L(\theta)$. If the function $\ln L(\theta)$ is strictly concave in θ, then the maximum likelihood estimate has to satisfy the first order condition of optimality $\partial \ln L(\theta)/\partial\theta = 0$, where 0 is the zero vector. The above modeling approach provides a valid parameter estimation procedure based on the maximum likelihood method, without the two-stage procedure based on the least square method.

4. Software Release Planning

Of our next concern is the formulation of the optimal software release problem (e.g.,[45–49]) based on the two-dimensional SDMs. The main objective here is to derive simultaneously the optimal software release timing measured by calendar time and the optimal testing-effort expenditure, which minimize the relevant expected software cost. This leads to the determination of the economic software testing schedule. Define:

t_0: software release time measured by calendar time

s_0: cumulative test-execution time measured by CPU time

k_c: software-testing cost per unit calendar time

k_e: software-testing cost per unit execution time

c_t: software-fault correction/removal cost per unit fault in testing phase

c_o: software-fault correction/removal cost per unit fault in operational phase

L_s: lifetime of software product (given constant).

During the testing phase, two kinds of testing costs proportional to the calendar time and the test-execution time, respectively, are considered, where $k_c t_0$ is the cost to keep the testing team or the testing personnel, and $k_e s_0$ is the cost to develop the test cases. Here, it is assumed again that the development cost of test cases is proportional to the time length of software test-execution time. On the other hand, when one software fault is detected in the testing phase and the operational phase, the unit costs to fix/remove it are given by c_t and c_o, respectively. The expected debugging cost during the testing phase is $c_t \Lambda(t_0, s_0)$ with fixed (t_0, s_0). If the lifetime of software product L_s is given in advance, then the expected number of software faults detected in the operational phase is given by $\Lambda(L_s, s_0) - \Lambda(t_0, s_0)$, that is, the expected debugging cost during the operational phase

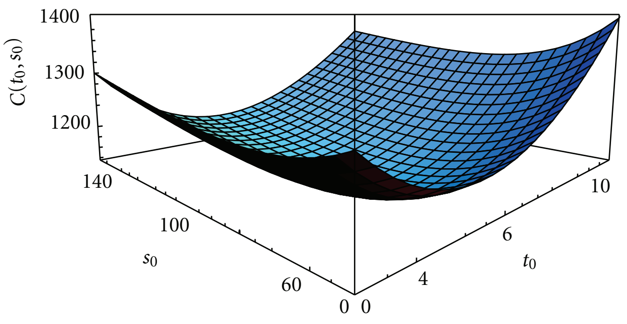


FIGURE 3: Unimodality of the expected total software cost function.

becomes $c_o\{\Lambda(L_s, s_0) - \Lambda(t_0, s_0)\}$. Thus, the expected total software cost during the software life cycle is given by

$$C(t_0, s_0) = k_c t_0 + k_e s_0 + c_t \Lambda(t_0, s_0) + c_o\{\Lambda(L_s, s_0) - \Lambda(t_0, s_0)\}. \quad (19)$$

If $L_s \to \infty$ as a special case, then $\lim_{L_s \to \infty} \Lambda(L_s, s_0) = \omega$ and the underlying optimization problem can be simplified as

$$\min_{(t_0, s_0)} : C(t_0, s_0) = k_c t_0 + k_e s_0 + c_t \Lambda(t_0, s_0) + c_o\{\omega - \Lambda(t_0, s_0)\}. \quad (20)$$

For the expected total software cost during the software life cycle, the optimal software release policy is defined by the optimal pair (t_0^*, s_0^*) which minimizes it. That is, the pair (t_0^*, s_0^*) is the minimizer of $C(t_0, s_0)$ provided that the associated Hessian matrix is positive definite at (t_0^*, s_0^*). In this situation, it must satisfy the first order condition of optimality

$$\left.\frac{\partial C(t_0, s_0)}{\partial t_0}\right|_{t_0 = t_0^*} = \left.\frac{\partial C(t_0, s_0)}{\partial s_0}\right|_{s_0 = s_0^*} = 0 \quad (21)$$

if there exists a unique solution (t_0^*, s_0^*). Fortunately, it can be shown that the two-dimensional minimization problem is rather tractable. Figure 3 illustrates the behavior of expected total software cost function with respect to (t_0, s_0), where $k_c = 100$ (\$), $k_e = 1$ (\$), $c_t = 1$ (\$), $c_o = 20$ (\$), $L_s \to \infty$, and $M(t_0, s_0) = \omega F_{\mathrm{MO}}(t_0, s_0)$ with $\omega = 87.5452$, $a = 0.25202$, $b = 0.03984$, and $c = 0.00803$. It can be easily seen that the function is unimodal in (t_0, s_0) and that there exists a unique optimal software release policy $(t_0^*, s_0^*) = (5.566, 85.547)$. In this simple example, the time length of software testing should be planned with the release time $t_0 = 5.566$ (weeks) ≈ 39 (days) and the test-execution time $s_0 = 85.542$ (CPU hr). When the software delivery schedule is fixed, t_0 is replaced by the delivery time, and the underlying problem is reduced to the simpler one-dimensional minimization problem, $\min_{s_0} C(t_0, s_0)$ with given t_0.

5. Real Data Analysis

5.1. Data Sets. In this section, we consider two real data sets collected in the actual software development projects for the real time command and control systems [50]. We present these data in Tables 1 and 2, where the data set DS no. 1 (DS no. 2) consists of 54 (53) faults count data observed during 17 (16) weeks with total 32.8 (18.5) (CPU hr). More precisely, DS no. 1 is the fault data for a middle size of program with 27.7 KLC of a real-time command control system. On the other hand, DS no. 2 is also the fault data for the similar program with 33.5 KLC.

TABLE 1: Data set DS no. 1.

t_j	s_j	x_j	t_j	s_j	x_j
1	0.053	1	10	16.200	42
2	0.115	2	11	17.100	42
3	0.273	4	12	18.600	46
4	0.354	5	13	20.600	47
5	1.400	13	14	21.800	47
6	3.150	22	15	23.000	49
7	6.110	28	16	25.200	51
8	11.080	35	17	32.800	54
9	11.500	39			

TABLE 2: Data set DS no. 2.

t_j	s_j	x_j	t_j	s_j	x_j
1	0.358	8	9	4.580	48
2	0.690	14	10	6.620	50
3	0.945	17	11	9.980	51
4	1.270	17	12	11.300	51
5	1.670	28	13	13.700	51
6	2.210	35	14	15.300	53
7	2.610	35	15	17.700	53
8	3.760	41	16	18.500	53

5.2. Goodness-of-Fit Test. Using these two data sets, we derive the maximum likelihood estimate of the parameter θ, say $\hat{\theta}$, and perform the goodness-of-fit test of the two-dimensional SDMs with the maximum logarithmic likelihood (LLF), the Akaike information criterion (AIC) and the residual square sum (RSS) at the data observation points of 100%, 75%, 50%, and 25%. Tables 3 and 4 present the goodness-of-fit results for the data sets DS no. 1 and DS no. 2, respectively, where MO, BW, GO, GL, LT, and WT denote the Marshall-Olkin SDM, the bivariate Weibull SDM in the two-dimensional NHPPs, the one-dimensional exponential SDM, the one-dimensional generalized exponential SDM, the one-dimensional logistic testing-effort dependent SDM, and the one-dimensional Weibull testing-effort dependent SDM, respectively. From these results, it can be seen that the two-dimensional SDMs can provide the better goodness-of-fit performance in terms of AIC and RSS in many cases. Of course, it is worth noting that the comparison between the two-dimensional and the one-dimensional SDMs is not fair, because the corresponding AICs mean the approximate distances in the sense of distribution between the real (but unknown) software fault-count process and the probability model with different dimension. In other words,

Table 3: Goodness-of-fit test with DS no. 1.

100%	MO	BW	GO	GL	LT	WT
LLF	−45.266	−33.740	−41.468	−41.247	−43.385	−38.518
AIC	98.532	79.480	86.936	88.493	90.769	81.036
RSS	71.160	9.216	25.513	44.988	27.078	15.730
75%	MO	BW	GO	GL	LT	WT
LLF	−33.334	−25.132	−31.540	−24.431	−26.493	−24.842
AIC	74.668	62.263	67.080	54.861	56.986	53.684
RSS	30.450	2.503	24.673	2.605	6.632	1.784
50%	MO	BW	GO	GW	LT	WT
LLF	−13.358	−12.394	−22.547	−15.423	−14.748	−16.038
AIC	34.716	36.787	49.095	36.846	33.497	36.077
RSS	1.887	0.311	50.024	1.928	1.400	1.142
25%	MO	BW	GO	GL	LT	WT
LLF	−4.327	−4.382	−4.577	−4.504	−4.444	−4.412
AIC	16.653	20.765	13.155	15.009	12.889	12.824
RSS	0.010	0.164	0.094	0.043	0.028	0.044

Table 4: Goodness-of-fit test with DS no. 2.

100%	MO	BW	GO	GL	LT	WT
LLF	−48.967	−34.314	−39.951	−39.016	−37.885	−42.403
AIC	105.933	80.629	83.901	84.032	79.770	88.807
RSS	47.519	3.626	12.010	7.868	6.158	9.100
75%	MO	BW	GO	GL	LT	WT
LLF	−29.559	−28.191	−34.645	−34.570	−32.730	−38.166
AIC	67.117	68.383	73.291	75.140	69.459	80.331
RSS	4.664	3.558	9.203	8.623	6.390	11.790
50%	MO	BW	GO	GL	LT	WT
LLF	−22.231	−22.083	−25.327	−24.996	−25.606	−27.059
AIC	52.463	56.167	54.653	55.992	55.211	58.118
RSS	4.314	4.708	6.252	6.264	6.626	8.731
25%	MO	BW	GO	GL	LT	WT
LLF	−7.307	−5.493	−6.808	−5.866	−6.895	−7.192
AIC	22.614	22.985	17.615	17.732	17.789	18.384
RSS	0.751	0.261	0.700	0.141	0.745	0.970

the goodness-of-fit comparison based on the AIC should be made among the SDMs with the same dimension. Then, the BW can show the better goodness-of-fit performance than the MO in almost cases. In Figures 4 and 5, we show the temporal behavior of estimates for the cumulative number of detected software faults with 100% data of DS no. 1 and DS no. 2, respectively. It is observed that the two-dimensional SDMs clarify the relationship between the calendar time and the testing-effort expenditure and enable measuring the software testing progress effectively.

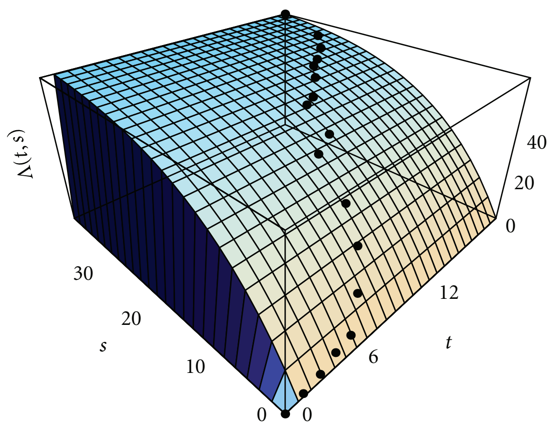
5.3. Prediction Performance. Next, we predict the behavior of the number of software faults detected in the future. Two possible cases are considered: in case 1, the model parameters are estimated at the observation point t_n, but at each calendar time $t = t_{n+1}, t_{n+2}, \ldots$, the testing-effort expenditure is scheduled in advance ($s = s_{n+1}, s_{n+2}, \ldots$) (given) or estimated with the regression function in (14); that is, $s = \hat{s}_{n+1}, \hat{s}_{n+2}, \ldots$ (regression). On the other hand, in case 2, we consider the other situation where after estimating the model parameters at the observation point s_n, the testing-effort expenditure to consume the scheduled test cases, $s = s_{n+1}, s_{n+2}, \ldots$, is known in advance, but the calendar time to complete respective test cases is estimated with the regression function in (18), that is, $t = \hat{t}_{n+1}, \hat{t}_{n+2}, \ldots$. Tables 5 and 6 (Tables 7 and 8) present the prediction squares error (PSE) for estimates of the cumulative number of software faults detected in the future, where PSE is defined by

$$\mathrm{PSE} = \frac{1}{n' - n} \sum_{k=n+1}^{n'} \left[x_k - \Lambda\left(t_k, s_k; \hat{\theta}\right) \right]^2, \qquad (22)$$

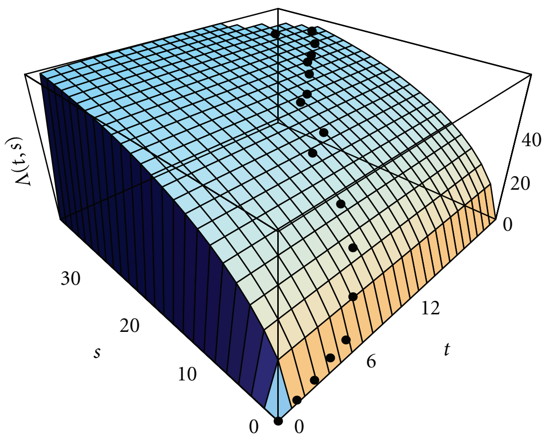
and $n'(> n)$ is the total number of data for analysis. In this example, we set $n' = 17$ and 16 for DS no. 1

Table 5: Prediction error in case 1 with DS no. 1.

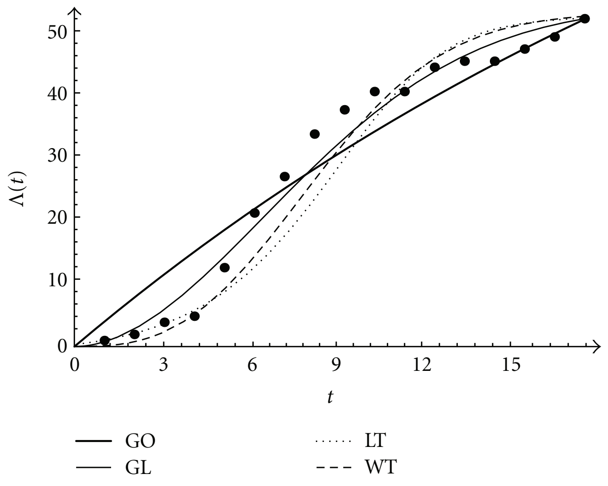
75%	MO	BW	GO	GL	LT	WT
Given	42.218	1.945	71.224	9.731	16.877	10.880
Regression	7.911	18.681				
50%	MO	BW	GO	GL	LT	WT
Given	99.002	103.981	159.551	209.869	119.644	35.762
Regression	120.303	120.310				
25%	MO	BW	GO	GL	LT	WT
Given	645.622	1293.560	724.258	1186.030	1277.290	1158.760
Regression	1333.690	1333.450				



(a) MO



(b) BW



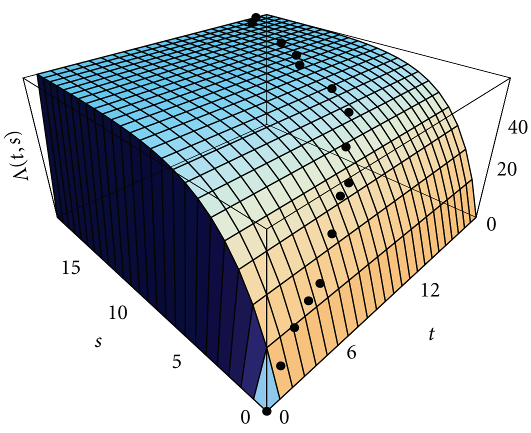
(c) One-dimensional SRMs

Figure 4: Estimates of cumulative number of detected software faults (DS no. 1).

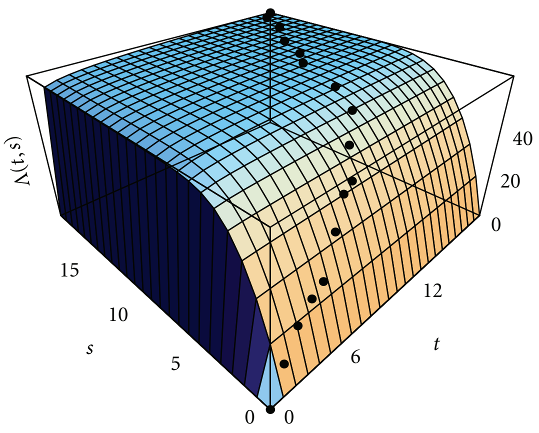
and DS no. 2, respectively. From these results, the two-dimensional SDMs can show the better performance for prediction at the 75% observation point. However, in the other observation points, no remarkable results are found, so that the prediction performance depends on the kind of data.

Next, we assess the quantitative software reliability as the probability that no software failure occurs during a specified time period, where the software reliability is defined by

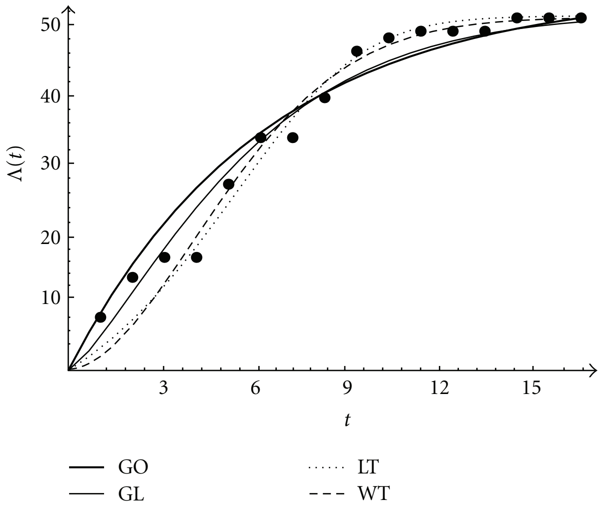
$$R(x \mid t_n, s_n) = \exp\left[-\left\{\Lambda\left(t_n + x, s_n; \hat{\theta}\right) - \Lambda\left(t_n, s_n; \hat{\theta}\right)\right\}\right]. \tag{23}$$



(a) MO



(b) BW



(c) One-dimensional SRMs

Figure 5: Estimates of cumulative number of detected software faults (DS no. 2).

Once the software is released at the calendar time t_n to the user or market, it should be noted that no software test is carried out in general, so that the cumulative testing-effort expenditure is considered as a constant, and the software reliability is defined as a function of only the calendar time. Figures 6 and 7 illustrate the estimation results of software reliability at the 100% observation point. From these figures, it can be found that the two-dimensional SDMs (MO and BW) overestimate the reliability function much more than the typical one-dimensional SDMs (note that the testing-effort dependent SDMs are not available in the situation with the constant testing-effort expenditure after release, because $W(t)$ is an increasing function of the calendar time). Since the corresponding software products are actually released at the observation points, however, it can be concluded that the values of software reliability based on one-dimensional SDMs are too small and rather questionable.

5.4. Optimal Software Release Policy. Finally, we investigate the dependence of cost parameters characterizing the software testing process on the optimal software release policy

Table 6: Prediction error in case 2 with DS no. 1.

	MO	BW	GL
75%	30.661	2.154	4.889
50%	97.368	119.921	111.164
25%	651.142	1333.160	911.021

(Tables 11 and 12). In Tables 9 and 10, we present the optimal software release policies maximizing the expected total software costs with the two-dimensional SDMs proposed in this paper, where the Marshall-Olkin SDM is assumed with the same parameters in Figure 3. In these tables, the policy $(t_0^*, s_0^*) = (0, 0)$ denotes that it is optimal not to carry out the software test economically. Of course, such a situation with relatively small difference between the fault correction/removal costs in the testing and operational phases will not be realistic. As a remarkable point in these tables, as both cost ratios k_c/k_e and c_t/c_o increase, the associated optimal pair (t_0^*, s_0^*) monotonically decrease. To

Table 7: Prediction error in case 1 with DS no. 2.

75%	MO	BW	GO	GL	LT	WT
Given	2.431	0.656	10.252	5.825	1.656	0.863
Regression	2.994	0.691				
50%	MO	BW	GO	GL	LT	WT
Given	9.812	34.861	77.538	114.273	3.240	4.330
Regression	92.790	107.750				
25%	MO	BW	GO	GL	LT	WT
Given	811.770	895.649	842.364	894.170	824.662	795.296
Regression	895.753	677.371				

Table 8: Prediction error in case 2 with DS no. 2.

	MO	BW	GL
75%	2.431	1.844	2.431
50%	9.812	34.861	9.812
25%	811.770	895.890	811.770

Table 9: Dependence of cost ratio k_c/k_e on the optimal software release policy with the Marshal and Olkin SRM: $c_t = 1$, $c_o = 20$.

k_c/k_e	(t_0^*, s_0^*)	$C(t_0^*, s_0^*)$
1/1	(23.284, 91.379)	226.999
5/1	(17.097, 91.155)	304.576
10/1	(14.432, 90.881)	382.635
20/1	(11.766, 90.333)	512.098
50/1	(8.240, 88.640)	804.229
100/1	(5.566, 85.542)	1141.750
200/1	(2.872, 77.581)	1548.600
300/1	(1.225, 63.745)	1749.270

Table 10: Dependence of cost ration c_t/c_o on the optimal software release policy with the Marshal and Olkin SRM: $k_c = 100$, $k_e = 1$.

c_t/c_o	(t_0^*, s_0^*)	$C(t_0^*, s_0^*)$
1/80	(11.098, 119.890)	1724.020
1/50	(9.253, 109.094)	1529.560
1/30	(7.219, 96.576)	1315.360
1/20	(5.566, 85.542)	1141.750
1/15	(4.357, 76.585)	1015.440
1/10	(2.539, 60.175)	829.632
1/5	(0.000, 0.000)	437.726
1/1	(0.000, 0.000)	87.545

Table 11: Dependence of cost ratio k_c/k_e on the optimal software release policy with the bivariate Weibull SRM: $c_t = 1$, $c_o = 20$.

k_c/k_e	(t_0^*, s_0^*)	$C(t_0^*, s_0^*)$
1/1	(12.280, 139.110)	1133.400
5/1	(2.462, 136.827)	1153.190
10/1	(1.232, 135.845)	1161.730
20/1	(0.616, 134.864)	1170.270
50/1	(0.247, 133.569)	1181.570
100/1	(0.123, 132.591)	1190.130
200/1	(0.062, 131.615)	1198.680
300/1	(0.041, 131.044)	1203.690

Table 12: Dependence of cost ratio c_t/c_o on the optimal software release policy with the bivariate Weibull SRM: $k_c = 100$, $k_e = 1$.

c_t/c_o	(t_0^*, s_0^*)	$C(t_0^*, s_0^*)$
1/80	(0.567, 245.157)	3944.060
1/50	(0.344, 202.974)	2593.110
1/30	(0.196, 161.844)	1668.020
1/20	(0.123, 132.591)	1190.130
1/15	(0.088, 113.591)	943.667
1/10	(0.052, 89.176)	687.952
1/5	(0.019, 53.380)	412.309
1/1	(0.000, 0.000)	120.758

the contrary, the expected total software cost with fixed c_t/c_o increases but the cost with fixed k_c/k_e decreases.

6. Conclusions

In this paper, we have developed two-dimensional software reliability models based on bivariate fault-detection time distributions related with calendar time and test execution time and compared their goodness-of-fit and prediction performances with real project data. The numerical examples have suggested that our new SDMs provide better performance than the existing NHPP-based SDMs in many cases.

In the future, we will apply the same technique to the case with many factors influencing the cumulative number of failure occurrences. For instance, when the code coverage is considered as Grottke [23], a different modeling framework will be needed to incorporate it. Also, if the third time measure is introduced like the number of test cases, the higher-dimensional SDMs should be developed to handle such a complex case.

Acknowledgment

The present research was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (C), Grant no. 23510171 (2011–2013).

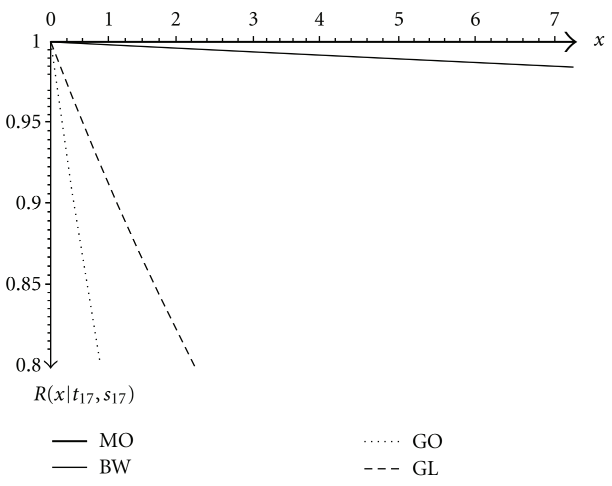


FIGURE 6: Estimating software reliability with DS no. 1.

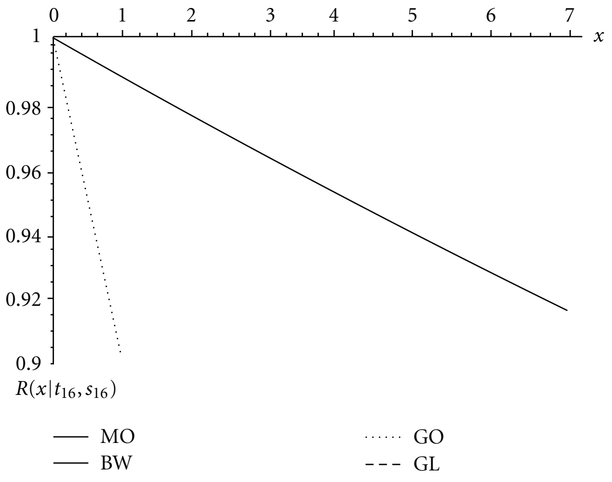


FIGURE 7: Estimating software reliability with DS no. 2.

This work is an extended version of our conference paper [25].

References

[1] Z. Jelinski and P. B. Moranda, "Software reliability research," in *Statistical Computer Performance Evaluation*, W. Freiberger, Ed., pp. 465–484, Academic Press, New York, NY, USA, 1972.

[2] M. R. Lyu, Ed., *Handbook of Software Reliability Engineering*, McGraw-Hill, New York, NY, USA, 1996.

[3] J. D. Musa, A. Iannino, and K. Okumoto, *Software Reliability, Measurement, Prediction, Application*, McGraw-Hill, New York, NY, USA, 1987.

[4] H. Pham, *Software Reliability*, Springer, Singapore, 2000.

[5] N. D. Singpurwalla and S. P. Wilson, *Statistical Methods in Software Engineering*, Springer, New York, NY, USA, 1997.

[6] S. S. Gokhale, M. R. Lyu, and K. S. Trivedi, "Analysis of software fault removal policies using a non-homogeneous continuous time Markov chain," *Software Quality Journal*, vol. 12, no. 3, pp. 211–230, 2004.

[7] J. G. Shanthikumar, "A general software reliability model for performance prediction," *Microelectronics Reliability*, vol. 21, no. 5, pp. 671–682, 1981.

[8] A. L. Goel and 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.

[9] N. Langberg and N. D. Singpurwalla, "A unification of some software reliability models," *SIAM Journal on Scientific and Statistical Computing*, vol. 6, no. 3, pp. 781–790, 1985.

[10] D. R. Miller, "Exponential order statistic models of software reliability growth," *IEEE Transactions on Software Engineering*, vol. 12, no. 1, pp. 12–24, 1986.

[11] S. S. Gokhale and K. S. Trivedi, "A time/structure based software reliability model," *Annals of Software Engineering*, vol. 8, no. 1–4, pp. 85–121, 1999.

[12] C. Y. Huang, M. R. Lyu, and S. Y. Kuo, "A unified scheme of some nonhomogenous poisson process models for software reliability estimation," *IEEE Transactions on Software Engineering*, vol. 29, no. 3, pp. 261–269, 2003.

[13] H. Pham and X. Zhang, "An NHPP software reliability model and its comparison," *International Journal of Reliability, Quality and Safety Engineering*, vol. 4, no. 3, pp. 269–282, 1997.

[14] F. N. Parr, "An alternative to the Rayleigh curve for software development effort," *IEEE Transactions on Software Engineering*, vol. 6, no. 3, pp. 291–296, 1980.

[15] K. Pillai and V. S. Sukumaran Nair, "A model for software development effort and cost estimation," *IEEE Transactions on Software Engineering*, vol. 23, no. 8, pp. 485–497, 1997.

[16] J. Tian, P. Lu, and J. Palma, "Test-execution-based software reliability measurement and modeling for large commercial software," *IEEE Transactions on Software Engineering*, vol. 21, no. 5, pp. 405–414, 1995.

[17] S. Yamada, H. Ohtera, and H. Narihisa, "Software reliability growth models with testing effort," *IEEE Transactions on Reliability*, vol. 35, no. 1, pp. 19–23, 1986.

[18] S. Yamada, J. Hishitani, and S. Osaki, "Software-reliability growth with a Weibull test-effort. A model & application," *IEEE Transactions on Reliability*, vol. 42, no. 1, pp. 100–105, 1993.

[19] C. Y. Huang and S. Y. Kuo, "Analysis of incorporating logistic testing-effort function into software reliability modeling," *IEEE Transactions on Reliability*, vol. 51, no. 3, pp. 261–270, 2002.

[20] C. Y. Huang, "Performance analysis of software reliability growth models with testing-effort and change-point," *Journal of Systems and Software*, vol. 76, no. 2, pp. 181–194, 2005.

[21] S. Y. Kuo, C. Y. Huang, and M. R. Lyu, "Framework for modeling software reliability, using various testing-efforts and fault-detection rates," *IEEE Transactions on Reliability*, vol. 50, no. 3, pp. 310–320, 2001.

[22] M. H. Chen, M. R. Lyu, and W. Eric Wong, "Effect of code coverage on software reliability measurement," *IEEE Transactions on Reliability*, vol. 50, no. 2, pp. 165–170, 2001.

[23] M. Grottke, "A vector Markov model for structural coverage growth and the number of failure occurrences," in *Proceedings of the 13th International Symposium on Software Reliability Engineering*, pp. 304–315, IEEE CS Press, 2002.

[24] T. Ishii and T. Dohi, "Testing-effort dependent software reliability models based on two-dimensional NHPPs," *Transactions of IEICE*, vol. 89, no. 8, pp. 1684–1694, 2006.

[25] T. Ishii and T. Dohi, "Two-dimensional software reliability models and their application," in *Proceedings of the 12th Pacific Rim International Symposium on Dependable Computing (PRDC '06)*, pp. 3–10, IEEE CPS, December 2006.

[26] T. Ishii, T. Fujiwara, and T. Dohi, "Bivariate extension of software reliability modeling with number of test cases," *International Journal of Reliability, Quality and Safety Engineering*, vol. 15, no. 1, pp. 1–17, 2008.

[27] S. Inoue and S. Yamada, "A bivariate Weibull-type software reliability growth model and its goodness-of-fit evaluation," *Journal Information Processing*, vol. 49, no. 8, pp. 2851–2861, 2008 (Japanese).

[28] S. Inoue and S. Yamada, "Two-dimensional software reliability measurement technologies," in *Proceedings of the IEEE International Conference on Industrial Engineering and Engineering Management (IEEM '09)*, pp. 223–227, IEEE CPS, December 2009.

[29] J. Baik, D. N. P. Murthy, and N. Jack, "Two-dimensional failure modeling with minimal repair," *Naval Research Logistics*, vol. 51, no. 3, pp. 345–362, 2004.

[30] Y. H. Chun and K. Tang, "Cost analysis of two-attribute warranty policies based on the product usage rate," *IEEE Transactions on Engineering Management*, vol. 46, no. 2, pp. 201–209, 1999.

[31] D. N. P. Murthy, B. P. Iskandar, and R. J. Wilson, "Two dimensional failure free warranties: two-dimensional point process models," *Operations Research*, vol. 43, pp. 356–366, 1995.

[32] D. Roy, "Bivariate models from univariate life distributions: a characterization cum modeling approach," *Naval Research Logistics*, vol. 51, no. 5, pp. 741–754, 2004.

[33] S. C. Yang and J. A. Nachlas, "Bivariate reliability and availability modeling," *IEEE Transactions on Reliability*, vol. 50, no. 1, pp. 26–35, 2001.

[34] S. Yamada, M. Ohba, and S. Osaki, "S-shaped reliability growth modeling for software error detection," *IEEE Transactions on Reliability*, vol. 32, no. 5, pp. 475–484, 1983.

[35] A. L. Goel, "Software reliability models: assumptions, limitations and applicability," *IEEE Transactions on Software Engineering*, vol. 11, no. 12, pp. 1411–1423, 1985.

[36] K. Trivedi, *Probability and Statistics With Reliability, Queueing, and Computer Science Applications*, John Wiley & Sons, New York, NY, USA, 2nd edition, 2001.

[37] R. E. Barlow and F. Proschan, *Statistical Theory of Reliability and Life Testing Probability Models*, Holt, Rinehart and Winston, New York, NY, USA, 1975.

[38] A. Basu, "Bivariate failure rate," *Journal of the American Statistical Assosiation*, vol. 66, pp. 103–104, 1971.

[39] F. Downton, "Bivariate exponential distributions in reliability theory," *Journal of the Royal Statistical SocietySeries B*, vol. 32, pp. 408–417, 1970.

[40] E. J. Gumbel, "Bivariate exponential distributions," *Journal of the American Statistical Association*, vol. 55, pp. 698–707, 1960.

[41] N. L. Johnson and S. Kotz, *Distributions in Statistics: Continuous Multivariate Distributions*, John Wiley & Sons, Toronto, Canada, 1972.

[42] A. W. Marshall and I. Olkin, "A multivariate exponential distribution," *Journal of the American Statistical Association*, vol. 62, pp. 30–44, 1967.

[43] A. W. Marshall and I. Olkin, "A generalized bivariate exponential distribution," *Journal of Applied Probability*, vol. 4, pp. 291–302, 1967.

[44] J. C. Lu and G. K. Bhattacharyya, "Some new constructions of bivariate Weibull models," *Annals of the Institute of Statistical Mathematics*, vol. 42, no. 3, pp. 543–559, 1990.

[45] R. H. Hou, S. Y. Kuo, and Y. P. Chang, "Optimal release times for software systems with scheduled delivery time based on the HGDM," *IEEE Transactions on Computers*, vol. 46, no. 2, pp. 216–221, 1997.

[46] K. Okumoto and A. L. Goel, "Optimum release time for software systems based on reliability and cost criteria," *Journal of Systems and Software*, vol. 1, no. 4, pp. 315–318, 1980.

[47] H. Pham and X. Zhang, "A software cost model with warranty and risk costs," *IEEE Transactions on Computers*, vol. 48, no. 1, pp. 71–75, 1999.

[48] X. Teng and H. Pham, "A software cost model for quantifying the gain with considerations of random field environments," *IEEE Transactions on Computers*, vol. 53, no. 3, pp. 380–384, 2004.

[49] M. Xie and B. Yang, "A study of the effect of imperfect debugging on software development cost," *IEEE Transactions on Software Engineering*, vol. 29, no. 5, pp. 471–473, 2003.

[50] J. D. Musa, "Software reliability data," Technical Report, Data and Analysis Center for Software, Rome Air Development Center, New York, NY, USA, 1979.