Invited Paper

Software Quality/Reliability Measurement and Assessment: Software Reliability Growth Models and Data Analysis

SHIGERU YAMADA*

Software reliability assessment is very important in developing a quality software product efficiently. This paper discusses the quantitative measurement and assessment of software reliability. The techniques are based on the software reliability growth models (SRGM's) developed in Japan, which are characterized by nonhomogeneous Poisson processes. By making the assumptions on which they are based more realistic, the models discussed here were designed to describe a software error-detection process or a software failure-occurrence process during the testing phase of software development. A summary of existing SRGM's is given, and the maximum-likelihood estimations based on the SRGM's are discussed for software reliability data analysis and software reliability assessment. Through the use of a software reliability assessment tool that incorporates several leading SRGM's, examples of software reliability assessment are given for some sets of observed test data taken from actual software projects.

1. Introduction

Recently, computer systems have come to be indispensable tools in various activities of society. Among the elements of such systems, the software is becoming particularly complex and large-scale, which has led to problems in software development expressed in the phrase "software crisis." At the same time, the quality assurance of software has become one of the most important problems in the development of software production technologies.

The software development process consists of four successive phases, namely specification, design, coding, and testing. Many software errors (or faults) introduced in the first three phases are detected and corrected, mostly during the testing phase. In this sense, the testing phase is very important in verifying and ensuring the software quality for users. In fact, a considerable development effort is needed at this stage. The following are known as the standard characteristics of software quality: functionality, reliability, usability, efficiency, maintainability, and portability. Software reliability is a particularly important quality characteristic as a taken-for-granted quality [1]. It is defined as the probability of no occurrence of a software failure during a certain period on a specified condition[2]. A software failure is defined as an unacceptable departure from normal program operation caused by a software error latent in the software. If the total number of errors latent in the software can be estimated with a high accuracy in the testing phase, the software reliability can be measured and assessed quantitatively. It will then be possible to control the progress of the testing and to predict the time at which the software can be released for operational use.

For this reason, there have been many studies on software reliability models for analyzing the observed data during the testing phase and assessing the reliability of the developed software. One of the most useful models is well known as a software reliability growth model [3-10]. This model represents the relationship between the time span of software testing and the number of detected errors as a process of growth in software reliability, describing the error-detection phenomenon or the failure-occurrence phenomenon during the testing phase. On the basis of the software reliability growth model, it is possible to estimate and predict the expected initial error content, the expected number of remaining errors at an arbitrary time during testing, the mean time between software failures, the software reliability, and so on.

This paper aims at quantitative measurement and assessment of the software reliability, and discusses several leading software reliability growth models developed in Japan to assist in the task. These models are based on nonhomogeneous Poisson processes that describe the time-dependent behavior of software errors detected or software failures occurring during the testing phase. First, we give general descriptions of a software reliability growth model based on a nonhomogeneous Poisson process and quantitative

*Department of Industrial and Systems Engineering, Faculty of Engineering, Hiroshima University, Higashi-Hiroshima-shi, 724, Japan.

measures for software reliability assessment. Then, we summarize the existing software reliability growth models developed in Japan. Next, classifying the observed test data available for software reliability assessment into two types, we present the maximum-likelihood estimations based on the SRGM's for software reliability data analysis. We also discuss the procedure of data analysis for software reliability assessment, and related tools. Finally, using a software reliability assessment tool that incorporates several leading software reliability growth models mentioned above, we give examples of software reliability assessment based on two data sets taken from actual software projects.

2. Software Reliability Growth Model (SRGM)

2.1 Non-Homogeneous Poisson Process (NHPP) Model

During the testing phase of software development, many testing resources are required in order to detect and correct software errors. During the testing phase, computer software is subject to software failures caused by errors latent in the software. Test data such as the times of software failures or the numbers of detected errors can then be observed. If it is assumed that the correction of errors does not introduce any new errors, the cumulative number of detected errors increases as they are corrected, and the mean time interval between software errors becomes longer (Fig. 1). This means that the probability of no failure occurring in a fixed time-interval, that is, the reliability, increases with the progress of software testing. A mathematical tool that describes such an error-detection or failure-occurrence phenomenon is called a *software reliability growth model* (*SRGM*). Many SRGM's have been developed for measuring and assessing software reliability (e.g. [11-20]). The following assumptions are usually made in the area of software reliability growth modeling:

1) A software is subject to failures at random times caused by errors latent in the software.
2) A software failure is caused by a software error.
3) Each time a failure occurs, the error that caused it is immediately removed, and no new errors are introduced.

The testing time (for example, the calendar time, the testing effort (man-power), the CPU time (machine execution time), or the number of executed test-cases) is generally used as the unit of error-detection or failure-occurrence period for describing the time-dependent behavior of the cumulative number of errors detected or failures occurring during the testing phase. The following random variables can be defined for software reliability measurement (Fig. 2):

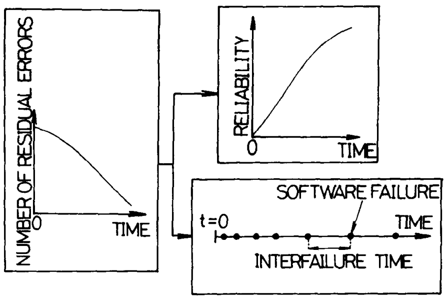
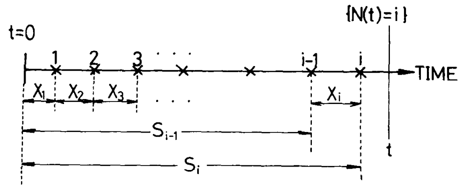


Fig. 1 Software reliability growth.



(X : Error-Detection or Failure-Occurrence)

Fig. 2 Random variables for software reliability measurement.

X_i = the time-interval between the $(i-1)$-th and i-th failures (or error detections),

S_i = the time of the i-th failure,

$N(t)$ = the cumulative number of errors (or failures) detected in the time-interval $(0, t]$.

Figure 2 shows that the event $\{N(t)=i\}$ has occurred. From the above definitions, we have the following relationships:

$$S_i=\sum_{k=1}^{i} X_k,\ X_i=S_i-S_{i-1} \quad (i=1, 2, \cdots; X_0=S_0=0). \tag{1}$$

Let $\{N(t), t\geq 0\}$ be a counting process that has independent increments so that the numbers of errors detected during disjoint time-intervals are independent. The notation $\Pr\{A\}$ means the probability of event A. An SRGM based on a *nonhomogeneous Poisson process* (NHPP) [21] with mean value function $H(t)$ can be formulated according to the definitions and assumptions above [3, 9] as:

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

where $H(t)$ is a *mean value function* indicating the expected cumulative number of errors detected up to testing time t. If we let

$$H(t)=\int_0^t h(x)dx, \tag{3}$$

the $h(t)$ is called an *intensity function* of the NHPP, which means the instantaneous error-detection rate.

Defining a as the expected cumulative number of errors to be eventually detected, that is, the expected initial error content to be estimated, if $a=H(\infty)$ (as is usual in software reliability growth modeling), we can easily show that

$$\lim_{t\to\infty} \Pr\{N(t)=n\}=\frac{a^n}{n!}e^{-a} \quad (n=0, 1, 2, \cdots). \tag{4}$$

Equation (4) implies that $N(t)$ obeys a Poisson distribution with meãn a after an infinitely long period of testing.

2.2 Reliability Assessment Measures

Several quantitative measures for software reliability assessment are derived from the NHPP with $H(t)$ given by Eq. (2).

Let $\bar{N}(t)$ denote the number of errors remaining in the software at testing time t, namely $N(\infty)-N(t)$. Then, the expectation $\mathrm{E}[\bar{N}(t)]$ and the variance $\mathrm{Var}[\bar{N}(t)]$ of $\bar{N}(t)$ are given by

$$\begin{aligned} n(t) &= \mathrm{E}[\bar{N}(t)] \\ &= a-H(t) \\ &= \mathrm{Var}[\bar{N}(t)]. \end{aligned} \tag{5}$$

The so-called software reliability is the conditional survival probability of X_i on the condition that $S_{i-1}=t$, and is given by

$$\begin{aligned} R(x\,|\,t) &\equiv \Pr[X_i>x\,|\,S_{i-1}=t\} \\ &= \exp[-\{H(t+x)-H(t)\}] \quad (t\geq 0,\, x\geq 0), \end{aligned} \tag{6}$$

which is independent of the number of failures $i(i=1, 2, \cdots)$. The software reliability in Eq. (6) represents the probability that a software failure does not occur in $(t, t+x]$.

The mean time between software failures (MTBF or MTBSF) has been often adopted as a reliability measure when data on the times of software failures are available [5]. Since the probability distribution of $S_i(i=1, 2, \cdots)$ for the NHPP model given by Eqs. (2) and (4) is improper, the MTBF does not exist [22]. However, a measure of the MTBF can be alternatively obtained by calculating the inverse of the instantaneous error-detection rate $h(t)$ (the intensity function of an NHPP given in Eq. (3)) as

$$\mathrm{MTBF}(t)=\frac{1}{h(t)}. \tag{7}$$

Equation (7) is called the instantaneous MTBF at testing time t [9].

2.3 Existing NHPP Models

A software reliability growth curve representing a relationship between the time span of software testing and the cumulative number of detected errors is observed in a software error-detection process during the testing phase. There are two types of shape for the observed software reliability growth curves: exponential and S-shaped growth curves (see Fig. 3). The SRGM's describing exponential and S-shaped growth curves are called exponential and S-shaped SRGM's, respectively. Several existing SRGM's based on NHPP's are briefly summarized in the following.

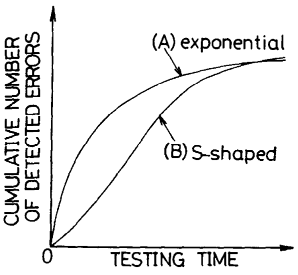


Fig. 3 Exponential and S-shaped software reliability growth curves.

Assuming that the expected number of errors detected per unit testing time (the instantaneous error-detection rate) is proportional to the current residual error content, many SRGM's are formulated as

$$\frac{dH(t)}{dt}=b(t)[a-H(t)] \quad (b(t)>0,\, t\geq 0), \tag{8}$$

where $b(t)$ is the error-detection rate per error at testing time t. Solving the differential equation (8) in terms of $H(t)$ under the condition $H(0)=0$ yields

$$H(t)=a\left(1-\exp\left[-\int_0^t b(x)dx\right]\right) \quad (t\geq 0). \tag{9}$$

If $b(t)=b$ (constant), then we have the *exponential SRGM* proposed by Goel and Okumoto [11, 23], which describes a software failure-occurrence process in the testing. The mean value function showing an exponential growth curve is given by

$$H(t)\equiv m(t)=a(1-e^{-bt}) \quad (a>0,\, b>0). \tag{10}$$

In contrast to the exponential SRGM with the homogeneous error-detection rate b (which implies a constant error-detection rate throughout the testing), the detectability of an error is considered to be nonhomogeneous over the testing period, since the errors detected early in the testing are different from those detected later on. Then, assuming that there are two types of error, of which Type 1 (Type 2) errors are easy (difficult) to detect, Yamada et al. [24, 25] proposed a nonhomogeneous error-detection rate model in which error detection processes for Type 1 and 2 errors are respectively described by the exponential SRGM above. This NHPP model, called the *modified exponential SRGM*, has a mean value function of

$$H(t)\equiv m_p(t)=a\sum_{i=1}^{2} p_i\,(1-e^{-b_i t})$$

$$(0<b_2<b_1<1,\ \sum_{i=1}^{2} p_i=1,\ 0<p_i<1\ (i=1,2)),\quad (11)$$

where b_i is the error-detection rate per Type i error ($i=1, 2$), and $p_i a$ is the expected initial error content of Type i error ($i=1, 2$). From Eq. (11), the error-detection rate per error for the modified exponential SRGM is given by

$$b(t)=\sum_{i=1}^{2}\left[\frac{p_i e^{-b_i t}}{p_1 e^{-b_1 t}+p_2 e^{-b_2 t}}\right] b_i. \quad (12)$$

In a software error-removal process it should be assumed that a testing process consists of not only a software failure-occurrence process, but also a software error-isolation process. In Eq. (8), if $b(t)=b$ and $m(t)$ in Eq. (10) is substituted for a, then we have the *delayed S-shaped SRGM* proposed by Yamada et al. [26, 27] for such an error-detection process:

$$H(t)\equiv M(t)=a[1-(1+bt)e^{-bt}]\quad (a>0,\ b>0),\quad (13)$$

which shows an S-shaped growth curve. The parameter b in Eq. (13) represents the failure-occurrence rate (and the error-isolation rate). The error-detection rate per error for the delayed S-shaped SRGM is given by

$$b(t)=\frac{b^2 t}{(1+bt)}. \quad (14)$$

Another S-shaped SRGM was proposed by Ohba et al. [28, 29]. It is called the *inflection S-shaped SRGM*, and describes a software failure-occurrence process with a mutual dependency of detected errors. In the error-detection process, the more failures are detected, the more undetected failures become detectable. For such a faillure-detection process, if in Eq. (8)

$$b(t)=b\left\{r+(1-r)\frac{H(t)}{a}\right\}, \quad (15)$$

then we have a mean value function

$$H(t)\equiv I(t)=\frac{a(1-e^{-bt})}{(1+ce^{-bt})}\ (a>0,\ b>0,\ c=(1-r)/r>0), \quad (16)$$

which shows an S-shaped growth curve. The parameters b and c represent the failure-detection rate and the inflection factor, respectively. The error-detection rate per error for the inflection S-shaped SRGM is given by

$$b(t)=\frac{b}{1+ce^{-bt}}. \quad (17)$$

We need to consider the effect of testing-effort on software reliability growth in order to develop more realistic SRGM's. The testing-effort is measured by the amount of man-power, the CPU time, the number of executed test cases, and so on. On the assumption that the error-detection rate per error is proportional to the current error content and the proportionality is the current testing-effort expenditures, if in Eq. (8)

$$b(t)=rw(t), \quad (18)$$

then we have the *testing-effort dependent SRGM* proposed by Yamada et al. [20, 30, 31]:

$$H(t)\equiv T(t)=a(1-\exp[-rW(t)])\quad (a>0,\ r>0), \quad (19)$$

$$W(t)=\int_0^t w(x)dx, \quad (20)$$

where r is the error-detection rate per unit testing-effort expenditures. Yamada et al. [31] offered a Weibull curve as the testing-effort function $w(t)$ in Eqs. (18) and (20) because of its flexibility for describing a number of testing-effort expenditure patterns:

$$w(t)=\alpha\beta m t^{m-1}\exp[-\beta t^m]\quad (\alpha>0,\ \beta>0,\ m>0), \quad (21)$$

where α, β, and m are constant parameters for specifying the function form. In particular, the parameter α represents the total testing-effort required by the testing. The parameters α, β, and m can be estimated by the method of least-squares for the observed testing-effort data. When $m=1$ and $m=2$, we have exponential and Rayleigh testing-effort functions, respectively. From Eq. (20), the total testing-effort in the testing time-interval $(0, t]$ is given by

$$W(t)=\alpha(1-\exp[-\beta t^m]). \quad (22)$$

Therefore, from Eqs. (19) and (22), the expected number of errors to be eventually detected is given by

$$T(\infty)=a(1-e^{-r\alpha})\neq a. \quad (23)$$

Equation (23) means that even if a software is tested for an infinitely long time, some errors will not be detected, because the testing-effort function tends to zero as $t\to\infty$.

Generally, software reliability assessment during the testing phase is closely related to the quality and quantity of executed test-cases. Then, defining a testing-domain function given by

$$u(t)=a(1-pe^{-vt})\quad (a>0,\ v>0,\ 0<p\leq 1), \quad (24)$$

and, in Eq. (8) if $b(t)=b$ and $u(t)$ in Eq. (24) is substituted for a, then we have the *SRGM with testing-domain* proposed by Ohtera et al. [32, 33] to describe the error-detection process recognized by execution of test-cases:

$$H(t)\equiv D(t)=a\left[1-\frac{1}{(v-b)}\{(v-b+bp)e^{-bt}-bp\,e^{-vt}\}\right]\quad (v\neq b), \quad (25)$$

where v is the testing-domain growth rate, p is the parameter representing the error distribution patterns in a tested software, and b is the error-detection rate per error. In Eq. (24), the quantity $(1-pe^{-vt})$ is the ratio of the testing-domain coverage to the final testing-domain size ($=a$) to be covered, where $p=1$ indicates a uniform error distribution.

Besides the stochastic SRGM's discussed above, deter-

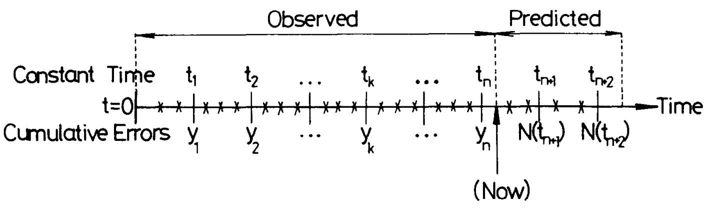

Observed
Predicted
Constant Time t_1 t_2 ... t_k ... t_n t_{n+1} t_{n+2}
t=0
Time
Cumulative Errors y_1 y_2 ... y_k ... y_n $N(t_{n+1})$ $N(t_{n+2})$
(Now)

Fig. 4 Error-detection count data (t_i, y_i) $(i=1, 2, \cdots, n)$.

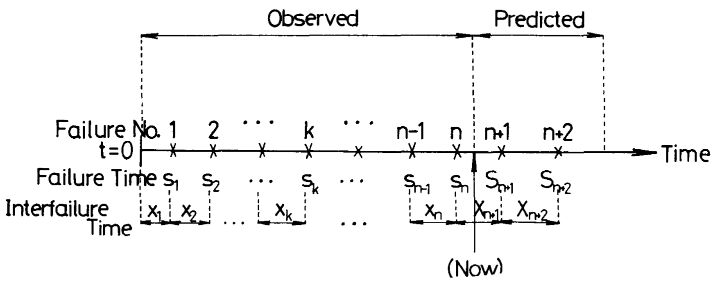

Observed
Predicted
Failure No. 1 2 ... k ... n-1 n n+1 n+2
t=0
Time
Failure Time s_1 s_2 ... s_k ... s_{n-1} s_n s_{n+1} s_{n+2}
Interfailure Time x_1 x_2 ... x_k ... x_n x_{n+1} x_{n+2}
(Now)

Fig. 5 Failure-occurrence time data s_i $(i=1, 2, \cdots, n)$.

ministic SRGM's formulated by logistic and Gompertz growth curves, have been widely used to estimate the initial error content [34, 35]. In Japan, some computer manufacturers and software houses have actually applied the logistic and Gompertz growth curve models. The expected cumulative number of errors detected up to testing time t is given for the *logistic growth curve model* as

$$L(t)=\frac{k}{1+me^{-pt}} \quad (k>0, m>0, p>0), \tag{26}$$

and for the *Gompertz growth curve model* as

$$G(t)=ka^{(b^t)} \quad (k>0, 0<a<1, 0<b<1), \tag{27}$$

where k, p, m, a, and b are constant parameters to be estimated by regression analysis. The parameter k in both models is the expected initial error content of a software.

3. Software Reliability Data Analysis

3.1 Types of Data

In order to assess software reliability during the testing phase it is necessary to estimate unknown parameters in the applied SRGM's by using the observed test data. A set of test data is usually collected and recorded in one of two typical ways.

The most commonly used (especially in Japan) and collected data are called *error-detection count data* (or grouped data [5]). These are used when the project managers want to estimate the number of errors detected during a constant time-interval of testing, that is, the realization of random variables $N(t_i)$ $(i=1, 2, \cdots, n)$. Figure 4 shows an illustration for error-detection count data where the data set on the cumulative number of detected errors, y_i, in a given time-interval $(0, t_i]$ $(i=1, 2, \cdots, n;\ 0<t_1<t_2<\cdots<t_n)$ is recorded. For the error-detection count data, we want to predict the behavior of $N(t_i)$ by a future time $t_i(i=n+1, n+2, \cdots)$, using the observed data (t_i, y_i) $(i=1, 2, \cdots, n)$.

The other commonly collected data are called *failure-occurrence time data*. These data, recorded as the times of successive software failures, that is, the realization of random variables S_i $(i=1, 2, \cdots, n)$, are the most detailed and desirable in terms of estimation, and are most often used in SRGM's based on the CPU time (or execution time) as the unit of testing time. Generally, the CPU time is measured on the processor on which the program being tested is running. As we discussed in Section 2.1, the data may be recorded as time-intervals between software failures. However, it is difficult in practice to measure and record data on the times of failures. Figure 5 shows data on the times of failures where the data set on n successive times of failures s_i $(i=1, 2, \cdots, n;\ 0\leq s_1\leq s_2\leq\cdots\leq s_n)$ is recorded. From Eq. (1), the observed data s_i $(i=1, 2, \cdots, n)$ can be converted into x_i $(i=1, 2, \cdots, n)$ by calculating $x_i=s_i-s_{i-1}$. For the data on the times of failures, we want to predict the behavior at a future time S_i or the time-interval between failures X_i $(i=n+1, n+2, \cdots)$ by using the observed data s_i $(i=1, 2, \cdots, n)$ or x_i $(i=1, 2, \cdots, n)$.

3.2 Maximum-Likelihood Estimation

Parameter estimation is of primary importance in software reliability data analysis. We discuss statistical inference procedures for the NHPP models discussed in Section 2 based on a *method of maximum-likelihood* [5, 9, 10] which is the most important and widely used formal estimation technique. Here, we assume that the

Table 1 Likelihood equations for error-detection count data.

Model	Likelihood Equations
exponential SRGM $m(t)=a(1-e^{-bt})$	$a=\dfrac{y_n}{(1-e^{-bt_n})}$ $\dfrac{y_n t_n e^{-bt_n}}{(1-e^{-bt_n})}=\sum_{k=1}^{n}\dfrac{(y_k-y_{k-1})(t_k e^{-bt_k}-t_{k-1}e^{-bt_{k-1}})}{(e^{-bt_{k-1}}-e^{-bt_k})}$
modified exponential SRGM $m_p(t)=a\sum_{i=1}^{2}p_i(1-e^{-b_it})$	$a=\dfrac{y_n}{\sum_{i=1}^{2}p_i(1-e^{-b_it_n})}$ $\dfrac{y_n t_n e^{-b_jt_n}}{\sum_{i=1}^{2}p_i(1-e^{-b_it_n})}=\sum_{k=1}^{n}\dfrac{(y_k-y_{k-1})(t_k e^{-b_jt_k}-t_{k-1}e^{-b_jt_{k-1}})}{\{\sum_{i=1}^{2}p_i(e^{-b_it_{k-1}}-e^{-b_it_k})\}}$ $(j=1,2)$
delayed S-shaped SRGM $M(t)=a[1-(1+bt)e^{-bt}]$	$a=\dfrac{y_n}{[1-(1+bt_n e^{-bt_n})]}$ $\dfrac{y_n t_n^2 e^{-bt_n}}{[1-(1+bt_n e^{-bt_n})]}=\sum_{k=1}^{n}\dfrac{(y_k-y_{k-1})(t_k^2 e^{-bt_k}-t_{k-1}^2 e^{-bt_{k-1}})}{\{(1+bt_{k-1})e^{-bt_{k-1}}-(1+bt_k)e^{-bt_k}\}}$
inflection S-shaped SRGM $I(t)=\dfrac{a(1-e^{-bt})}{(1+c\cdot e^{-bt})}$	$a=\dfrac{y_n(1+c\cdot e^{-bt_n})}{(1-e^{-bt_n})}$ $\sum_{k=1}^{n}(y_k-y_{k-1})[\dfrac{(t_k e^{-bt_k}-t_{k-1}e^{-bt_{k-1}})}{(e^{-bt_{k-1}}-e^{-bt_k})}+\dfrac{c\cdot t_k e^{-bt_k}}{(1+c\cdot e^{-bt_k})}+\dfrac{c\cdot t_{k-1}e^{-bt_{k-1}}}{(1+c\cdot e^{-bt_{k-1}})}]$ $=\dfrac{y_n t_n e^{-bt_n}(1-c+2c\cdot e^{-bt_n})}{(1-e^{-bt_n})(1+c\cdot e^{-bt_n})}$ (for specified c)
testing-effort dependent SRGM $T(t)=a[1-e^{-r\cdot W(t)}]$	$a=\dfrac{y_n}{[1-e^{-rW(t_n)}]}$ $\dfrac{y_n W(t_n)e^{-rW(t_n)}}{[1-e^{-rW(t_n)}]}=\sum_{k=1}^{n}\dfrac{(y_k-y_{k-1})[W(t_k)e^{-rW(t_k)}-W(t_{k-1})e^{-rW(t_{k-1})}]}{[e^{-rW(t_{k-1})}-e^{-rW(t_k)}]}$

mean value function $H(t)$ includes N model parameters w_i $(i=1, 2, \cdots, n)$ as well as parameter a, where $\mathbf{w}=(w_1, w_2, \cdots, w_N)$. We use **t**, **y**, and **s** to denote $(t_1, t_2, \cdots, t_n)$, $(y_1, y_2, \cdots, y_n)$, and $(s_1, s_2, \cdots, s_n)$, respectively.

Suppose that the error-detection count data (t_i, y_i) $(i=1, 2, \cdots, n)$ are observed during the testing phase. Then, the joint probability mass function of the observed data, that is, the likelihood function for the unknown parameters in an NHPP model with $H(t)$, is given by

$$L(a, \mathbf{w}\,|\,\mathbf{t}, \mathbf{y})=\Pr\{N(t_1)=y_1, N(t_2)=y_2, \cdots, N(t_n)=y_n\}$$

$$=\prod_{i=1}^{n}\frac{\{H(t_i)-H(t_{i-1})\}^{y_i-y_{i-1}}}{(y_i-y_{i-1})!}\times\exp[-\{H(t_i)-H(t_{i-1})\}], \quad (28)$$

where $t_0=0$ and $y_0=0$. Taking the natural logarithm of Eq. (28) and equating its partial derivatives with respect to the unknown parameters to zero yields

$$\frac{\partial \ln L(a, \mathbf{w}\,|\,\mathbf{t}, \mathbf{y})}{\partial a}=\frac{\partial \ln L(a, \mathbf{w}\,|\,\mathbf{t}, \mathbf{y})}{\partial w_i}=0 \quad (i=1, 2, \cdots, N). \quad (29)$$

Then, the $(N+1)$ maximum-likelihood estimates $\hat{a}$ and

Table 2 Likelihood equations for failure-occurrence time data.

Model	Likelihood Equations
exponential SRGM $m(t)=a(1-e^{-bt})$	$a=\dfrac{n}{(1-e^{-bs_n})}$ $\dfrac{n}{b}=\sum_{k=1}^{n}s_k+\dfrac{ns_n e^{-bs_n}}{(1-e^{-bs_n})}$
modified exponential SRGM $m_p(t)=a\sum_{i=1}^{2}p_i(1-e^{-b_i t})$	$a=\dfrac{n}{\sum_{i=1}^{2}p_i(1-e^{-b_i s_n})}$ $\dfrac{ns_n e^{-b_j s_n}}{\sum_{i=1}^{2}p_i(1-e^{-b_i s_n})}=\sum_{k=1}^{n}\dfrac{(e^{-b_j s_k}-b_j s_k e^{-b_j s_k})}{\{\sum_{i=1}^{2}p_i b_i e^{-b_i s_k}\}}$ $(j=1,2)$
delayed S-shaped SRGM $M(t)=a[1-(1+bt)e^{-bt}]$	$a=\dfrac{n}{[1-(1+bs_n e^{-bs_n})]}$ $\dfrac{2n}{b}=\sum_{k=1}^{n}s_k+\dfrac{nbs_n^2 e^{-bs_n}}{[1-(1+bs_n)e^{-bs_n}]}$
inflection S-shaped SRGM $I(t)=\dfrac{a(1-e^{-bt})}{(1+c\cdot e^{-bt})}$	$a=\dfrac{n(1+c\cdot e^{-bs_n})}{(1-e^{-bs_n})}$ $\dfrac{ns_n e^{-bs_n}(1+c)}{(1-e^{-bs_n})(1+c\cdot e^{-bs_n})}=\dfrac{n}{b}-\sum_{k=1}^{n}s_k+2\sum_{k=1}^{n}\dfrac{c\cdot s_k e^{-bs_k}}{(1+c\cdot e^{-bs_k})}$ (for specified c)

$\hat{w}_i$ $(i=1, 2, \cdots, N)$ can be obtained by solving the simultaneous likelihood equations (29). Table 1 summarizes the likelihood equations for the existing NHPP models discussed in Section 2.3, which can be numerically solved.

On the other hand, suppose that the failure-occurrence time data s_i $(i=1, 2, \cdots, n)$ are observed during the testing phase. Then, the joint density function of the observed data, that is, the likelihood function for the unknown parameters in an NHPP model with $H(t)$, is given by

$$L(a, \mathbf{w}\,|\,\mathbf{s})=\exp\,[-H(s_n)]\prod_{i=1}^{n}h(s_i). \tag{30}$$

Taking the natural logarithm of Eq. (30) and equating its partial derivatives with respect to the unknown parameters to zero yields

$$\frac{\partial \ln L(a, \mathbf{w}\,|\,\mathbf{s})}{\partial a}=\frac{\partial \ln L(a, \mathbf{w}\,|\,\mathbf{s})}{\partial w_i}=0 \quad (i=1, 2, \cdots, N). \tag{31}$$

Then, the $(N+1)$ maximum-likelihood estimates $\hat{a}$ and $\hat{w}_i(i=1, 2, \cdots, N)$ can obtained by solving the simultaneous likelihood equations (31). Table 2 summarizes the likelihood equations for the existing NHPP models discussed in Section 2.3, which can be solved numerically.

Therefore, using the model parameters estimated above, we can obtain the maximum-likelihood estimates of the mean value function $H(t)$ and the reliability assessment measures, such as $n(t)$, $R(x\,|\,t)$, and MTBF(t), discussed in Section 2.2. For example, these estimates for the exponential SRGM are given by using the maximum-likelihood estimates $\hat{a}$ and $\hat{b}$ as

$$\begin{aligned}
&\hat{m}(t)=\hat{a}(1-e^{-\hat{b}t}),\\
&\hat{n}(t)=\hat{a}e^{-\hat{b}t},\\
&\hat{R}(x\,|\,t)=\exp\,[-\hat{a}\{e^{-\hat{b}t}-e^{-\hat{b}(t+x)}\}],\\
&\widehat{\mathrm{MTBF}}(t)=e^{\hat{b}t}/(\hat{a}\hat{b}).
\end{aligned}$$

In particular, to evaluate the variability of $N(t)$, that is, the number of errors detected (or failures occurred) by time t, we can approximately obtain the confidence bounds for $N(t)$ as

$$\hat{H}(t) \pm K_\gamma \sqrt{\hat{H}(t)}, \tag{32}$$

where K_γ is $100(1+\gamma)/2$ percentile of the standard normal distribution.

It is important to perform a goodness-of-fit test to check statistically whether the applied SRGM provides a good fit with the observed data. We discuss the goodness-of-fit test based on the Kolmogorov-Smirnov (K-S) test statistics (e.g. [36]) for an NHPP model [9, 37], which is useful even if the sample size of the observed data is small. The Kolmogorov-Smirnov test statistic is given by

$$D = \max_{1 \le i \le n} \{D_i\}, \tag{33}$$

$$D_i = \max\left\{ \left| \frac{\hat{H}(t_i)}{\hat{H}(t_n)} - \frac{y_i}{y_n} \right|, \left| \frac{\hat{H}(t_i)}{\hat{H}(t_n)} - \frac{y_{i-1}}{y_n} \right| \right\}, \tag{34}$$

for the error-detection count data (t_i, y_i) $(i=1, 2, \cdots, n)$, and is given by

$$D = \max_{1 \le i \le n-1} \{D_i\}, \tag{35}$$

$$D_i = \max\left\{ \left| \frac{\hat{H}(s_i)}{\hat{H}(s_n)} - \frac{i}{n-1} \right|, \left| \frac{\hat{H}(s_i)}{\hat{H}(s_n)} - \frac{i-1}{n-1} \right| \right\}, \tag{36}$$

for the failure-occurrence time data s_i $(i=1, 2, \cdots, n)$. The values of the test statistics in Eqs. (33) and (35) are compared with the critical values $D_{n;\alpha}$ and $D_{n-1;\alpha}$ with the sample sizes n and $(n-1)$ for the specified level of significance α, respectively. The critical values $D_{n;\alpha}$ and $D_{n-1;\alpha}$ are available from statistical tables (e.g. [9, 36, 38]). If the calculated value D in Eq. (33) or Eq. (35) is less than the selected critical value, then it can be concluded that the observed data fit the applied SRGM.

3.3 Procedures and Tools

The procedures of software reliability data analysis and assessment discussed above are shown in Fig. 6. Yamada et al. [9, 39] developed a software reliability evaluation tool called SRET in which the analysis and assessment procedures shown in Fig. 6 are implemented in a program package, using the BASIC language on MS-DOS. SRET uses three SRGM's based on a NHPP, namely, the exponential, delayed S-shaped, and inflection S-shaped SRGM's, and two deterministic SRGM's, namely, the logistic and Gompertz growth curve models. SRET is very useful in the testing phase, since the software engineers or managers can perform software reliability assessment easily in an interactive mode without knowing the details of the process of data analysis.

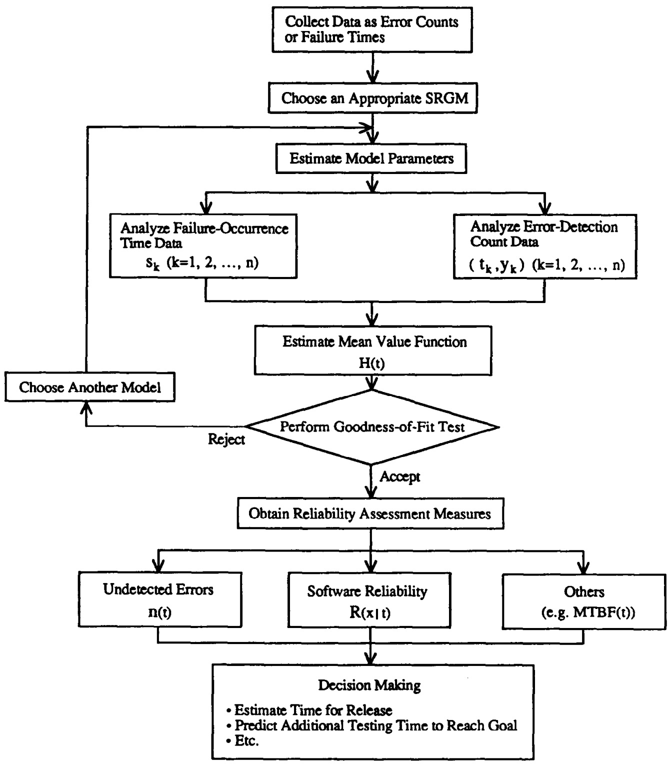
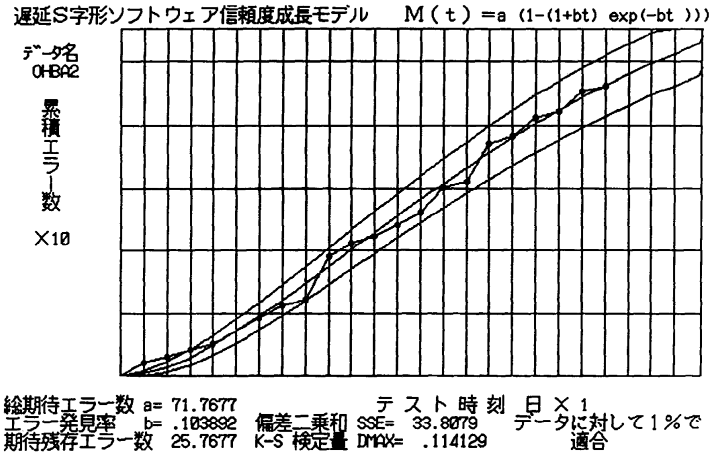

Collect Data as Error Counts or Failure Times
Choose an Appropriate SRGM
Estimate Model Parameters
Analyze Failure-Occurrence Time Data s_k (k=1, 2, ..., n)
Analyze Error-Detection Count Data (t_k,y_k) (k=1, 2, ..., n)
Estimate Mean Value Function H(t)
Choose Another Model
Perform Goodness-of-Fit Test
Reject
Accept
Obtain Reliability Assessment Measures
Undetected Errors n(t)
Software Reliability R(x|t)
Others (e.g. MTBF(t))
Decision Making
• Estimate Time for Release
• Predict Additional Testing Time to Reach Goal
• Etc.

Fig. 6 Analysis and assessment procedures in the SRET.

Table 3 Examples of software quality/reliability assessment tools developed in Japan.

Tool	Integrated SRGM	Developer	Reference
SORPS	•exponential SRGM •delayed S-shaped SRGM •inflection S-shaped SRGM	IBM Japan	[18]
SPARC	•delayed S-shaped SRGM •logistic growth curve model •Gompertz growth curve model	Toshiba	[40]
Software Reliability Evaluation Program	•exponential SRGM •delayed S-shaped SRGM •inflection S-shaped SRGM •logistic growth curve model •Gompertz growth curve model	Toshiba Engineering	[41]
SOREM	•exponential SRGM •delayed S-shaped SRGM •logistic growth curve model •Gompertz growth curve model	NEC	[42]



```
総期待エラー数 a= 71.7677                テ ス ト 時 刻  日 X 1
エラー発見率   b= .103892  偏差二乗和 SSE=  33.8079   データに対して1%で
期待残存エラー数 25.7677  K-S 検定量 DMAX=  .114129        適合
```

Fig. 7 Data analysis based on the delayed S-shaped SRGM for DATA 1.

Several tools based on similar ideas to SRET have been developed in order to assess the quality and reliability of the developed software product from the standpoint of a SRGM. Table 3 shows examples of tools developed by some Japanese computer manufacturers and software houses.

4. Examples of Application

Here, assisted by the SRET discussed in Section 3.3, we describe some examples of software reliability assessment to give numerical illustrations of the application of SRGM's based on an NHPP. The following sets of test data observed in actual software testing are used:

DATA 1: Online data entry software package test data

DATA 2: Large-scale bond/stock trading management system test data.

Both the data sets are error-detection count data.

The first data set (t_i, y_i) $(i=1, 2, \cdots, 20)$ was cited by Ohba [18]. The software, developed by IBM Japan, Ltd., consists of approximately 40,000 lines of object code. The testing time was measured on the basis of the number of shifts (days) spent running test cases and analyzing the results. The number of test personnel was constant through the testing. Figure 7 shows the analysis results and the estimated mean value function

$$\hat{M}(t)=71.77[1-(1+0.1039t)e^{-0.1039t}], \tag{37}$$

along with the 90 percent confidence bounds given by using the delayed S-shaped SRGM. By calculating Eqs. (33) and (34), the K-S goodness-of-fit test shows that

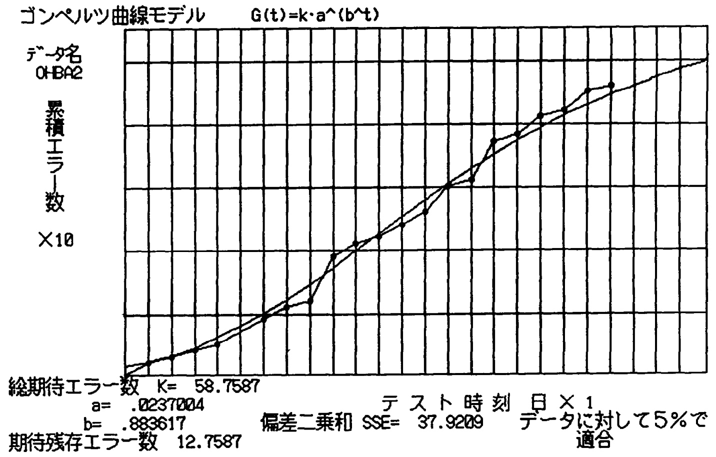


Fig. 8 Data analysis based on the Gompertz growth curve model for DATA 1.

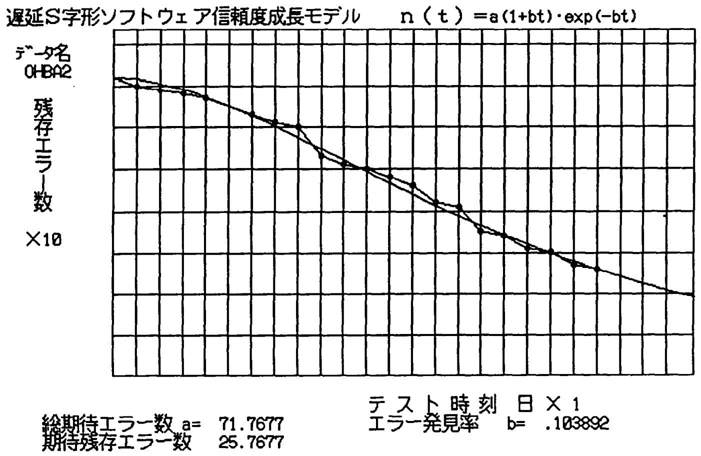


Fig. 9 Estimated expected number of remaining errors based on the delayed S-shaped SRGM for DATA 1.

the delayed S-shaped SRGM with estimated mean value function $\hat{M}(t)$ in (37) is well-fitted to the observed test data. SRET can calculate the sum of square errors, SSE, between the actual cumulative number of errors y_i and the estimated number of errors $\hat{y}_i(=\hat{H}(t_i))$ detected in the time-interval $(0, t_i]$ $(i=1, 2, \cdots, n)$ in order to compare the estimation accuracy of the applied models:

$$\mathrm{SSE}=\sum_{i=1}^{n}(y_i-\hat{y}_i)^2 \quad (\hat{y}_i=\hat{H}(t_i)). \tag{38}$$

For example, compared with the estimated Gompertz growth curve model

$$\hat{G}(t)=58.76(0.0237)^{(0.8836)^t}, \tag{39}$$

which is shown in Fig. 8, we have SSE=33.81 for the delayed S-shaped SRGM, and SSE=37.92 for the Gompertz growth curve model. Thus, the delayed S-shaped SRGM gives better results than the Gompertz growth curve model. According to the field tracking data of this software product, the actual value of the parameter a, that is, the initial error content, is 69 (the estimated value is 72). In this case, the exponential SRGM with the mean value function in Eq. (10) does not correspond to the observed test data at all, since the estimated initial error content is infinite. Thus, we may conclude that the delayed S-shaped SRGM accurately describes the observed software reliability growth. On the basis of the data analysis above, the estimated expected number of remaining errors $\hat{n}(t)$ in Eq. (5) and the estimated software reliability $\hat{R}(x|t)$ in Eq. (6) are respectively shown in Figs. 9 and 10, where the termination time of testing, t, in the estimated software reliability is 21 (days).

The second data set (t_i, y_i) $(i=1, 2, \cdots, 26)$ was ob-

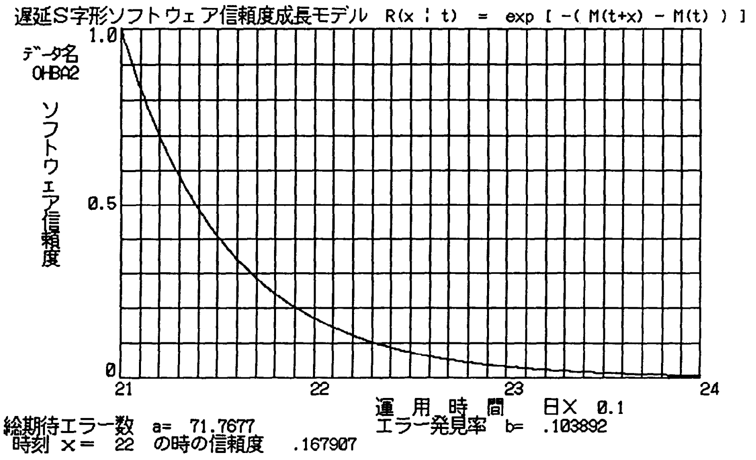


Fig. 10 Estimated software reliability based on the delayed S-shaped SRGM for DATA 1.

served during the system testing phase. The software system, developed by the Mitsubishi Trust and Banking Co. and Nissho Electronics Co., consists of approximately 50 online programs, 150 batch programs, and their support programs, which form a total of approximately 3,000 modules written in COBOL and Assembler languages. In this case, the testing time was measured on the basis of calendar time (days). The observed software reliability growth curve of detected errors was obviously S-shaped, and therefore four S-shaped SRGM's, namely, the delayed and inflection S-shaped SRGM's and the logistic and Gompertz growth curve models, were applied. From synthetical evaluation, the result of software reliability assessment based on the delayed S-shaped SRGM was considered to be best. Figure 11 shows the analysis results using the delayed S-shaped SRGM:

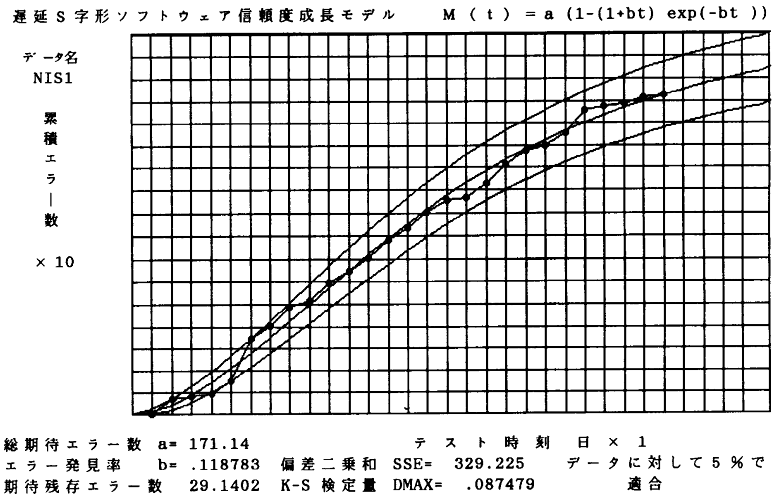
$$\hat{M}(t)=171.1[1-(1+0.1188t)e^{-0.1188t}]. \tag{40}$$

5. Concluding Remarks

From the standpoint that the basic approach to software quality management should be to ensure software reliability, namely, a quality characteristic as taken-for-granted quality, we have discussed a method of software reliability measurement and assessment based on the SRGM's developed in Japan. In particular, we have discussed the quantitative measures for software reliability assessment and the maximum-likelihood estimation for data analysis based on SRGM's described by an NHPP (NHPP models). The data sets for software reliability analysis, which are observed during the testing phase, have been classified into two types: error-detection count data sets and failure-occurrence time data sets. Some applications of several leading SRGM's summarized in this paper to observed test data taken from actual software projects have also been illustrated by a software reliability assessment tool called SRET.

Besides the SRGM's discussed in this paper, several techniques for software quality or reliability assessment have been developed in Japan. Ido et al. [43] have improved the conventional capture-recapture method based on the production unit concept, in which pseudo-errors are seeded before testing and the capture ratios of the pseudo-errors and the inherent errors detected during the testing phase are measured to estimate the initial error content. Yamada et al. [44–46] have proposed several discrete SRGM's that adopt the number of test runs or the number of executed test-cases as the unit of testing time. Ohba [47, 48] and Yamada et al. [49] have offered a viable method for software quality assessment, which integrates the conventional capture-recapture method and the SRGM's described by an NHPP. Analyzing the relationships between the error rate of the program and a variety of environment factors affecting software development, such as the programmer's skill, Takahashi and Kamayachi [50] have developed a Dendenkosha Information Processing System (DIPS) reliability model based on a multiple regression analysis. On the basis of the hyper-geometric distribution and the new concepts related to the test-run, Tohma et al. [51, 52] have proposed software reliability models for estimating the number of residual software errors. Integrating the software reliability growth process, Sunazuka [53] has modeled a software quality improvement process describing an error-introduction phenomenon during software production and an error-removal phenomenon during testing. Sando and Fujii [54] have discussed the reliability growth analysis of a discrete-type program, based on the idea of quasi-error seeding as an improved testing method.

The results of software reliability assessment based on SRGM's discussed in this paper should be used for software development management. Some promising applications are found in the areas of a testing-effort



```
総期待エラー数 a= 171.14                    テ ス ト 時 刻  日 × 1
エラー発見率   b= .118783  偏差二乗和 SSE=  329.225    データに対して５％で
期待残存エラー数  29.1402  K-S 検定量 DMAX=  .087479         適合
```

Fig. 11 Data analysis based on the delayed S-shaped SRGM for DATA 2.

control problem and an optimal software release problem. The testing-effort control problem based on the testing-effort dependent SRGM discussed in section 2.3, which predicts the testing-effort required to achieve the objective number of errors detected by the testing, has been discussed by Ohtera et al. [55, 56] and Yamada et al. [57, 58]. The optimal software release problem based on the SRGM's suggested by Koch and Kubat [59] and Okumoto and Goel [60], which decides when to stop software testing and transfer a software system to the user, has been discussed by Yamada et al. [61-63] and Ohtera and Yamada [64].

The SRGM's discussed in this paper are useful primarily in assessing and monitoring software reliability, viewed as taken-for-granted quality. Such SRGM's should provide software engineers and managers with guidance related to many decision-making problems for successful software development projects. For this purpose, more useful and applicable SRGM's that incorporate information about a software system and the development process will be needed.

Acknowledgments

The author thanks Professor S. Osaki and Mr. J. Hishitani of Hiroshima University for their helpful suggestions, and Mr. N. Kishimoto of the Mitsubisi Trust and Banking Co. and Mr. S. Kimura and Mr. S. Nakamura of Nissho Electronics Co. for valuable test data and useful comments.

References

1. Matsumoto, Y. and Ohno, Y. (eds.) *Japanese Perspectives in Software Engineering*, Addison-Wesley, Singapore (1989).
2. IEEE, *IEEE Standard Glossary of Software Engineering Terminology*, ANSI/IEEE, Std 729-1983, New York (1983).
3. Goel, A. L. Software reliability models: Assumptions, limitations, and applicability, *IEEE Trans. Softw. Eng.* SE-**11**, 12 (Dec. 1985), 1411-1423.
4. Mellor, P. Software reliability modeling: The state of the art, *Information and Software Technology*, **29**, 2 (Mar. 1987), 81-98.
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.* SE-**8**, 4 (Aug. 1982), 354-371.
7. Shanthikumar, J. G. Software reliability models: A review, *Microelectronics and Reliability*, **23**, 5 (1983), 903-943.
8. Shooman, M. L. *Software Engineering: Design, Reliability, and Management*, McGraw-Hill, New York (1983).
9. Yamada, S. *Software Reliability Assessment Technology* (in Japanese), HBJ Japan, Tokyo (1989).
10. Yamada, S. and Ohtera, H. *Software Reliability: Theory and Practical Application* (in Japanese), Soft Research Center, Tokyo (1990).
11. Goel, A. L. and Okumoto, K. Time-dependent error-detection rate model for software reliability and other performance measures, *IEEE Trans. Reliability*, R-**28**, 3 (Aug. 1979), 206-211.
12. Jelinski, Z. and Moranda, P. B. Software reliability research, in *Statistical Computer Performance Evaluation*, ed. Freiberger, W., Academic Press, New York (1972), 465-484.
13. Kremer, W. Birth-death and bug counting, *IEEE Trans. Reliability*, R-**32**, 1 (Apr. 1983), 27-47.
14. 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.
15. Moranda, P. B. Event-altered rate models for general reliability analysis, *IEEE Trans. Reliability*, R-**28**, 5 (Dec. 1979), 376-381.
16. Musa, J. D. The measurement and management of software reliability, *Proc. IEEE*, **68**, 9 (Setp. 1980), 1131-1143.
17. 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.
18. Ohba, M. Software reliability analysis models, *IBM J. Res. Dev.* **28**, 3 (July 1984), 428-443.
19. Yamada, S. and Osaki, S. Software reliability growth modeling: Models and applications, *IEEE Trans. Softw. Eng.*, SE-**11**, 12 (Dec. 1985), 1431-1437.
20. Yamada, S., Ohtera, H. and Narihisa, H. Software reliability growth models with testing-effort, *IEEE Trans. Reliability*, R-**35**, 1 (Apr. 1986), 19-23.
21. Ascher, H. and Feingold, H. *Repairable Systems Reliability: Modeling, Inference, Misconceptions, and Their Causes*, Marcel Dekker, New York (1984).
22. Hishitani, J., Yamada, S. and Osaki, S. Comparison of two estimation methods of the mean time-interval between software

failures, *Proc. IEEE Int. Phoenix Conf. Computers and Communications* (1990), 418-424.

23. GOEL, A. L. Software error detection model with applications, *J. Systems and Software*, **1** (1980), 243-249.

24. YAMADA, S. and OSAKI, S. Nonhomogeneous error detection rate models for software reliability growth, in *Stochastic Models in Reliability Theory*, ed. Osaki, S. and Hatoyama, Y., *Springer-Verlag*, Berlin (1984), 120-143.

25. YAMADA, S., OSAKI, S. and NARIHISA, H. A software reliability growth model with two types of errors, *R.A.I.R.O.—Operations Research*, **19**, 1 (Feb. 1985), 87-104.

26. YAMADA, S. and OSAKI, S. Software reliability growth models and their comparisons (in Japanese), *Trans. IECE Japan*, **J65**-D, 7 (July 1982), 906-912.

27. YAMADA, S., OHBA, M. and OSAKI, S. S-shaped reliability growth modeling for software error detection, *IEEE Trans. Reliability*, R-**32**, 5 (Dec. 1983), 475-478, 484.

28. OHBA, M. Inflection S-shaped software reliability growth model, in *Stochastic Models in Reliability Theory*, ed. Osaki, S. and Hatoyama, Y., *Springer-Verlag*, Berlin (1984), 144-162.

29. OHBA, M. and YAMADA, S. S-shaped software reliability growth models, *Proc. 4th Int. Conf. Reliability and Maintainability* (1984), 430-436.

30. YAMADA, S. Test-effort dependent software reliability growth models and comparisons of goodness-of-fit (in Japanese), *Trans. IEICE Japan*, **J70**-D, 12 (1987), 2438-2445.

31. YAMADA, S., OHTERA, H. and NARIHISA, H. A testing-effort dependent reliability model for computer programs, *Trans. IECE Japan*, **E69**, 11 (Nov. 1986), 1217-1224.

32. OHTERA, H., YAMADA, s. and NARIHISA, H. Software reliability growth model for testing-domain (in Japanese), *Trans. IEICE Japan*, **J73**-D-I, 2 (Feb. 1990), 170-174.

33. OHTERA, H., YAMADA, S. and OHBA, M. Software reliability growth model with testing-domain and comparisons of goodness-of-fit, *Proc. Int. Symp. Reliability and Maintainability* (1990), 289-294.

34. KANNO, A. *Software Engineering* (in Japanese), JUSE, Tokyo (1979).

35. MITSUHASHI, T. *A Method of Software Quality Evaluation* (in Japanese), JUSE, Tokyo (1981).

36. NEAVE, H. R. and WORTHINGTON, P. L. *Distribution-Free Tests*, Unwin Hyman, London (1988).

37. GOEL, A. L. Software Reliability Modeling and Estimation Techniques, Tech. Rep. RADC-TR-82-263, Rome Air Development Center (1982).

38. CONOVER, W. J. *Practical Nonparametric Statistics* (*2nd ed.*), John Wiley & Sons, New York (1980).

39. YAMADA, S., ISOZOKAI, R. and OSAKI, S. A software reliability evaluation tool: SRET (in Japanese), *Trans. IEICE Japan*, **J72**-D-I, 1 (Jan. 1989), 24-32.

40. NAKAMURA, H., UCHIHIRA, N., SATOH, M. and MIZUTANI, H. A method of predicting software quality prior to the testing phase (in Japanese), *Nikkei Electronics*, 1985-02-25 (Feb. 1985), 233-260.

41. KOMURO, Y. Evaluation of software products' testing phase—Application to software reliability growth models, in *Reliability Theory and Application*, ed. Osaki, S. and Cao, J., *World Scientific*, Singapore (1987), 188-194.

42. UEMURA, M., YAMADA, S. and FUJINO, K. Software reliability evaluation method: Application of delayed S-shaped NHPP model and other related models, *Proc. Int. Symp. Reliability and Maintainability* (1990), 467-472.

43. IDO, S., MURATA, M. and NAKAGAWA, T. On the estimation of hidden bugs by the capture-and-recapture method and its application (in Japanese), *IPS Japan Proc. WGSE Meeting*, 19 (1981).

44. YAMADA, S., OSAKI, S. and NARIHISA, H. Software reliability growth modeling with number of test runs, *Trans. IECE Japan*, **E67**, 2 (Feb. 1984), 79-83.

45. YAMADA, S. and OSAKI, S. Discrete software reliability growth models, *J. Applied Stochastic Models and Data Analysis*, **1**, 1 (July 1985), 65-77.

46. YAMADA, S. and OSAKI, S. A generalized discrete software reliability growth model and its application, in *Reliability Theory and Applications*, ed. Osaki, S. and Cao, J., *World Scientific*, Singapore (1987), 411-421.

47. OHBA, M. Software quality = Test accuracy × Test coverage, *Proc. 6th Int. Conf. Softw. Eng.* (1982), 287-293.

48. OHBA, M. SPQL: Software test result evaluation and quality assessment, *Proc. IEEE Int. Conf. Comm.* (1984), 1011-1015.

49. YAMADA, S., OHBA, M. and OSAKI, S. S-shaped software reliability growth models and their applications, *IEEE Trans. Reliability*, R-**33**, 4 (Oct. 1984), 289-292.

50. TAKAHASHI, M. and KAMAYACHI, Y. An empirical study of a model for program error prediction, *IEEE Trans. Softw. Eng.*, SE-**15**, 1 (Jan. 1989), 82-86.

51. TOHMA, Y., TOKUNAGA, K., NAGASE, S. and MURATA, Y. Structural approach to the estimation of the number of residual software faults based on the hyper-geometric distribution, *IEEE Trans. Softw. Eng.*, SE-**15**, 3 (Mar. 1989), 345-355.

52. TOHMA, Y., JACOBY, R., MURATA, Y. and YAMAMOTO, M. Hyper-geometric distribution model to estimate the number of residual software faults, *Proc. COMPSAC 89* (1989), 610-617.

53. SUNAZUKA, T. Model for a Software quality improvement process model by matrix expressions (in Japanese), *Trans. IPS Japan*, **31**, 7 (July 1990), 1104-1112.

54. SANDO, H. and FUJII, S. Reliability growth analysis for discrete-type software using quasi-error seedings, in *Reliability Theory and Applications*, edl Osaki, S. and Cao, J., *World Scientific*, Singapore (1987), 319-327

55. OHTERA, H., YAMADA, S. and NARIHISA, H. Testing-effort control based on software reliability growth models (in Japanese), *Trans. IEICE Japan*, **J70**-D, 5 (May 1987), 889-895.

56. OHTERA, H. and YAMADA, S. Optimal allocation and control problems for software-testing resources, *IEEE Trans. Reliability*, R-**39**, 2 (June 1990), 171-176.

57. YAMADA, S., OHTERA, H. and NARIHISA, H. A testing-effort dependent software reliability model and its application, *Microelectronics and Reliability*, **27**, 3 (1987), 507-522.

58. YAMADA, S. and OHTERA, H. Software reliability growth models for testing-effort control, *European J. Operational Research*, **46**, 3 (1990), 343-349.

59. KOCH, H. S. and KUBAT, P. Optimal release time for computer software, *IEEE Trans. Softw. Eng.*, SE-**9** (May 1983), 323-327.

60. OKUMOTO, K. and GOEL, A. L. Optimum release time for software systems based on reliability and cost criteria, *J. Systems and Software*, **1** (1980), 315-318.

61. YAMADA, S., NARIHISA, H. and OSAKI, S. Optimum release policies for a software system with a scheduled software delivery time, *Int. J. Systems Science*, **15**, 8 (Aug. 1984), 905-914.

62. YAMADA, S. and OSAKI, S. Cost-reliability optimal release policies for software systems, *IEEE Trans. Reliability*, R-**34**, 5 (Dec. 1985), 422-424.

63. YAMADA, S. and OSAKI, S. Optimal software release policies with simultaneous cost and reliability criteria, *European J. Operational Research*, **31**, 1 (1987), 46-51.

64. OHTERA, H. and YAMADA, S. Optimum release-time considering an error-detection phenomenon during operation, *IEEE Trans. Reliability*, R-**39**, 5 (Dec. 1990), 596-599.

(Received November 16, 1990)