Research Contribution

Reliability Assessment Measures Based on Software Reliability Growth Model with Normalized Method

JUN HISHITANI*, SHIGERU YAMADA* and SHUNJI OSAKI*

It is very important to assess software reliability quantitatively by using failure time data observed during software testing. This paper discusses a method of assessing software reliability on the basis of a software reliability model described by a nonhomogeneous Poisson process, that is to say, an exponential reliability growth model. The mean time between software failures and the software reliability function are adopted as reliability assessment measures. Since the time-intervals between software failures for the model are improper, it is shown that the means of the distributions can be obtained from the normalized distributions. Numerical examples of the mean time between software failures and the software reliability function are given by applying the method proposed here to actual data.

1. Introduction

It is very important to develop a highly reliable software system, since a computer system that has broken down as a result of software failure may cause various problems. In general, a software system is developed through successive four phases: specification, design, coding, and testing. Software reliability measurement and assessment in the testing phase, which is the last stage of the software development process, have been often discussed. The purpose of software testing is to detect and remove software faults latent in the software system. During the testing phase software failures are observed and recorded. A software failure is defined as an unacceptable departure from expected operation caused by a fault in the software system. We assume that all detected faults are removed and that no new faults are introduced into the program. On this assumption, the cumulative number of detected faults increases as they are corrected, and the time-interval between software failures becomes longer. This means that the probability of software failure-occurrence decreases, in other words that software reliability increases, as the testing goes on. A mathematical tool that deals with such a software failure-occurrence phenomenon during the testing phase is called a software reliability growth model [1]. Many software reliability growth models have been developed for describing the failure-occurrence (or fault-detection) phenomenon and assessing the software reliability (such as those described by Goel and Okumoto [2], Jelinski and Moranda [3], Littlewood [4], Moranda [5], Musa [6], Musa and Okumoto [7], and Yamada and Osaki [8]). As quantitative measures for the software reliability assessment, we can use the expected number of remaining faults in the system, the mean time-interval between software failures, the software reliability function, and so on, which are derived from the software reliability growth models.

In this paper, we analyze software failure time data observed during software testing by using a software reliability growth model based on a nonhomogeneous Poisson process (NHPP) [9]. In particular, using as a basis an exponential software reliability growth model proposed by Goel and Okumoto [2], we discuss methods of estimating the mean time-interval between software failures and the software reliability as quantitative assessment measures. Since the distribution of software failure-occurrence time in this model is improper, simplified and normalized methods of obtaining these measures for software reliability assessment are proposed. First, we summarize a software reliability growth model based on an NHPP, and discuss the software reliability analysis for the exponential software reliability growth model. Next, we investigate the distribution of the software failure-occurrence time in the exponential software reliability growth model, and propose simplified and normal methods of estimating the mean time-interval between software failures and the software reliability. We also derive the conditional mean time-interval between software failures from the normalized software reliability function. Finally, we compare the two methods proposed in this paper in terms of the goodness of fit by using actual software failure time data.

*Faculty of Engineering, Hiroshima University, Higashi-Hiroshima-shi, 724, Japan.

2. Software Reliability Growth Model

2.1 Model Description

During the testing phase of software development, many testing resources are used to detect and correct software faults. A software system is subject to software failures caused by the faults remaining in the system during the testing phase. Consequently, test data such as the software failure-occurrence times or the numbers of detected faults can be observed. These test data can be used to describe a software failure-occurrence phenomenon and to assess software reliability, on the basis of existing software reliability growth models. The following assumptions are made:

1. A software failure is caused by a software fault.
2. Each time a failure occurs the fault that caused it can be removed immediately.
3. Correction of a detected fault does not introduce any new faults.

Let $\{N(t), t \geq 0\}$ denote a counting process representing the cumulative number of faults detected up to testing time t. Then, by using a software reliability growth model, we can describe a software failure-occurrence phenomenon based on an NHPP as [10].

$$\Pr\{N(t)=n\}=\frac{\{H(t)\}^n}{n!}\exp[-H(t)] \quad (n=0, 1, 2, \cdots), \tag{1}$$

$$H(t)=\int_0^t h(s)ds, \tag{2}$$

where $H(t)$ is a mean value function that indicates the expected cumulative number of faults detected up to testing time t, and $h(t)$ is an intensity function that indicates the fault-detection rate at testing time t. Pr{A} in Eq. (1) means the probability of event A. Defining a as the expected initial fault content or the expected cumulative number of faults to be eventually detected, we usually assume that $H(\infty)=a$.

In general, the software reliability growth represents the mathematical relationship between the cumulative number of detected faults and the time span of software testing. During the testing phase, two types of reliability growth curves of the detected faults are typically observed: exponential and S-shaped software reliability growth curves. We therefore call the models describing these software failure-occurrence phenomena the exponential and S-shaped software reliability growth models, respectively.

In this paper, we discuss the exponential software reliability growth model with mean value function $m(t)$ proposed by Goel and Okumoto [2]:

$$H(t)\equiv m(t)=a(1-e^{-bt}) \quad (a>0, b>0), \tag{3}$$

where b is the fault-detection rate per fault remaining in the system.

2.2 Estimation of Parameters

Let X_k denote a random variable representing the time-interval between the $(k-1)$-st and k-th failures $(k=1, 2, \cdots)$. Then, $S_k \equiv \sum_{i=1}^{i=k} X_i$ is a random variable representing the k-th failure-occurrence time, where $X_k=S_k-S_{k-1}$ $(k=1, 2, \cdots, n;\ S_0=0)$. The joint probability density function of $\{S_1, S_2, \cdots, S_n\}$ for an NHPP with $H(t)$ in Eq. (2) is given by

$$f_{S_1,S_2,\cdots,S_n}(s_1, s_2, \cdots, s_n)=\exp[-H(s_n)]\prod_{i=1}^{n} h(s_i), \tag{4}$$

where $0 \leq s_1 \leq s_2 \leq \cdots \leq s_n < \infty$. Suppose that data on the failure-occurrence times s_k $(k=1, 2, \cdots, n;\ 0 \leq s_1 \leq s_2 \leq \cdots \leq s_n)$ (that is, s_k is a realization of S_k) are observed during the testing phase. The likelihood function for the unknown parameters in an NHPP model with $H(t)$ is then given by Eq. (4). Denoting the likelihood function in Eq. (4) by L and taking the natural logarithm of L yields

$$\ln L=-H(s_n)+\sum_{k=1}^{n}\ln h(s_k). \tag{5}$$

Solving the likelihood equations obtained from Eq. (5), we then can estimate the parameters of a fitted software reliability growth model based on an NHPP by the method of maximum likelihood.

For the exponential software reliability growth model, substituting Eq. (3) in Eq. (5) gives

$$\ln L=-a(1-e^{-bs_n})+n(\ln a+\ln b)-b\sum_{k=1}^{n}s_k. \tag{6}$$

We can obtain the maximum likelihood estimates of the unknown parameters a and b by solving the simultaneous likelihood equations $\partial \ln L/\partial a = \partial \ln L/\partial b=0$. Thus, we have

$$\left.\begin{aligned} n/a&=1-e^{-bs_n} \\ n/b&=\sum_{k=1}^{n}s_k+as_ne^{-bs_n} \end{aligned}\right\}, \tag{7}$$

which can be solved numerically.

3. Software Reliability Assessment Measures

We can estimate the mean value function $m(t)$ in Eq. (3) by using the estimated model parameters discussed above. It is very useful in quantitative assessment of assessing software reliability to estimate and predict software reliability measures derived from the estimated model. In this paper, we adopt the mean time between software failures and the software reliability as such measures.

3.1 Normalization of Failure Time Distribution

The mean time between software failures has been often adopted as a software reliability measure when the failure time data measured in CPU time or testeffort time are available.

From Eq. (4), the marginal density of $S_k, S_{k+1}, \cdots, S_n$ $(k=1, 2, \cdots, n)$ is given by

$$f_{S_k, S_{k+1}, \cdots, S_n}(s_k, s_{k+1}, \cdots, s_n) = \frac{\prod_{j=k}^{n} h(s_j) \cdot \{H(s_j)\}^{k-1}}{\Gamma(k)} \times \exp[-H(s_n)], \tag{8}$$

where $\Gamma(k)$ denotes a Gamma function. Further, deriving the marginal density of S_k from Eq. (8), the probability density function of S_k $(k=1, 2, \cdots, n)$ can be obtained as:

$$f_{S_k}(t) = \frac{h(t) \cdot \{H(t)\}^{k-1}}{\Gamma(k)} \exp[-H(t)] \quad (k=1, 2, \cdots, n). \tag{9}$$

From Eq. (9), the cumulative distribution function is given by

$$F_{S_k}(t) = \frac{1}{\Gamma(k)} \int_0^{H(t)} u^{k-1} e^{-u} du. \tag{10}$$

It should be noted that the cumulative distribution function is improper since

$$F_{S_k}(\infty) = 1 - \frac{1}{\Gamma(k)} \int_a^{\infty} u^{k-1} e^{-u} du < 1, \tag{11}$$

where $H(\infty)=a$, e.g., $m(\infty)=a$ in Eq. (3). Eq. (11) implies that a mean of the distribution of S_k does not exist. Therefore, the joint distribution function of $\{S_1, S_2, \cdots, S_n\}$ is also improper. Consequently, a mean time between software failures does not exist.

However, the mean time between failures can be obtained approximately by calculating the inverse transformation of the mean value function for the observed failure-occurrence time [2]. That is, for the exponential software reliability growth model, we can obtain the mean time to the k-th software failure, $\hat{s}_k$ by solving the following equation in terms of the observed failure time s_k:

$$k = m(s_k) \quad (k=1, 2, \cdots, n). \tag{12}$$

Then, we have

$$\hat{s}_k = m^{-1}(k) = \frac{-1}{b} \ln\left(1 - \frac{k}{a}\right) \quad (k=1, 2, \cdots, n). \tag{13}$$

Thus we can use this simplified method to estimate the mean time between software failures as $\hat{x}_k = \hat{s}_k - \hat{s}_{k-1}$ $(k=1, 2, \cdots, n)$.

In this paper, to obtain the mean time between software failures more accurately, we normalize the failure time distribution $F_{S_k}(t)$ in Eq. (10) as

$$G_{S_k}(t) \equiv \frac{F_{S_k}(t)}{F_{S_k}(\infty)} = \frac{F_{S_k}(t)}{\int_0^a u^{k-1} e^{-u} du / \Gamma(k)}. \tag{14}$$

In this way we can show that $G_{S_k}(0)=0$ and $G_{S_k}(\infty)=1$, and thus that $G_{S_k}(t)$ is proper. The probability density function is given by

$$g_{S_k}(t) \equiv \frac{f_{S_k}(t)}{F_{S_k}(\infty)} = \frac{f_{S_k}(t)}{\int_0^a u^{k-1} e^{-u} du / \Gamma(k)}. \tag{15}$$

3.2 Mean Time Between Software Failures

From Eq. (15), the mean of S_k, that is, the mean time to the k-th software failure, is given by

$$E[S_k] = \int_0^{\infty} t g_{S_k}(t) dt \quad (k=1, 2, \cdots, n) \tag{16}$$

Thus, the mean time between software failures can be obtained numerically from the equation

$$E[X_k] = E[S_k] - E[S_{k-1}] \quad (k=1, 2, \cdots, n). \tag{17}$$

From Eqs. (15) and (16), we have

$$E[S_k] = \frac{\int_0^a H^{-1}(u) u^{k-1} e^{-u} du}{\int_0^a u^{k-1} e^{-u} du}. \tag{18}$$

For the exponential software reliability growth model with $m(t)$ in Eq. (3), the mean time to the k-th software failure is calculated as

$$E[S_k] = \frac{1}{b} \frac{\int_0^a \{\ln a - \ln(a-u)\} u^{k-1} e^{-u} du}{\int_0^a u^{k-1} e^{-u} du} = \frac{1}{b} \left[\ln a - \frac{\int_0^a \ln(a-u) u^{k-1} e^{-u} du}{\int_0^a u^{k-1} e^{-u} du} \right], \tag{19}$$

since

$$H^{-1}(u) \equiv m^{-1}(u) = \frac{\ln a - \ln(a-u)}{b}. \tag{20}$$

The mean time between software failures $E[X_k]$ $(k=1, 2, \cdots, n)$ for this model can then be obtained from Eqs. (17) and (19).

3.3 Normalization of Software Reliability

For the software reliability growth model described by Eqs. (1) and (2), the software reliability, which represents the conditional probability that a software failure does not occur in the time interval $(t, t+x]$, given that the last failure time is t, is

$$R(x \mid t) \equiv \Pr\{X_k > x \mid S_{k-1} = t\} = \exp[-\{H(t+x) - H(t)\}] \quad (t \geq 0, x \geq 0), \tag{21}$$

which is independent of $k (k=1, 2, \cdots, n)$. From Eqs. (3) and (21), we have

$$R(x \mid t) = \exp[-e^{-bt} m(x)], \tag{22}$$

for the exponential software reliability growth model. Thus $R(0|t)=1$ and $R(\infty|t)=\exp[-ae^{-bt}]$. From Eq. (22), the conditional probability that a failure occurs in $(t, t+x]$ is given by

$$\begin{aligned} F(x|t)&=1-R(x|t) \\ &=1-\exp[-e^{-bt}m(x)]. \end{aligned} \tag{23}$$

Thus

$$F(0|t)=0, \quad F(\infty|t)=1-\exp[-ae^{-bt}]<1, \tag{24}$$

which implies that the conditional distribution $F(x|t)$ is also improper and that a mean does not exist.

In a similar way to that in which we found the failure time distribution $G_{S_k}(t)$ in Eq. (14), we can normalize the conditional distribution as

$$\begin{aligned} G(x|t)&\equiv\frac{F(x|t)}{F(\infty|t)} \\ &=\frac{1-\exp[-ae^{-bt}(1-e^{-bx})]}{1-\exp[-ae^{-bt}]}. \end{aligned} \tag{25}$$

From Eq. (25), $G(0|t)=0$ and $G(\infty|t)=1$. Therefore, the software reliability function is given by

$$\begin{aligned} S(x|t)&\equiv 1-G(x|t) \\ &=\frac{1-\exp[ae^{-b(t+x)}]}{1-\exp[ae^{-bt}]}. \end{aligned} \tag{26}$$

By the normalized method, we can obtain the mean of the distribution (25) as

$$\begin{aligned} E[X|t]&=\int_0^{\infty} S(x|t)dx \\ &=\frac{1}{b(e^{n(t)}-1)}\int_0^{n(t)}\frac{e^u-1}{u}du, \end{aligned} \tag{27}$$

where

$$n(t)=ae^{-bt}, \tag{28}$$

which is the expected number of remaining faults at testing time t for the exponential software reliability growth model. Eq. (27) shows the conditional mean time between software failures when then total testing time t is given.

4. Numerical Illustration

We analyze actual software failure time data observed during the testing phase to give numerical examples for application of the method described above. The data sets denoted by DATA1 and DATA2, which were cited by Goel and Okumoto [2] and Musa et al. [12], respectively, are analyzed here. DATA1 is available in the form s_k ($k=1, 2,\cdots, 26$) measured on the basis of the numbers of days. DATA2 is available in the form s_k ($k=1, 2,\cdots, 15$) measured on the basis of the numbers of CPU seconds.

We apply the exponential software reliability growth model with $m(t)$ in Eq. (3) to these data sets. The model parameters can be estimated by solving Eq. (7) numerically. We obtain the estimated parameters as $\hat{a}=33.99$ and $\hat{b}=0.00579$ for DATA1 and $\hat{a}=23.46$ and $\hat{b}=0.00345$ for DATA2, that is,

$$\text{DATA1:} \quad \hat{m}(t)=33.99(1-e^{-0.00579 \cdot t}), \tag{29}$$

$$\text{DATA2:} \quad \hat{m}(t)=23.46(1-e^{-0.00345 \cdot t}). \tag{30}$$

Using the estimation results above, we calculate the mean time between failures by using the simplified and normalized methods discussed in Section 3. Tables 1 and 2 show the estimated mean times between failures for DATA1 and DATA2, respectively, which are calculated by using the simplified method of Eq. (13) and the normalized method of Eq. (17). We find that the mean time between failures becomes longer (that is, software reliability grows) for each method as software testing goes on. Further, we calculate the sum of square errors between the actual time-interval and the estimated mean time-interval between failures to compare the simplified method in Section 3.1 and the normalized method in Section 3.2 in terms of goodness of fit. The sum of square error is given by

$$F=\sum_{k=1}^{n}(x_k-\hat{E}[X_k])^2, \tag{31}$$

where n is the total number of observed data, x_k calculated from s_k-s_{k-1} is the actual time-interval, and

Table 1 Estimated mean time between software failures for DATA1.

Failure No. k	Actual x_k	Simplified $\hat{x}_k$	Normalized $\hat{E}[X_k]$	Normalized $\hat{E}[X\|t=s_{k-1}]$
1	9	5.16	5.25	5.24
2	12	5.32	5.40	5.53
3	11	5.48	5.59	5.94
4	4	5.66	5.79	6.35
5	7	5.86	6.00	6.50
6	2	6.06	6.22	6.78
7	5	6.28	6.47	6.87
8	8	6.52	6.73	7.08
9	5	6.78	7.03	7.43
10	7	7.05	7.35	7.66
11	1	7.35	7.70	7.99
12	6	7.68	8.09	8.04
13	1	8.04	8.54	8.34
14	9	8.43	9.03	8.39
15	4	8.86	9.60	8.87
16	1	9.34	10.24	9.09
17	3	9.88	10.98	9.14
18	3	10.47	11.80	9.31
19	6	11.15	12.71	9.48
20	1	11.92	13.71	9.84
21	11	12.81	14.75	9.90
22	33	13.83	15.79	10.60
23	7	15.04	16.80	13.03
24	91	16.47	17.72	13.62
25	2	18.21	18.49	24.87
26	1	20.36	19.07	25.22
	F	7180	7062	8121

Table 2 Estimated mean time between software failures for DATA2.

Failure No. k	Actual x_k	Simplified $\hat{x}_k$	Normalized $\hat{E}[X_k]$	Normalized $\hat{E}[X\mid t=s_{k-1}]$
1	10	12.64	12.96	12.95
2	9	13.22	13.58	13.43
3	13	13.85	14.30	13.87
4	11	14.54	15.10	14.54
5	15	15.31	16.00	15.14
6	12	16.16	17.03	15.99
7	18	17.12	18.22	16.71
8	15	18.19	19.62	17.86
9	22	19.41	21.25	18.88
10	25	20.80	23.17	20.49
11	19	22.40	25.38	22.50
12	30	24.28	27.85	24.18
13	32	26.50	30.48	27.13
14	25	29.16	33.12	30.73
15	40	32.42	35.56	33.93
F		240	233	237

$\hat{E}[X_k]$ is the estimated mean time between the $(k-1)$-st and k-th failures. From Table 1 (Table 2), we have $F=7180$ ($F=240$) by the simplified method and $F=7062$ ($F=233$) by the normalized method for DATA1 (DATA2). This means that for these data sets, the normalized method fits better than the simplified method in calculating the mean time between failures.

We also obtain the estimated conditional mean times between failures, $E[X\mid t=s_{k-1}]$'s, from Eq. (27), as shown in Tables 1 and 2, where the total testing time t is equal to the actual $(k-1)$-st failure time s_{k-1} ($k=1, 2,\cdots, n$; $s_0=0$). Figures 1 and 2 show the relationship between the total testing time t and the estimated conditional mean time between failures $E[X\mid t=s_{k-1}]$ for DATA1 and DATA2, respectively.

Further, Figures 3 and 4 show the software reliability function $S(x\mid t)$ in Eq. (26) for DATA1 and DATA2, respectively, where the total testing times are $t=250$ (days) and $t=296$ (sec.) for DATA 1 and DATA2, respectively.

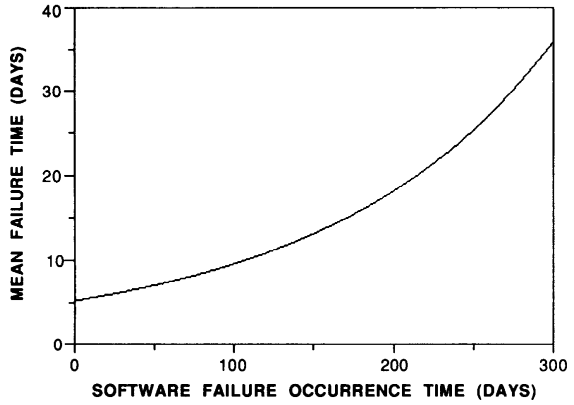

Fig. 1 Estimated conditional mean time between failures for DATA 1, based on the exponential software reliability growth model.

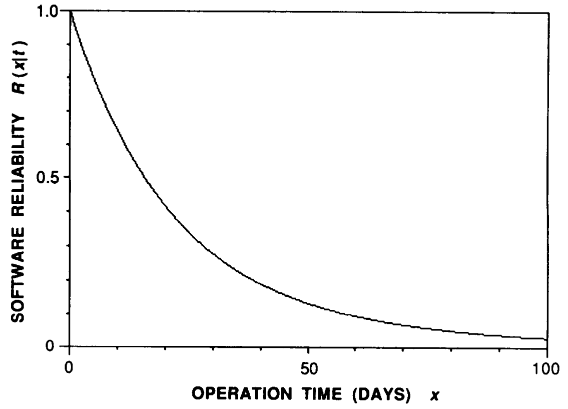

Fig. 3 Estimated software reliability function for DATA 1, using the normalized method.

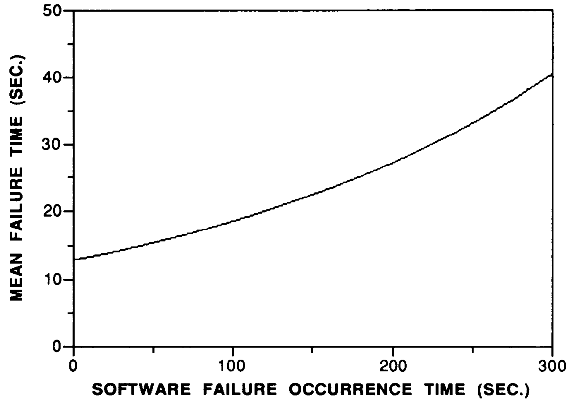

Fig. 2 Estimated conditional mean time between failures for DATA 2, based on the exponential software reliability growth model.

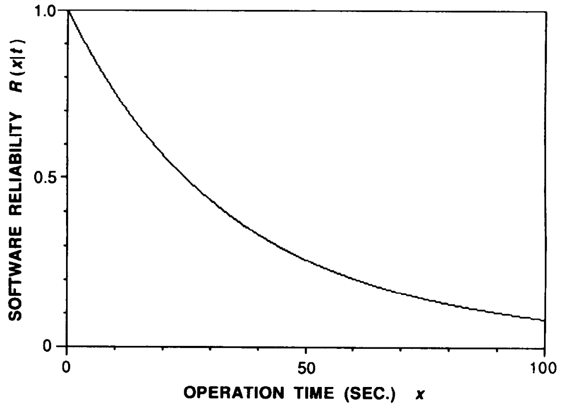

Fig. 4 The estimated software reliability function for DATA 2, using the normalized method.

5. Conclusion

Using as a basis a software reliability growth model described by an NHPP, we have discussed methods of assessing software reliability from failure time data. In particular, we have focused on the mean time between software failures and the software reliability as reliability assessment measures, and have proposed normalized methods of failure-occurrence time distributions. For analytical illustration, we have adopted an exponential software reliability growth model based on an NHPP. In general, the mean time to software failure cannot be accurately obtained, because the software failure times in the software reliability growth model have improper distributions. We have therefore proposed simplified and normalized methods for obtaining the mean time to software failure. We have also applied the normalized method to the software reliability function. Further, numerical examples of the mean time between software failures and the software reliability function have been given by analyzing actual software failure time data, and the simplified and normalized methods have been compared in terms of goodness of fit. As a result, we have confirmed that the estimation result given by the normalized method corresponds closely to the actual data.

In this paper, we have adopted the exponential software reliability growth model for analytical illustration. Of course, the normalized method can be used with other software reliability growth models based on NHPP's. For example, if we adopt a delayed S-shaped software reliability growth model [11] with mean value function

$$H(t) \equiv M(t) = a[1-(1+bt)e^{-bt}] \quad (a>0,\ b>0), \tag{32}$$

then we can obtain the mean of $G_{S_k}(t)$ in Eq. (14) as

$$E[S_k] \cong \frac{1}{b} \frac{\int_0^a \sqrt{2\ln\left(\frac{a}{a-u}\right)}\, u^{k-1} e^{-u} du}{\int_0^a u^{k-1} e^{-u} du}, \tag{33}$$

by using mathematical approximation. The mean time between software failures can then be obtained from Eq. (17).

In future, we will analyze more actual sets of data on software failure-occurrence time and study the applicability of the normalized method.

References

1. Ramamoorthy, C. V. and Bastani, F. B. Software reliability-Status and perspectives, *IEEE Trans. Softw. Eng.* SE-**8**, 4 (July 1982), 354-371.
2. Goel, A. L. and Okumoto, K. Time-dependent error-detection rate model for software reliability and other performance measures, *IEEE Trans. Reliab.* R-**28**, 3 (Aug. 1979), 206-211.
3. Jelinski, Z. and Moranda, P. B. Software reliability research, in *Statistical Computer Performance Evaluation*, ed. Freiberger, W., Academic Press, New York (1972), 465-484.
4. Littlewood, B. Theories of software reliability: How good are they and how can they be improved?, *IEEE Trans. Softw. Eng.* SE-**6**, 5 (Sept. 1980), 489-500.
5. Moranda, P. B. Event-altered rate models for general reliability analysis, *IEEE Trans. Reliab.* R-**28**, 5 (Dec. 1979), 376-381.
6. Musa, J. D. The measurement and management of software reliability, *Proc. IEEE*, **68**, 9 (Sept. 1980), 1131-1143.
7. Musa, J. D. and Okumoto, K. A logarithmic Poisson execution time model for software reliability measurement, *Proc. 7th Int. Conf. Software Engineering* (1984), 230-238.
8. Yamada, S. and Osaki, S. Software reliability growth modeling: Models and applications, *IEEE Trans. Softw. Eng.*, SE-**11**, 12 (Dec. 1985), 1431-1437.
9. Ascher, H. and Feingold, H. *Repairable Systems Reliability: Modeling, Inference, Misconceptions and Their Causes,* Marcel Dekker, New York (1984).
10. Yamada, S. *Software Reliability Assessment Technology* (in Japanese), HBJ Japan, Tokyo (1989).
11. Yamada, S., Ohba, M. and Osaki, S. S-shaped reliability growth modeling for software error detection, *IEEE Trans. Reliab.*, R-**32**, 5 (Dec. 1983), 475-478.
12. Musa, J. D., Iannino, A. and Okumoto, K. *Software Reliability: Measurement, Prediction, Application*, McGraw-Hill, New York (1987).

(Received December 18, 1989; revised May 8, 1990)