Vol. 34 No. 7 Transactions of Information Processing Society of Japan July 1993

Regular Paper

Software Reliability Measurement with Prior-Information on Initial Fault Content

MITSUHIRO KIMURA,† SHIGERU YAMADA,† HIROAKI TANAKA †† and SHUNJI OSAKI †

Most existing software reliability growth models, have been restricted in use with respect to the software reliability assessments during the later stages of integration or system testing. By introducing the prior probability distribution on the initial fault content in a software system, software reliability growth modeling as a binomial model is discussed here; allowing us to assess software reliability even during the earlier stage of testing. The model is described by a nonhomogeneous Markov process based on the assumption that the fault-detection rate is proportional to the number of remaining faults in the system. In particular, assuming the prior distribution to be a Poisson and a binomial distributions, we discuss the software reliability measurement. Several assessment measures for software reliability and the maximum-likelihood estimates for the required model parameters are derived. The application of this model is demonstrated by analyzing a set of actual fault-detection data observed during integration testing.

1. Introduction

Recently, since more breakdowns of a computer system are caused by software failures than by hardware ones, it is of great importance to produce reliable software systems by using engineering technologies. In general, an implemented software system is tested in the final phase of the software development to detect and correct software faults latent in the system. We can describe the software fault-detection or software failure-occurrence phenomenon by analyzing the fault or failure data observed during the testing phase. Then, the progress of reliability improvement can be evaluated. A software failure is defined as an unacceptable departure of the program operation caused by a software fault remaining in the system.

During the software testing phase, a problem on assessing software reliability arises. One of the most useful tools assessing software reliability quantitatively is a *software reliability growth model* which describes the fault-detection or failure-occurrence phenomenon during the testing and operational phases (see e.g. the references3)-5), 8), 9)). A software reliability growth is defined as a mathematical relationship between the time spent in operating a software system and the software reliability measures such as the cumulative number of detected faults and the time-interval between software failures (see e.g. Ramamoorthy and Bastani[6]), Yamada[9])). Using the software reliability growth model, we can estimate several software reliability measures such as the initial fault content, the mean time between failures, the expected number of remaining faults, the software reliability function, and so on.

For making effective decisions, it is of great importance to assess the software reliability during the earlier stage of the testing phase. Because of the data requirements to use the existing software reliability growth models, most are employed during the late stage of testing after integration testing. Then, the software reliability assessment during the earlier testing phase has been accomplished by using the subjective information of the project manager, the experienced results from past projects, and the simulated test data to estimate the needed model parameters. If we develop a software reliability growth model which can be employed during the early testing phase, then it will provide the project manager with useful information to control the resources for the testing to ensure highly

† Department of Industrial and Systems Engineering, Faculty of Engineering, Hiroshima University
†† Department of Applied Mathematics and Physics, Faculty of Engineering, Kyoto University

reliable software.

In this paper, we discuss the software reliability measurement during the early stage of software testing phase. Describing uncertain information on the initial fault content in the software system by the prior probability distribution, we propose a plausible software reliability growth model as a binomial model which was first developed by Shanthikumar[7]. It is assumed that the number of faults detected during the testing is represented by a nonhomogeneous Markov process and the fault-detection rate is proportional to the number of remaining faults in the system, which is a function of the testing time. In this paper, the term 'testing time' means the time spent for the dynamic software analyses.

Describing the fault-detection phenomenon in the software testing by a nonhomogeneous Markov process, a binomial reliability model for software system is briefly reviewed in section 2. Based on the binomial reliability model, software reliability measurement with prior-information on the initial fault content in the system is discussed in section 3. In particular, we discuss the case in which the initial fault content is assumed to obey a Poisson and a binomial distribution. Several assessment measures for software reliability are derived from the newly developed model in section 4. In the case in which the initial fault content obeys a binomial distribution, i.e. the size effect of software program on software reliability growth is considered, the model parameters are estimated by a method of maximum-likelihood in section 5. Section 6 applies the model to actual fault-detection data, and shows numerical examples of software reliability measurement.

2. Binomial Reliability Model

Let $M(t)$ be the number of remaining faults in the software system at the testing time t ($t \geq 0$). Considering a probabilistic nature associated with the fault-detection procedures in the software testing phase, we can regard $M(t)$ as a non-negative counting process. Here we assume that the software system contains m_0 faults at the initial testing time $t=0$.

In the software testing phase, since the introduced faults are detected and eliminated, then $M(t)$ gradually decreases as the test goes on. In this respect, it is natural to consider that the degree of its decrease is not constant throughout the testing phase and that it gradually varies being dependent on the number of remaining faults. Therefore, we have to construct a stochastic model reflecting such a software reliability characteristic. We assume the following properties with respect to the stochastic fluctuation of $M(t)$:

(a) $M(t)$ is a Markov process.

(b) A fault detected at the software testing phase is immediately eliminated and no new faults are introduced in the fault elimination procedure.

(c) The probability that one fault is detected and eliminated in the infinitesimal time interval $(t, t+\Delta t]$ is proportional to the number of remaining faults in the system at the time t, i.e.

$$\mathrm{P}[M(t+\Delta t)=m-1 \mid M(t)=m] = a(t)\, m\Delta t + o(\Delta t), \tag{1}$$

where $a(t)$ is the fault-detection rate per unit time and per fault remaining at the time t, and $o(\Delta t)$ means

$$\lim_{\Delta t \to 0} o(\Delta t)/\Delta t = 0.$$

(d) $M(t)$ has a rarity property, i.e.

$$\mathrm{P}[M(t)-M(t+\Delta t) \geq 2] = o(\Delta t). \tag{2}$$

These properties are schematically illustrated in **Fig. 1**.

Let us denote the transition probability of $M(t)$ as

$$P_M(m, t \mid m_0) \equiv \mathrm{P}[M(t)=m \mid M(0)=m_0] \quad (m=0, 1, 2, \cdots, m_0). \tag{3}$$

Shanthikumar[7] clarified that this probability

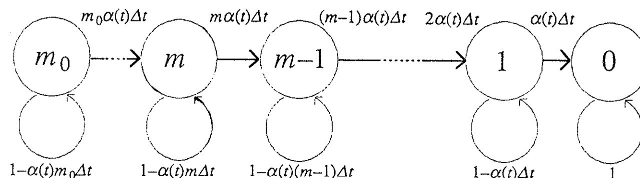


Fig. 1 Transition probabilities of the process $M(t)$.

distribution is the following binomial distribution under the assumptions (a)-(d):

$$P_M(m, t|m_0) = \binom{m_0}{m} e^{-mA(t)}(1-e^{-A(t)})^{m_0-m}$$

$$(m=0, 1, 2, \cdots, m_0), \quad (4)$$

$$A(t) = \int_0^t a(t')\,dt', \quad (5)$$

where $\binom{m_0}{m}$ is a binomial coefficient defined as

$$\binom{m_0}{m} = \frac{m_0!}{(m_0-m)!m!}.$$

On the other hand, let $N(t)$ be the total number of detected and eliminated faults up to the testing time t. Then, since the relationship $M(t)+N(t)=m_0$ holds between $M(t)$ and $N(t)$ with probability one, the transition probability of $N(t)$ is given as follows:

$$P_N(n, t|0) = \binom{m_0}{n}(1-e^{-A(t)})^n e^{-(m_0-n)A(t)}$$

$$(n=0, 1, 2, \cdots, m_0). \quad (6)$$

3. Modeling with Prior-Information on Initial Fault Content

3.1 Probability Distribution of Initial Fault Content

In the preceding section, it is assumed that m_0 is a constant whose value can be specified, which is the total number of the faults latent in the software system at the initial time $t=0$, i.e. the total number of introduced faults until the testing phase. However, in the actual situation, we can not specify the value of m_0 at the early stage of the testing phase, and we can not but estimate it based on the fault data obtained throughout the software testing phase. Moreover, it will be varied depending on many factors such as the kind of developed software systems, the progress of the development process, the applied development technologies and tools, and so on.

In this paper, we express such an uncertain information in terms of a probability distribution. Let us denote the probability distribution of the initial fault content $M(0)$ as

$$p_{m_0} = \mathrm{P}[M(0)=m_0], \quad (7)$$

and the probability distribution of the process $M(t)$ considering this prior distribution as

$$P_M(m, t) = \mathrm{P}[M(t)=m]. \quad (8)$$

Then, with the Bayes' rule, we obtain

$$P_M(m, t) = \sum_{m_0(\geq m)} P_M(m, t|m_0)\,p_{m_0}$$

$$(m=0, 1, 2, \cdots). \quad (9)$$

Substituting (4) into (9), we can obtain the following expression for the probability mass function of $M(t)$:

$$P_M(m, t) = (1-e^{-A(t)})^{-m} e^{-mA(t)} \times \sum_{m_0(\geq m)} \binom{m_0}{m}(1-e^{-A(t)})^{m_0} p_{m_0}. \quad (10)$$

Similarly, the probability mass function of the process $N(t)$ is given by:

$$P_N(n, t) = (1-e^{-A(t)})^n e^{nA(t)} \sum_{m_0(\geq n)} \binom{m_0}{n} \times e^{-m_0 A(t)} p_{m_0}. \quad (11)$$

From (10) and (11), if we can use the information of the prior probability distribution on the initial fault content from software engineer's experience, then we can perform software reliability analysis in more detail during the earlier stage of the testing.

3.2 The Case of Poisson Distribution

As the most common case, we consider the case in which the initial fault content $M(0)$ obeys a Poisson distribution, i.e. p_{m_0} is given as

$$p_{m_0} = \frac{\mu^{m_0}}{m_0!} e^{-\mu} \quad (\mu = \mathrm{E}[M(0)];$$

$$m_0=0, 1, 2, \cdots), \quad (12)$$

where μ represents the expected value of $M(0)$.

Substituting (12) into (10) and (11), we can obtain

$$P_M(m, t) = \frac{\{\mu e^{-A(t)}\}^m}{m!} \exp\{-\mu e^{-A(t)}\}, \quad (13)$$

$$P_N(n, t) = \frac{\{\mu(1-e^{-A(t)})\}^n}{n!} \times \exp\{-\mu(1-e^{-A(t)})\}. \quad (14)$$

Equations (13) and (14) mean that both of $M(t)$ and $N(t)$ are the well-known non-homogeneous Poisson processes (NHPP's). Especially, if the fault-detection rate $a(t)$ is a constant and independent of the testing time t, it is clear that the distribution (13) coincides with the exponential reliability growth model proposed by Goel and Okumoto[2].

3.3 The Case of Binomial Distribution

Next, we will consider the case in which $M(0)$ obeys a binomial distribution, i.e.

$$p_{m_0} = \binom{K}{m_0} \lambda^{m_0}(1-\lambda)^{K-m_0}$$

$$(0<\lambda<1;\ m_0=0, 1, 2, \cdots, K). \quad (15)$$

This distribution is obtained under the follow-

ing assumptions:

(a) The program size at the end of the coding phase is K, i.e. the software system consists of K lines of codes (L.O.C.) at the beginning of the testing phase.

(b) Each code contains one fault with a constant probability λ.

(c) Each failure which is caused by a fault occurs independently and randomly in time.

Therefore, under these assumptions, we can incorporate the effect of program size in software reliability analysis.

Substituing (15) into (10) and (11), we can obtain the following probability distribution.

$$P_M(m, t) = \binom{K}{m}(\lambda e^{-A(t)})^m (1-\lambda e^{-A(t)})^{K-m} \quad (m=0, 1, 2, \cdots, K), \tag{16}$$

$$P_N(n, t) = \binom{K}{n}\{\lambda(1-e^{-A(t)})\}^n \{1-\lambda(1-e^{-A(t)})\}^{K-n} \quad (n=0, 1, 2, \cdots, K). \tag{17}$$

Equations (16) and (17) mean that both of $M(t)$ and $N(t)$ obey a binomial distribution.

4. Software Reliability Measures

In this section, based on the probability distribution of the number of remaining faults $M(t)$ obtained in the preceding section, we derive some useful quantitative measures for the software reliability assessment. It is clear that the software reliability measures obtained in this section can be used at the earlier stage of testing if we have the information of the prior probability distribution on the initial fault content.

4.1 Software Reliability

Let $R(\tau|t)$ be the probability that no software failure occurs in the time interval $(t, t+\tau]$ $(\tau \geq 0)$ under the condition that all software faults detected up to the testing time t are eliminated in the testing phase. We call this conditional survival function the software reliability in the testing phase (see Goel and Okumoto[2] and Yamada and Osaki[10]). By the use of the probability distribution of $M(t)$, the software reliability $R(\tau|t)$ can be expressed as follows:

$$R(\tau|t) = \sum_{k=0}^{\infty} \mathrm{P}[M(t+\tau)=k | M(t)=k] \times P[M(t)=k]. \tag{18}$$

Substituting (10) into (18), we can obtain

$$R(\tau|t) = \sum_{k=0}^{\infty}\left[e^{-kA(t+\tau)}(1-e^{-A(t)})^{-k} \times \sum_{m_0(\geq k)} \binom{m_0}{k}(1-e^{-A(t)})^{m_0} p_{m_0}\right]. \tag{19}$$

If $M(0)$ obeys a Poisson distribution, substituting (12) into (19), then we can obtain the following expression for the software reliability:

$$R(\tau|t) = \exp\{\mu(e^{-A(t+\tau)} - e^{-A(t)})\}. \tag{20}$$

On the other hand, if $M(0)$ obeys a binomial distribution, substituting (15) into (19), then we can obtain

$$R(\tau|t) = \{1+\lambda(e^{-A(t+\tau)} - e^{-A(t)})\}^K. \tag{21}$$

4.2 Expected Number of Remaining Faults and Its Variance

As well as the software reliability in the testing phase, an information on the current number of faults remaining in the system is important to estimate the progress situation of the software testing phase. Since it is a random variable in our model, its expected value and variance are useful measures. These can be easily calculated from (10) as follows:

$$\mathrm{E}[M(t)] = \mathrm{E}[M(0)]e^{-A(t)}, \tag{22.a}$$

$$\mathrm{Var}[M(t)] = \mathrm{Var}[M(0)]e^{-2A(t)} + \mathrm{E}[M(0)]e^{-A(t)}(1-e^{-A(t)}). \tag{22.b}$$

If $M(0)$ obeys a Poisson distribution, then they are expressed as

$$\mathrm{E}[M(t)] = \mu e^{-A(t)}, \tag{23.a}$$

$$\mathrm{Var}[M(t)] = \mu e^{-A(t)}. \tag{23.b}$$

Similarly, if $M(0)$ obeys a binomial distribution, then we have

$$\mathrm{E}[M(t)] = K\lambda e^{-A(t)}, \tag{24.a}$$

$$\mathrm{Var}[M(t)] = K\lambda e^{-A(t)}(1-\lambda e^{-A(t)}). \tag{24.b}$$

Therefore, from (23.a) and (24.a), the expected initial fault content, $\mathrm{E}[M(0)]$'s, are given by μ and $K\lambda$ when $M(0)$ obeys a Poisson and a binomial distribution, respectively.

5. Maximum-Likelihood Estimations

In this section, based on a method of maximum-likelihood, we discuss the estimation method of the model parameters in the binomial reliability model discussed in section 3. In particular, we deal with the case of a binomial distribution for the initial fault content in order to clarify the effect of the program size on software reliability growth. For the case of a Poisson distribution, we can easily estimate the model parameters based on an NHPP (see Yamada[9,10]).

First, we assume that the fault-detection rate $\alpha(t)$ is proportional to the testing-effort function $w(t)$ (see Yamada et al.[11]), i.e.

$$\alpha(t)=aw(t), \tag{25}$$

where $w(t)$ is the current testing-effort expenditures at the testing time t. In this paper, the 'testing-effort expenditures' is defined as testing resources measured such as CPU time, manpower, executed test cases and so on. If the testing-effort expenditures consumed for the fault-detection is constant throughout the testing, then we can assume that $w(t)=1$ without loss of generality. Otherwise, $w(t)$ is determined from the testing-effort data. For example, Yamada et al.[11] proposed testing-effort functions expressed by exponential and Rayleigh curves. In this paper, we assume that the testing-effort function $w(t)$ has been already determined.

Under the assumptions above, the model parameters to be specified are a in (25), K and λ in (16) or (17). We use a method of maximum-likelihood to estimate these parameters based on fault-detection data observed in the testing.

Let y_k $(k=1, 2, \cdots, n)$ be the cumulative number of detected software faults up to the testing time t_k $(k=1, 2, \cdots, n;\ 0<t_1<t_2<\cdots<t_n)$. The likelihood function l for the process $N(t)$ is given by the following joint probability mass function:

$$l=\Pr[N(t_1)=y_1, N(t_2)=y_2, \cdots, N(t_n)=y_n]. \tag{26}$$

Using the Bayes' rule, we have

$$l=\Pr[N(t_2)=y_2, \cdots, N(t_n)=y_n | N(t_1)=y_1]\Pr[N(t_1)=y_1]. \tag{27}$$

Further, iterating the same procedure and using the Markov property, we can obtain the following expression:

$$l=\left\{\prod_{k=2}^{n}\Pr[N(t_k)=y_k|N(t_{k-1})=y_{k-1}]\right\}\times\Pr[N(t_1)=y_1]. \tag{28}$$

The conditional probability $\Pr[N(t_k)=y_k|N(t_{k-1})=y_{k-1}]$ can be easily evaluated by considering that the initial time is moved to $t=t_{k-1}$, and that the distribution range of $N(t)$ after that time is confined to the range $0\leq N(t)\leq K-y_{k-1}$, hence,

$$\Pr[N(t_k)=y_k|N(t_{k-1})=y_{k-1}]=\binom{K-y_{k-1}}{y_k-y_{k-1}}\times\{x(t_k, t_{k-1})\}^{y_k-y_{k-1}}\times\{1-x(t_k, t_{k-1})\}^{K-y_k}, \tag{29}$$

where,

$$x(t_k-t_{k-1})=\frac{\lambda(e^{-A_{k-1}}-e^{-A_k})}{1-\lambda(1-e^{-A_{k-1}})},\quad A_k=A(t_k). \tag{30}$$

Substituting (11) and (29) into (28), we finally obtain the likelihood function l as follows:

$$l=\prod_{k=1}^{n}\left[\binom{K-y_{k-1}}{y_k-y_{k-1}}\{x(t_k, t_{k-1})\}^{y_k-y_{k-1}}\times\{1-x(t_k, t_{k-1})\}^{K-y_k}\right], \tag{31}$$

where $t_0=0$ and $y_0=0$.

Substituting (25) into (31) and taking the natural logarithm of (31) yield

$$L\equiv\log l=\log K!-\log\{(K-y_n)!\}-\sum_{k=1}^{n}\log\{(y_k-y_{k-1})!\}+y_n\log\lambda+\sum_{k=1}^{n}(y_k-y_{k-1})\times\log(e^{-aW(t_{k-1})}-e^{-aW(t_k)})+(K-y_n)\log\{1-\lambda\times(1-e^{-aW(t_n)})\}, \tag{32}$$

where $W(t)$ is the cumulative testing-effort function defined as

$$W(t)=\int_0^t w(t')\,dt'. \tag{33}$$

As mentioned in section 3, we can consider, as a common case, that the parameter K corresponds to the program size. If the program size is known, then we have only to determine λ and a. From (32), the maximum likelihood estimates λ^* and a^* can be obtained as the solutions of the following simultaneous likelihood equations:

$$\frac{\partial L}{\partial\lambda}=\frac{y_n}{\lambda}-(K-y_n)\cdot\frac{(1-e^{-aW(t_n)})}{[1-\lambda(1-e^{-aW(t_n)})]}=0, \tag{34}$$

$$\frac{\partial L}{\partial a}=\sum_{k=1}^{n}(y_k-y_{k-1})\cdot\frac{[W(t_k)e^{-aW(t_k)}-W(t_{k-1})e^{-aW(t_{k-1})}]}{[e^{-aW(t_{k-1})}-e^{-aW(t_k)}]}-(K-y_n)\cdot\frac{\lambda W(t_n)e^{-aW(t_n)}}{[1-\lambda(1-e^{-aW(t_n)})]}=0. \tag{35}$$

Solving (34) with respect to λ, we obtain

$$\lambda=\frac{y_n}{K(1-e^{-aW(t_n)})}. \tag{36}$$

Substituting (36) into (35),

$$\sum_{k=1}^{n}(y_k-y_{k-1})$$

$$\cdot \frac{[W(t_k)e^{-aW(t_k)} - W(t_{k-1})e^{-aW(t_{k-1})}]}{[e^{-aW(t_{k-1})} - e^{-aW(t_k)}]}$$
$$= y_n \cdot W(t_n) \frac{e^{-aW(t_n)}}{[1-e^{-aW(t_n)}]}. \quad (37)$$

The maxmum-likelihood estimates a^* and λ^* of unknown parameters a and λ can be obtained as the solution of (36) and (37).

It is noted that the solutions satisfy the following conditions:

$$0<\lambda^*<1, \quad a^*>0. \quad (38)$$

In particular, if we have the empirical information about λ, i.e. the fault rate, then we have only to solve (35) with respect to a.

6. Numerical Examples

In this section, we analyze actual software fault-detection data to show numerical examples for application of the model proposed in this paper.

We use the fault-detection data set denoted by DS-1 which was analyzed by Yamada et al.[11] This data set was first cited by Brooks and Motley[1]. The data set DS-1 takes the form of (τ_k, w_k, y_k) $(k=1, 2, \cdots, 35)$, where τ_k, w_k, and y_k represent the calendar time (months), the wall clock hours consumed at calendar time τ_k (i.e. a realization of $w(\tau_k)$), and the cumulative number of faults detected up to the calendar time τ_k (i.e. a realization of $N(\tau_k)$), respectively. The program size K of this software system written by ALC and CENTRAN is 1.24×10^5 lines of code.

Suppose that we begin to measure the software reliability at the middle of testing phase, t_{17}. We have the empirical information that the fault rate (the number of faults per one code prior to the testing) is 1.30%, that is $\lambda=1.30\times10^{-2}$. Therefore, by applying the well-known Newton-Raphson method to solve (37) for the observed data (τ_k, w_k, y_k) $(k=1, 2, \cdots, 17)$, the maximum likelihood estimate a^* of unknown parameter a is obtained as follows:

$$a^*=9.791\times10^{-4}. \quad (39)$$

In the above calculation, we use the cumulative wall clock hours $W_k=\sum_{i=1}^{k} w_i$ as a realization of $W(\tau_k)$. Hence the testing-effort function $w(t)$ in (25) is assumed to be equal to unity.

Accordingly, we estimate the expected cumulative number of faults detected up to testing time t as

$$\hat{E}[N(t)]=K\lambda-\hat{E}[M(t)] = (124000)(0.0130)(1-e^{-0.0009791t}). \quad (40)$$

Moreover, to compare the performance of software reliability prediction, we also apply an exponential software reliability growth model based on an NHPP to the same data. As a result, we can estimate the expected cumulative number of detected faults (i.e. the mean value function of the exponential software reliability growth model) as follows:

$$\hat{E}[N(t)]=2960(1-e^{-0.0004694t}). \quad (41)$$

Figure 2 shows $\hat{E}[N(t)]$'s in (40) and (41) along with the actual software fault data DS-1.

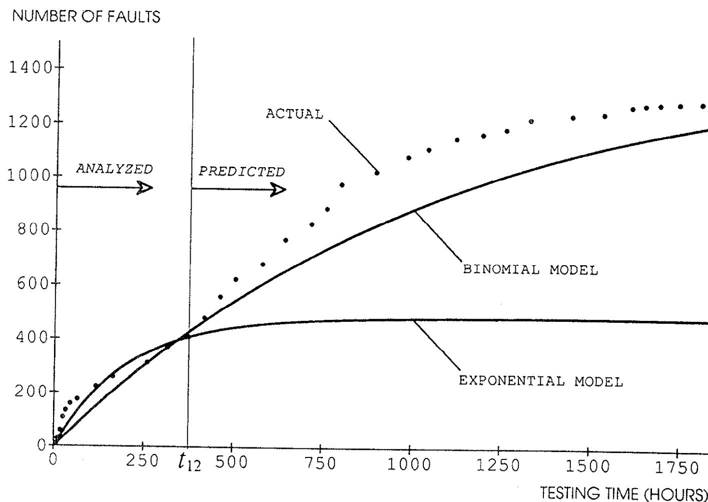


Fig. 2 The estimated expected number of detected faults, $\hat{E}[N(t)]$.

Both the models fit to the actual data points well by testing time $t_{17}=642.9$. However, we have found that the exponential software reliability growth model does not fit in the later stage of the testing phase. So, if the software development manager used such a model for software reliability prediction, he or she may make a wrong decision in terms of software management in the later stage of testing. This also means that the trustworthy information prior to testing can contribute to forecasting of software reliability.

By using of the estimated model parameter, the expected number of remaining faults, $\hat{E}[M(t)]$ is shown in **Fig. 3**. Further, **Figs. 4 and 5** show estimated software reliability functions $\hat{R}(\tau|t)$'s versus τ and t, respectively, where t in Fig. 4 is evaluated at the forecasting time mentioned above, i.e. $t_{17}=642.9$.

7. Concluding Remarks

In this paper, we have modified the binomial reliability model first proposed by Shanthikumar, and have derived the probability distribution of the number of detected software faults

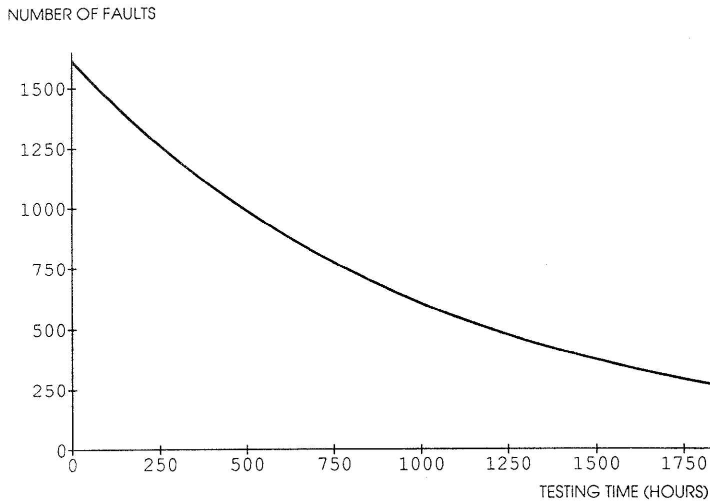


Fig. 3 The estimated expected number of remaining faults, $\hat{E}[M(t)]$.

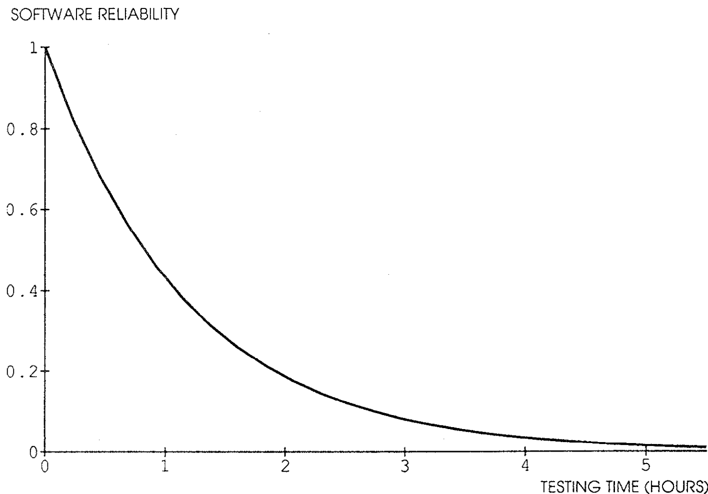


Fig. 4 Dependency of τ on the estimated software reliability, $\hat{R}(\tau|t)$ ($t=642.9$[hours]).

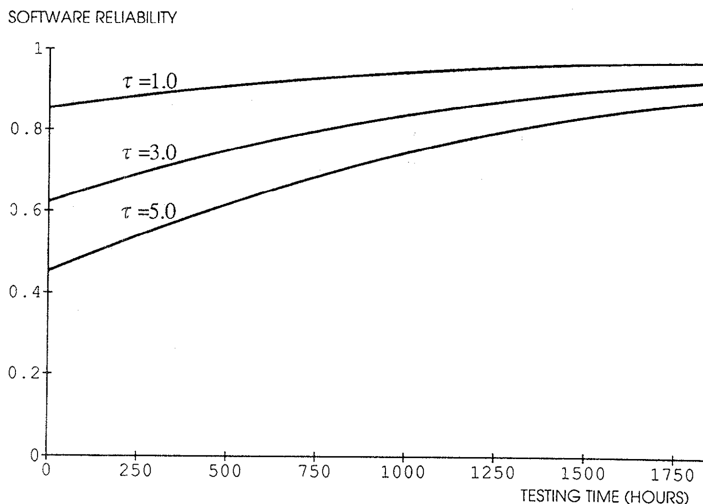


Fig. 5 Dependency of t on the estimated software reliability, $\hat{R}(\tau|t)$.

and some useful reliability measures by expressing the uncertain information on the initial fault content in terms of the probability distribution.

In particular, we have shown that both the processes $M(t)$ and $N(t)$ obey a binomial distribution if the initial fault content obeys a binomial distribution. This model is very useful to the point that we can easily take the objective program size into the reliability analysis. Further, according to the maximum-likelihood estimations discussed in section 5, we can estimate the program size K from the fault-detection data in the testing phase even if it is unknown. In addition if we can use the information of the prior probability distribution on the initial fault content from a software engineer's experience, we can perform software reliability analysis in more detail during the earlier stage of testing. These are other advantageous points of our binomial model.

In future, we are planning to study the testing-effort function to reflect an actual environment of software testing, and to study the problem of what type of distribution is best for the probability distribution of the initial fault content.

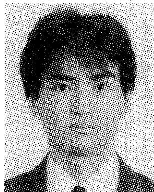
Acknowledgment Shigeru Yamada is pleased to acknowledge the support of a Grant-in-Aid for Scientific Research from the Ministry of Education, Science and Culture of Japan under Grant No. 04650316.

References

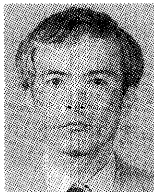
1) Brooks, W. D. and Motley, R. W.: Analysis of Discrete Software Reliability Models, Technical Report RADC-TR-80-84, Rome Air Development Center, New York (1980).
2) Goel, A. L. and Okumoto, K.: Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures, *IEEE Trans. Reliability*, Vol. R-28, No. 3, pp. 206-211 (1979).
3) Littlewood, B. and Miller, D. (eds.): *Software Reliability and Safety*, Elsevier Applied Science, London (1991).
4) Malaiya, Y. K. and Srimani, P. K. (eds.): *Software Reliability Models: Theoretical Developments, Evaluation and Application*, IEEE Computer Society Press, Los Alamitos (1990).
5) Musa, J. D., Iannino, A. and Okumoto, K.: *Software Reliability: Measurement, Prediction, Application*, McGraw-Hill, New York (1987).
6) Ramamoorthy, C. V. and Bastani, F. B.: Software Reliability-Status and Perspectives, *IEEE Trans. Softw. Eng.*, Vol. SE-8, No. 4, pp. 354-371 (1982).
7) Shanthikumar, J. G.: A General Software Reliability Model for Performance Prediction, *Microelectronics and Reliability*, Vol. 21, No. 5, pp. 671-682 (1981).
8) Special Issue on Software Reliability-PART I, *IEEE Trans. Softw. Eng.*, Vol. SE-11, No. 12 (1985).

9) Yamada, S.: *Software Reliability Assessment Technology: Introduction to Software Reliability Growth Model*, (in Japanese) HBJ Japan, Tokyo (1989).
10) Yamada, S.: Software Quality/Reliability Measurement and Assessment: Software Reliability Growth Models and Data Analysis, *J. Info. Process.*, Vol. 14, No. 3, pp. 254-266 (1991).
11) Yamada, S., Ohtera, H. and Narihisa, H.: Software Reliability Growth Models with Testing-Effort, *IEEE Trans. Reliability*, Vol. R-35, No. 1, pp. 19-23 (1986).

(Received August 31, 1992)
(Accepted April 14, 1993)



Mitsuhiro Kimura was born in Hiroshima Prefecture, Japan, on October 1, 1964. He received the BSE and MS degrees from Hiroshima University in 1989 and 1991, respectively. From 1991, he has been a graduate student in Graduate School of System Engineering, Hiroshima University, Higashi-Hiroshima-shi, Japan. His research area includes reliability theory, and software reliability models. He is a regular member of the Institute of Electronics, Information and Communication Engineers of Japan, the Operations Research Society of Japan, and the Information Processing Society of Japan.



Shigeru Yamada was born in Hiroshima Prefecture, Japan, on July 6, 1952. He received the BSE, MS, and Ph.D. degrees from Hiroshima University in 1975, 1977, and 1985, respectively. From 1977 to 1980 he worked at the Quality Assurance Department of Nippondenso Co., Aichi Prefecture, Japan. From 1983 to 1988 he was an Assistant Professor of Okayama University of Science, Okayama, Japan. Since 1988 he has been working as an Associate Professor of Hiroshima University, Higashi-Hiroshima-shi, Japan. He has published numerous technical papers in the area of software reliability models, reliability engineering, and quality control. Recently he has authored a book entitled Software Assessment Technology (in Japanese, HBJ Japan, Tokyo, 1989) and Software Reliability: Theory and Practical Application (in Japanese, Soft Research Center, Tokyo, 1990). Dr. Yamada received the Best Author Award from the Information Processing Society of Japan in 1992. He is a regular member of the Institute of Electronics, Information and Communication Engineers of Japan, the Information Processing Society of Japan, the Operations Research Society of Japan, IEEE, the IEEE Computer and Reliability Societies.

Hiroaki Tanaka was born in Osaka Prefecture, Japan, on September 27, 1961. He received the BSE, MS, and Ph. D. degrees from Kyoto University in 1984, 1986, and 1989, respectively. From 1989 to 1992 he was an Assistant Professor of Hiroshima University, Higashi-Hiroshima-shi, Japan. Since 1992 he has been working as an Assistant Professor of Kyoto University, Sakyo-ku, Kyoto, Japan. His research area includes stochastic model of fatigue crack propagation, structural reliability engineering, stochastic differential equation and its application, and software reliability models. He is a regular member of the Japan Society of Materials Science and the Japan Society of Mechanical Engineering.

Shunji Osaki was born in Aichi Prefecture, Japan, on January 3, 1942. He received the BSE degree from Nagoya Institute of Technology, in 1964, and the MS and Ph.D. degrees both from Kyoto University, Kyoto in 1967 and 1970, respectively. Since 1986 he has been working as a Professor of Hiroshima University, Higashi-Hiroshima-shi, Japan. His research area includes reliability theory, operations research, systems engineering, and applied mathematics. He acted a Guest Editor on Special Section: "Reliability Research Effort in Japan," IEEE Trans. Reliab., Vol. R-32, No. 4, Oct., 1984. Professor Osaki is a member of the Operations Research Society of Japan, the Institute of Electronics, Information and Communication Engineers of Japan, and the Japan Association of Automatic Control Engineers. He is an Associate editor of the Journal of Mathematical Analysis and Applications and the International Journal of Policy and Information.