Gompertz Software Reliability Model: Estimation Algorithm and Empirical Validation ⋆

Koji Ohishi and Hiroyuki Okamura and Tadashi Dohi

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

Abstract

Gompertz curve has been used to estimate the number of residual faults in testing phases of software development, especially by Japanese software development companies. Since the Gompertz curve is a deterministic function, the curve cannot be applied to estimating software reliability which is the probability that software system does not fail in a prefixed time period. In this article, we propose a stochastic model called the Gompertz software reliability model based on non-homogeneous Poisson processes. The proposed model can be derived from the statistical theory of extreme-value, and has a similar asymptotic property to the deterministic Gompertz curve. Also, we develop an EM algorithm to determine the model parameters effectively. In numerical examples with software failure data observed in real software development projects, we evaluate performance of the Gompertz software reliability model in terms of reliability assessment and failure prediction.

Key words: software reliability model, Gompertz curve, extremal distributions, reliability assessment, NHPP, EM algorithm

1 Introduction

During the last three decades, software reliability engineering has played a central role in quantitative assessment of software quality. In particular, soft-

⋆ This research was supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Young Scientists (B) and Grant No. 15700060 (2003-2004), Exploratory Research, Grant No. 15651076 (2003-2005).

Email address: {okamu,dohi}@rel.hiroshima-u.ac.jp (Koji Ohishi and Hiroyuki Okamura and Tadashi Dohi).

ware reliability, which is defined by the probability that software system does not fail in a prefixed time period, is one of the most important measures of software quality. To obtain highly accurate estimates for the reliability, a number of software reliability models (SRMs) have been proposed, so that they could represent a variety of software development environments [1–5].

Non-homogeneous Poisson process (NHPP) models have particularly gained much popularity for software engineers as well as researchers to estimate the software reliability, and also have been utilized to predict the number of residual faults and software release time. Goel and Okumoto [6], Yamada et al. [7] and Goel [8] proposed earlier and well-known NHPP-based SRMs. Based on their NHPP-based models, many researchers developed NHPP-based SRMs to represent the situation of software development processes.

In this paper, we formulate a new NHPP-based SRM by a different approach from existing ones. Traditionally, Japanese software development companies prefer regression analysis based on deterministic functions such as Gompertz and logistic curves to estimate the number of residual faults. To the best of our knowledge, Sakata [9] was the first one who utilized the Gompertz and logistic curves to predict the number of software faults detected. The basic idea behind his method is non-linear regression, i.e., to fit non-linear deterministic curves to the cumulative number of detected faults so as to minimize the sum of squared errors. However, the regression analysis based on deterministic curves is not appropriate to estimating software reliability. In general, computing software reliability requires the definition of the event of failure and its occurrence time (failure time). In the context of regression analysis, we cannot estimate the failure time, and thus the software reliability is not defined.

Nevertheless, the Gompertz and logistic curves are still used in industry, because these curves are well fitted to the cumulative number of faults observed in existing software development processes. This fact motivates us to develop a new NHPP-based SRM which has similar behavior of the Gompertz or logistic curve.

Ohba [10] proposed a NHPP-based SRM called the inflection S-shaped model which involves similar property as the logistic curve. The mean value function of the inflection S-shaped model is identical to the deterministic logistic curve. However, since his modeling approach was based on differential equations of the mean value function, it has not explained relationship between characteristics of failure times and the cumulative number of faults (failures).

In fact, the debugging theory, or equivalently, the theory of order statistics, [11, 12] can explain the relationship for almost all existing NHPP-based SRMs. In the debugging theory, we suppose that software initially contains the finite number of faults and that each fault is detected at independent and

identically distributed random times (fault-detection time distribution). If the initial number of faults is given by a Poisson distributed random variable, the number of faults detected before arbitrary time follows the NHPP whose mean behavior is same as the cumulative distribution function (c.d.f.) of the fault-detection time. In other words, the cumulative number of faults is determined by the fault-detection time distribution in the debugging theory.

The inflection S-shaped model can also be explained in the framework of debugging theory, and is the NHPP model whose fault-detection time obeys logistic distribution. However, it should be noted that logistic distribution is defined on both negative and positive domains. That is, there are two possible NHPP-based SRMs which are based on logistic distribution, and they depend on the methods to modify the logistic distribution into the c.d.f. with positive domain. If we modify the domain of logistic distribution with truncation, then the modified logistic distribution is called a truncated logistic distribution. The NHPP-based SRM with the truncated logistic distribution is exactly reduced to the inflection S-shaped model. On the other hand, the domain is changed by taking logarithm. The SRM corresponds to a log-logistic model [13]. The details of these logistic SRMs are shown in [14].

In this paper, we focus on the Gompertz curve. The Gompertz curve is introduced by B. Gompertz [15] to explain the law of human mortality theoretically and determine the value of life contingencies. In the context of reliability engineering, this curve is, for example, used to assess the reliability growth phenomenon of hardware products (see e.g. Virene [16]). Similar to the logistic SRMs [10, 13], we formulate Gompertz SRMs based on NHPPs. In particular, we show that the proposed models can be derived from the theory of extreme-value, and their asymptotic properties are quite similar to the deterministic Gompertz curve.

Moreover, we develop an EM (expectation-maximization) algorithm [17, 18] to estimate the model parameters of Gompertz SRMs. Since the Gompertz SRM has a stronger non-linearity than the existing NHPP-based SRMs, the computation procedure of maximum likelihood estimates is more complex, although Hossain and Dahiya [19] and Knafl and Morgan [20] succeeded to solve the likelihood equations for the exponential SRM [6]. Also, classical iterative algorithms, such as the Newton's method, are possibly used to solve the likelihood equations numerically. Since the Newton's method is, however, one of unconstrained optimization algorithms, it does not often function well to compute the estimates of Gompertz SRMs. In addition, the local convergence property of the Newton's method adversely affects the estimation procedure in Gompertz SRMs. Okamura et al. [14, 21–23] developed the framework of EM algorithms in the NHPP-based SRMs, and the framework can produce efficient algorithms to find maximum likelihood estimates of almost all kinds of NHPP-based SRMs. Since the specific EM algorithms generally depend on

the fault-detection time distributions, this paper concretely presents an EM algorithm for the Gompertz SRMs in detail.

The remaining part of this paper is planned as follows. In Section 2, we introduce the basic concept of Gompertz curve, its related work and the modeling framework of NHPP-based SRMs. In Section 3, we explain statistical properties of extreme-value statistics and develop the Gompertz SRMs based on the NHPP. Section 4 is devoted to the parameter estimation problem. After describing a couple of theoretical results on the maximum likelihood estimation for the Gompertz SRM, we develop the EM algorithm to estimate model parameters. In Section 5, we investigate performance of the proposed EM algorithm in terms of the convergence property, and also compare the EM algorithm with the Newton's method from the viewpoints of both accuracy and speed of convergence. Moreover, with software fault-detection data observed in real software development projects, we apply the Gompertz SRM to assess software reliability and to predict the number of initial fault contents. We empirically conclude that our new model may function better than the existing models in several data sets, and that the model is effective on the goodness-of-fit under some information criteria.

2 Software Reliability Models

2.1 Deterministic Curve Model

Let $M(t)$ be the cumulative number of software faults detected up to time t (≥ 0). Then the Gompertz curve is given by

$$M(t) = \omega a^{b^t}, \tag{1}$$

where $\lim_{t\to\infty} M(t) = \omega$ (> 0) denotes the initial number of faults before testing and $a,\ b \in (0,1)$ are arbitrary design parameters. It is straightforward to see that the Gompertz curve draws a typical S-shaped curve and that its point of inflection is given by $t = -\log(-\log a)/\log b$. Figure 1 depicts a schematic illustration of the Gompertz curves.

The non-linear regression analysis is used to estimate model parameters, i.e., ω, a and b. Satoh [24] and Satoh and Yamada [25] introduced a discrete Gompertz curve by discretizing the differential equations for the Gompertz curve, and applied the discrete Gompertz curve to predict the number of detected software faults. As mentioned in Introduction, the deterministic curves and their regression analysis possess a couple of statistical drawbacks.

To overcome such problems, Yamada [26] considered an NHPP model with the following mean value function:

$$M(t) = \omega\{a^{b^t} - a\}. \tag{2}$$

Eq. (2) allows us to modify the Gompertz curve so that it satisfies $M(0) = 0$. Then the number of detected faults at $t = 0$ becomes 0 in the NHPP model (see Fig. 1). On the other hand, Eq. (2) poses the difficulty of computing the maximum likelihood estimates due to its strong non-linearity. An alternative way to develop the NHPP model with the Gompertz curve is the use of discontinuous mean value function. That is, if we consider the following mean value function:

$$M(t) = \begin{cases} 0, & t = 0 \\ \omega a^{b^t}, & t > 0, \end{cases} \tag{3}$$

then the NHPP also behaves like the Gompertz curve in the range of $t > 0$. However, even if we apply Eq. (3) to assessment of software reliability, the parameter estimation problem is still remained, i.e., it is difficult to derive the maximum likelihood estimates for the mean value function in Eq. (3) by using real software fault-detection data.

2.2 Software Debugging Theory

Let $N(t)$ denote the number of software faults detected by time t, and be a stochastic point process (distinguish it from the deterministic function $M(t)$). We make the following assumptions:

Assumption A: Software failures caused by software faults occur at independent and identically distributed (i.i.d.) random times having the probability distribution function $F(t)$ with density $f(t) = dF(t)/dt$.

Assumption B: The initial number of software faults, N (> 0), is positive and finite.

Although Assumption A seems to be strong for practical situation, we can observe such phenomenon in real software testing processes, especially, system test phase. Assumption A does not indicate that the software faults are all detected under only one environment. Since a software fault is detected by some specific test cases, Assumption A means that test cases are randomly selected. In fact, the correlation due to selection of test cases appears in the number of detected faults, but in macroscopic behavior of the process, the correlation is modelled by variation of fault-detection times. In addition, when

the correlation is strong so that it could not be ignored, we can use mixed-type NHPP-based SRMs such as hyperexponential SRMs [27, 28]. In fact, almost all NHPP-based SRMs can be justified under the above two assumptions.

Under the above assumptions, the probability mass function of the number of faults detected by time t is given by

$$\Pr\{N(t) = n \mid N\} = \binom{N}{n} F(t)^n \overline{F}(t)^{N-n}, \tag{4}$$

where $\overline{F}(\cdot) = 1 - F(\cdot)$. Figure 2 illustrates the configuration of software debugging theory. When the software fault-detection time obeys an exponential distribution $F(t) = 1 - \exp\{-\beta t\}$ ($\beta > 0$), then the stochastic process $\{N(t), t \geq 0\}$ is reduced to a pure birth process with absorbing boundary at $N(t) = N$, and also it is well known that such a SRM is equivalent to the Jelinski and Moranda model [29].

If the number of initial fault contents is unknown, i.e., N is unknown, it will be appropriate to assume that N is a discrete (integer-valued) random variable. Langberg and Singpurwalla [11] proved that when the initial number of software faults N obeys the Poisson distribution with parameter ω (> 0), the number of software faults experienced before time t is given by the following NHPP:

$$\Pr\{N(t) = n\} = \sum_{x=0}^{\infty} \Pr\{N(t) = n \mid x\} \frac{\omega^x e^{-\omega}}{x!} = \frac{\{\omega F(t)\}^n}{n!} e^{-\omega F(t)}. \tag{5}$$

Equation (5) is equivalent to the probability mass function of the NHPP having the mean value function $\mathrm{E}[N(t)] = M(t) = \omega F(t)$. From this modeling framework, the most NHPP-based SRMs can be derived by choosing the software fault-detection time distribution $F(t)$. If $F(t) = 1 - \exp\{-\beta t\}$, then we can derive the Goel and Okumoto model [6] with mean value function $M(t) = \omega(1 - \exp\{-\beta t\})$. If the software fault-detection time distribution is given by the gamma distributions with second order and more or the Weibull distribution, then the resulting NHPP-based SRMs become the delayed S-shaped SRM [7], Zhao and Xie model [30] or Goel model [8], respectively. Miller [12] examines further a relationship between the NHPP-based SRMs and the order statistics based on the probability distribution $F(t)$.

3 Extreme-Value Modeling

3.1 Statistics of Extremes

Statistics of extremes is an area that treats the behavior of the largest or smallest value of a given sample population. Since the seminal contribution by Gnedenko [31] and Gumbel [32], the theory on statistics of extremes is well established (see [33], [34]). Here we concern only the extremal statistics of *smallest value*. Suppose that a software error occurs on k (> 0) program codes, and that the corresponding (only one) software fault can be detected at the first time when the faulty code is encountered. Suppose that the time length when each faulty code appears is an i.i.d. random variable T_i ($i = 1, 2, \cdots, k$) having the continuous probability distribution function $G(t)$ and the density $g(t) = dG(t)/dt$. Then, the smallest time, i.e., the fault-detection time is defined by $T_{min} = \min(T_1, T_2, \cdots, T_k)$. From the i.i.d. assumption, it can be seen that the probability distribution of T_{min} and its density function are derived as

$$F_k(t) = 1 - \{1 - G(t)\}^k, \tag{6}$$
$$f_k(t) = k\{1 - G(t)\}^{k-1} g(t), \tag{7}$$

respectively. Of our concern is the property of the probability distribution function of $F_k(t)$ as the number of faulty codes remaining in the software is sufficiently large, i.e., k approaches infinity.

Gnedenko [31] and independently Gumbel [32] proved that $F_k(t)$ in Eq. (6) converges to a particular functional form that does not depend on the exact form of the distribution but rather on the behavior of the initial distribution's tail in the direction of the extreme, that is, $\lim_{k\to\infty} F_k(t)$ can be reduced to one of the three asymptotic forms (which are commonly called as extreme Type I, Type II and Type III distributions). More specifically, when suitably normalized $(T_{min} - b_k)/a_k$ possesses a non-degenerate limiting distribution $F(t)$ as k approaches infinity where a_k (> 0) and b_k are arbitrary scale and location parameters with monotone property, then the resulting $F(t)$ must be a minimum stable law, i.e., it should be identical (except for location and scale change) to one of the following distributions:

$$F(t) = \begin{cases} 1 - e^{-t^\alpha}, & t \geq 0 \quad \text{(Type I)} \\ 1 - e^{-(-t)^{-\alpha}}, & t \leq 0 \quad \text{(Type II)} \\ 1 - e^{-\exp\{t\}}, & -\infty < t < \infty \quad \text{(Type III)} \end{cases} \tag{8}$$

with α (> 0). In this case we say that the probability distribution $\lim_{k\to\infty} F_k(t) = F(t)$ belongs to the minimum domain of attraction, i.e., $\overline{F}_k(a_k t + b_k) \to_L \overline{F}(t)$, where '$\to_L$' denotes point-wise convergence at continuity points of $\overline{F}(t)$. The extreme Type I distribution is the well-known Weibull distribution. Since the extreme Type II distribution called the Fréchet distribution is defined in the range of negative real value, it is not suitable to assume as the fault-detection time distribution. In this article, we are interested in the extreme Type III distribution called the Gumbel distribution.

Kaufman et al. [35] [36] consider the renewal processes with the extremal type of inter-failure time distributions and apply them to predict the MTBF (mean time between faults). Of course, since the time interval to detect successive faults is an i.i.d. random variable under the renewal assumption, this model fails to describe the well-known reliability growth phenomenon [1–5]. Another framework to treat the extremal distributions on the stochastic counting process is developed by Miller [37]. He shows that the stochastic point process of minimal order statistics converges to a minimal extremal-type NHPP with one of the following three mean value functions:

$$M(t) = \begin{cases} t^{\alpha}, & t > 0; \alpha > 0 \quad \text{(Type I)} \\ (-t)^{-\alpha}, & t < 0; \alpha > 0 \quad \text{(Type II)} \\ e^{t}, & -\infty < t < \infty \quad \text{(Type III)}. \end{cases} \tag{9}$$

If the mean value function is given by $M(at + b)$ with arbitrary a (> 0) and b, of course, the process is also a minimal extremal type of NHPP. In other words, the minimal extremal-type NHPP can be characterized by assuming that the probability distribution of the time to first failure is given by one of the extremal distributions, i.e., the mean value function of the NHPP is given by $M(t) = -\log \overline{F}(t)$ and is not bounded, say, $lim_{t\to\infty} M(t) \to \infty$. This process exhibits the reliability growth phenomenon but the initial number of software faults before testing should be infinite.

3.2 Gompertz Software Reliability Model

Here we pay our attention to the Gumbel (Type III) distribution in Eq. (8). If we apply the Gumbel distribution to represent the software fault-detection time distribution, some modifications will be needed since the support of the Gumbel distribution is defined on the real space, i.e., $-\infty < T_{\min} < \infty$. To construct the probability distribution function with positive support, we apply two methods: logarithm and truncation approaches in a fashion similar to [14]. When $\log T$ is regarded as the fault-detection time where T is the Gumbel

distributed random variable, then the resulting probability distribution is the Weibull distribution. On the other hand, when the probability distribution function is truncated at the origin; $F(t) := 1 - (1 - F(t))/(1 - F(0))$, the corresponding probability distribution is called the Gompertz distribution:

$$F(t) = 1 - \exp\left[\frac{\lambda}{\alpha}(1 - e^{\alpha t})\right], \qquad t \geq 0. \tag{10}$$

It is easy to check that the Gompertz distribution is IFR (Increasing Failure Rate) because the failure rate is given by $\lambda \exp(\alpha t)$. In Fig. 3, the probability density function of the Gompertz distribution, $f(t) = dF(t)/dt$, is plotted. The arbitrary moment of this distribution can be derived from its Laplace-Stieltjes transform:

$$F^*(s) = \int_0^\infty e^{-st} dF(t) = (\frac{\lambda e}{\alpha})^{\lambda/\alpha} \Gamma(1 - s/\alpha, \lambda/\alpha), \tag{11}$$

where $\Gamma(\cdot, \cdot)$ denotes the incomplete gamma function.

Substituting Eq. (10) into Eq. (5) yields the NHPP with the mean value function:

$$M(t) = \omega\Big\{1 - \exp[\frac{\lambda}{\alpha}(1 - e^{\alpha t})]\Big\}. \tag{12}$$

We call the above NHPP the Gompertz SRM in this paper. The software reliability for the Gompertz SRM is derived as

$$\begin{aligned} R(x \mid t) &= \exp\left\{-[M(t + x) - M(t)]\right\} \\ &= \exp\left\{\omega e^{\lambda/\alpha}\left[\exp\left(-\frac{\lambda}{\alpha}e^{\alpha(t+x)}\right) - \exp\left(-\frac{\lambda}{\alpha}e^{\alpha t}\right)\right]\right\}. \end{aligned} \tag{13}$$

The difference between the Gompertz SRM and the Goel model [6] depends on how to make the positive support in the Gumbel distribution. As mentioned before, there are two distinct ways to make the positive support. For example, the truncated normal distribution and the lognormal distribution can be derived from the normal distribution. The similar approach can be taken for the Gumbel distribution. When these modified doubly exponential distributions are assumed for the minimal extremal-type of NHPP by Miller [37], then the resulting SRMs are reduced to the familiar Power Law process [38] and the other. In Fig. 4, we illustrate the relationship among the extremal-type NHPP models, and provide a unified modeling framework of the NHPP-based SRMs based on the extreme-value theory.

In the latter argument, we denote the mean value functions for the finite and infinite bug models by $M_1(t) = \omega F(t)$ and $M_2(t) = -\log \overline{F}(t)$, respectively.

4 Parameter Estimation

4.1 Maximum Likelihood Estimation

Suppose that the time interval data on the software fault-detection, $\mathcal{I} = \{(t_1, y_1), (t_2, y_2), \ldots, (t_n, y_n)\}$, are available, where (t_i, y_i) denotes that y_i software faults are detected during the time interval $[t_{i-1}, t_i)$ with $0 < t_1 < t_2 < \cdots < t_n$. Given the set of fault-detection data $\mathcal{I}$, the logarithmic likelihood function (LLF) is given by, for $j = 1, 2$,

$$\mathrm{LLF}(\boldsymbol{\theta}) = \sum_{i=1}^{n} y_i \log \left[M_j(t_i; \boldsymbol{\theta}) - M_j(t_{i-1}; \boldsymbol{\theta}) \right] - M_j(t_n; \boldsymbol{\theta}) - \sum_{i=1}^{n} \log y_i! \quad (14)$$

where $\boldsymbol{\theta}$ is a set of the parameters. On the other hand, suppose that the time domain data on the software fault-detection, $\mathcal{D} = (s_1, \ldots, s_k, t_{obs})$, are available, which consist of the software fault-detection time s_i, $i = 1, \ldots, k$, and the observation point of time, t_{obs}. In this case, the LLF for the NHPP is given by

$$\mathrm{LLF}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \log \left[m_j(s_i; \boldsymbol{\theta}) \right] - M_j(t_{obs}; \boldsymbol{\theta}), \quad (15)$$

where $m_j(t; \boldsymbol{\theta}) = dM_j(t; \boldsymbol{\theta})/dt$.

The maximum likelihood estimate (MLE) is defined as the parameter $\boldsymbol{\theta}$ maximizing the LLF. If the function $LLF(\boldsymbol{\theta})$ is strictly concave in $\boldsymbol{\theta}$ and there exists a unique MLE, it has to satisfy the following first order condition of optimality (likelihood equation):

$$\frac{\partial \mathrm{LLF}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \mathbf{0}. \quad (16)$$

For example, in the Gompertz SRM, the likelihood equation under the interval data can be derived as follows.

$$
\begin{cases}
\sum_{i=1}^{n} \Big\{ y_i \lambda [\exp\{\frac{\lambda}{\alpha} e^{\alpha t_i}\} - \exp\{\frac{\lambda}{\alpha} e^{\alpha t_{i-1}}\} + (1-\alpha t_i)\exp\{\alpha t_i + \frac{\lambda}{\alpha} e^{\alpha t_{i-1}}\} \\
\quad -(1-\alpha t_{i-1})\exp\{\alpha t_{i-1} + \frac{\lambda}{\alpha} e^{\alpha t_i}\}] \Big\} \Big/ \alpha^2 \{\exp(\frac{\lambda}{\alpha} e^{\alpha t_i}) - \exp(\frac{\lambda}{\alpha} e^{\alpha t_{i-1}})\} \\
+ \sum_{i=1}^{n} y_i \lambda e^{\frac{\lambda}{\alpha}} \{1 + e^{\alpha t_n}(\alpha t_n - 1)\} \Big/ \alpha^2 \{e^{\frac{\lambda}{\alpha}} - \exp(\frac{\lambda}{\alpha} e^{\alpha t_n})\} = 0, \\
\\
\sum_{i=1}^{n} y_i [\exp\{\frac{\lambda}{\alpha} e^{\alpha t_i}\} - \exp\{\frac{\lambda}{\alpha} e^{\alpha t_{i-1}}\} \\
\quad + \exp\{\alpha t_i + \frac{\lambda}{\alpha} e^{\alpha t_{i-1}}\} - \exp\{\alpha t_{i-1} + \frac{\lambda}{\alpha} e^{\alpha t_i}\}] \Big/ \\
\quad \alpha \{\exp(\frac{\lambda}{\alpha} e^{\alpha t_i}) - \exp(\frac{\lambda}{\alpha} e^{\alpha t_{i-1}})\} \\
\quad - \sum_{i=1}^{n} y_i e^{\frac{\lambda}{\alpha}} (1 + e^{\alpha t_n}) \Big/ \alpha \{e^{\frac{\lambda}{\alpha}} - \exp(\frac{\lambda}{\alpha} e^{\alpha t_n})\} = 0.
\end{cases}
$$

4.2 EM Algorithm

In general, it is not so easy to solve the simultaneous likelihood equations numerically even if the usual non-linear optimization technique like the Newton's method is applied. Since the model parameters in SRMs have a few implicit constraints such as positive condition, the Newton's method does not always converge to MLEs if the initial values of the algorithm are far from the MLEs. This problem is caused by the local convergence property in the algorithms like the Newton's method. To overcome this problem, Okamura *et al.* [14, 21–23] propose iterative algorithms to estimate the model parameters for many specific SRMs using the EM algorithm.

The EM algorithm is an iterative method for the estimation problem with incomplete data [17, 18]. There are two parts: E-step and M-step. In the E-step, we calculate the expected value of the LLF for complete data, provided that incomplete data is observed. Calculating the expected LLF requires the actual values of model parameters, but provisional estimates of parameters are used in most cases. In the M-step, we find the estimates so as to maximize the expected LLF calculated in the E-step. After finding the values which maximize the expected LLF, the provisional estimates are updated by the parameters. By executing the E-step and the M-step iteratively until the provisional estimates converge to certain points, we get the MLEs for model parameters.

In this paper, we consider the EM algorithm for the Gompertz SRM. When focusing on the fact that the sample of the Gompertz distribution is regarded as a part of the sample of the Gumbel distribution, the EM algorithm for the Gompertz SRM is essentially equivalent to that for the SRM based on the Gumbel distribution. In other words, the following result is applicable to all

the extreme-value types of SRM.

Consider the time domain data on the software fault-detection, $\mathcal{D} = (s_1, \ldots, s_k, t_{obs})$, which follows the Gompertz distribution. Then the fault data $\mathcal{D}$ is obviously incomplete data, when the fault-detection times are sampled from the Gumbel distribution, because it is truncated at the origin and the time t_{obs}. That is, assuming that the truncated fault-detection times are observed, we can obtain the expected LLF based on the Gumbel distribution. Let X_i, $i = 1, \ldots, N$ be all the fault-detection times. The fault-detection time X_i may involve a negative value. If X_i takes a negative value, the corresponding fault-detection time is actually truncated. Under this situation, the sample X_i follows the doubly exponential distribution, so that we have the expected LLF:

$$\begin{aligned}\mathrm{LLF}(\omega, \lambda, \alpha) &= \mathrm{E}[N|\mathcal{D}] \log \omega - \omega + \mathrm{E}[N|\mathcal{D}] \log \lambda \\ &+\alpha \mathrm{E}\left[\sum_{i=1}^{N} X_i \middle| \mathcal{D}\right] - \frac{\lambda}{\alpha} \mathrm{E}\left[\sum_{i=1}^{N} e^{\alpha X_i} \middle| \mathcal{D}\right]. \end{aligned} \tag{17}$$

In Eq. (17), the expected values can be derived by using the following formula (see e.g. [23]):

$$\mathrm{E}\left[\sum_{i=1}^{N} h(X_i) \middle| \mathcal{D}\right] = \sum_{i=1}^{n} h(s_i) + \omega' \left(\int_{-\infty}^{0} h(x) f(x) dx + \int_{t_{obs}}^{\infty} h(x) f(x) dx \right), \tag{18}$$

where $h(t)$ is an arbitrary function and $f(t)$ is the probability density function of the Gumbel distribution.

The usual EM algorithm requires closed form solutions of likelihood equations for the fault-detection time distribution, so that the closed form constructs the update formulae in the M-step. However, in this case, it is not possible to find the explicit form of the parameter α, maximizing the expected LLF. Thus we apply the generalized EM (GEM) algorithm to estimate the parameters [17, 18]. The GEM algorithm does not always require the closed form solutions of likelihood equations, but the corresponding update in the M-step has to increase the expected LLF. Applying the GEM algorithm, we derive the following update formulae to estimate the parameters:

$$\omega := \mathrm{E}[N|\mathcal{D}; \omega', \lambda', \alpha'], \tag{19}$$

$$\lambda := \frac{\alpha' \mathrm{E}[N|\mathcal{D}; \omega', \lambda', \alpha']}{\mathrm{E}\left[\sum_{i=1}^{N} e^{\alpha' X_i} \middle| \mathcal{D}; \omega', \lambda', \alpha'\right]}, \tag{20}$$

$$\begin{aligned}\alpha := &\alpha' \mathrm{E}[N|\mathcal{D}; \omega', \lambda', \alpha'] \\ &\Big/ \mathrm{E}\left[\sum_{i=1}^{N} \left(\alpha' X_i + \log \frac{\lambda'}{\alpha'}\right) \left\{\frac{\lambda'}{\alpha'} e^{\alpha' X_i} - 1\right\} \middle| \mathcal{D}; \omega', \lambda', \alpha'\right]. \end{aligned} \tag{21}$$

Note that Eq. (21) is not the analytical solution of the parameter α which maximizes the expected LLF.

Finally, Eqs. (18)– (21) yield the estimation algorithm for the Gompertz SRM. However, in the above algorithm, the parameter ω involves the number of the truncated fault-detection time with negative value. Thus the estimate ω should be modified as $\omega\{1 - \exp(-\lambda/\alpha)\}$ in order to calculate the actual parameter in the Gompertz SRM.

5 Numerical Illustrations

5.1 Performance Test of EM Algorithm

We investigate the effectiveness of the EM algorithm in the Gompertz SRM by comparing them with BFGS quasi-Newton algorithm [39]. The BFGS quasi-Newton method can be used when the Hessian matrix cannot be written in the simple form. Instead of obtaining an estimate of the Hessian matrix at a single point, this method gradually builds up an approximate Hessian matrix by using gradient information. We examine the global/local convergence properties, and further investigate the convergence speed of algorithms with arbitrary initial values. To do these, we generate two sets of fault-detection time data by pseudo-random numbers. The number of software faults is 100 and the detection times are distributed by the Gompertz distribution with parameters $(\alpha, \lambda) = (1, 1)$ or $(\alpha, \lambda) = (0.1, 1)$. The EM algorithm and the BFGS quasi-Newton algorithm are executed to estimate the parameters in the Gompertz SRM.

To examine the global/local convergence properties and the convergence speed of algorithms quantitatively, both EM and BFGS quasi-Newton algorithms are executed 10 and 100 times. In these experiments, the initial values are selected as 1000 pairs for each data set, where 1000 pairs of values are given by some points near to the MLEs. If the estimates take negative values before 10 or 100 times iteration, stop the algorithm. We use three criteria for evaluation; rate of convergence, mean and variance of relative distance to MLEs, which are abbreviated to ROC, MRD and VRD, respectively. Note that ROC indicates the global/local convergence properties, and that MRD and VRD show the convergence speed and stability of algorithms. They are defined as follows.

$$\mathrm{ROC} = \frac{\begin{pmatrix}\text{number of estimates}\\ \text{with positive values}\end{pmatrix}}{(\text{total number of initial values})} \times 100(\%), \tag{22}$$

$$\mathrm{MRD} = \frac{\sum \left| \dfrac{(\text{MLEs}) - (\text{estimates})}{(\text{MLEs})} \right|}{\begin{pmatrix}\text{number of estimates}\\ \text{with positive values}\end{pmatrix}}, \tag{23}$$

$$\mathrm{VRD} = \frac{\sum \left| \dfrac{(\text{MLEs}) - (\text{estimates})}{(\text{MLEs})} \right|^2}{\begin{pmatrix}\text{number of estimates}\\ \text{with positive values}\end{pmatrix}} - \mathrm{MRD}^2. \tag{24}$$

It can be recognized that ROC is larger as the algorithm has higher convergence ability. Since we perform each algorithm with the fixed number of iterations, MRD represents the convergence speed, namely, MRD is smaller as the resulting estimates are closer to the MLEs under the fixed number of iterations and the same initial value. Moreover, VRD is smaller as the algorithm is more stable. The stable algorithm can provide the exact MLEs for arbitrary initial values.

Tables 1 and 2 present ROC, MRD and VRD for the Gompertz SRMs. In these tables, it is evident that the EM algorithm (EM) is superior to the quasi-Newton method (QN) in terms of the convergence property, because ROCs in the EM algorithm are always 100% in both cases. On contrary, ROCs for the quasi-Newton method is relatively low. This means that the quasi-Newton method is sensitive to the initial values of parameters. Thus the initial values in the quasi-Newton method must be carefully selected.

Next, we focus on the convergence speed and stability of algorithms. In general, it is known that the convergence speed of the EM algorithms is slower than those of the other methods. From Tables 1 and 2, there is no remarkable difference between the EM algorithm and the quasi-Newton method. In the case of $\alpha = 1$ and $\lambda = 1$, the EM algorithm is superior to the quasi-Newton method in terms of MRD. Furthermore, the VRD for the parameter ω in the quasi-Newton method is larger than that in the EM algorithm. This fact implies that it is difficult to estimate the parameter ω accurately by using the quasi-Newton method. In the case of $\alpha = 0.1$ and $\lambda = 1$, the VRD for the parameter ω in the quasi-Newton method is relatively large, but the VRDs in the EM algorithm are small. Thus the careful selection of initial values is

not needed in the EM algorithm, so that we can estimate the parameters with sufficient accuracy by using the EM algorithm without spending much effort.

5.2 Goodness-of-Fit Test

We apply two information criteria: Akaike information criterion (AIC) [40] and Bayesian information criterion (BIC) by Schwartz [41], and the common mean squared error (MSE) to measure the goodness-of-fit to the actual software fault data. From the page limitation, we analyze only the case where the time interval data are available. The criteria for goodness-of-fit test are defined by

$$\begin{aligned}
\mathrm{AIC} &= -2\log\left[LLF(\boldsymbol{\theta}|\mathcal{I})\right] + 2\phi \\
\mathrm{BIC} &= -2\log\left[LLF(\boldsymbol{\theta}|\mathcal{I})\right] + \phi\log\psi \\
\mathrm{MSE} &= \frac{\sum_{i=1}^{n}\left(y_i - M_j(t_i)\right)^2}{n}, \quad j = 1, 2,
\end{aligned}$$

where ϕ is the degree of freedom for the SRM, *i.e.* the number of free parameters, and ψ is the number of data used in the analysis. Four data sets (DS-1 $\sim$ DS-4) are used, where each data includes 81, 32, 62 or 41 failure-count data [1, 3] [1].

We compare the Gompertz SRM with the existing NHPP-based SRMs. The competitive models are given by

$$\begin{aligned}
&\text{Exponential [6]:} \quad M_1(t) = \omega\left\{1 - e^{-\beta t}\right\}, \\
&\text{Gamma [7]:} \quad M_1(t) = \omega\left\{1 - (1+\beta t)e^{-\beta t}\right\}, \\
&\text{Rayleigh [8]:} \quad M_1(t) = \omega\left\{1 - e^{-\beta t^2}\right\}, \\
&\text{Power Law [38]:} \quad M_2(t) = \frac{\lambda}{\alpha}t^{\alpha}, \\
&\text{Weibull [8]:} \quad M_1(t) = \omega\Big\{1 - \exp[-\frac{\lambda}{\alpha}t^{\alpha}]\Big\}.
\end{aligned}$$

In Figs. 5 $\sim$ 8, we plot the number of detected software faults and the mean value functions of six SRMs. In these figures, Gom, Exp, Gam, Ray, Pow and Wei denote Gompertz SRM in this paper, Exponential SRM [6], Gamma (delayed S-shaped) SRM [7], Rayleigh SRM [8], Power Law SRM [38] and Weibull (generalized exponential) SRM [8], respectively. Here in Fig. 7, we

[1] We pick the following data set from [1]; DS-1: DataSet 3 in Chapter 11, DS-2: DataSet 4 in Chapter 11, DS-3: Data Set J1 in Chapter 7 and DS-4: Data Set J2 in Chapter 7.

omit the mean value function of Exp, since it is identical to the behavior of Wei. From these figures, it is observed that the Gompertz SRM is relatively better fitted to the actual data even if the trends of data are different from each other.

Next, we calculate AIC, BIC and MSE for six SRMs with four data sets. It is noted that the smaller value of each criterion denotes the better performance in terms of the goodness-of-fit. Table 3 $\sim$ Table 6 present the goodness-of-fit results for DS-1 $\sim$ DS-4. In Table 3, it is found that the Weibull SRM is the best model on three test criteria. In Tables 4 and 6, one sees that the Gamma SRM provides the best performance for all cases. In Table 5, the Gompertz SRM can provide the best performance in terms of AIC and MSE, but the Power Law gives the smallest BIC. Throughout all cases, the Gompertz SRM was the best SRM in only the cases of AIC and MSE with DS-3. Nevertheless, the most attractive point of this model is that the Gompertz SRM is not totally inferior to the other representative SRMs. In general, the performance of SRMs strongly depends on the kind of data. For instance, even if the Exponential SRM is best fitted to a data set, it may be the worst model for the other data set. Hence, when the software reliability is assessed with the SRMs, one needs to select the representative models and to try them in parallel. In that sense, the Gompertz SRM should be considered as one of the typical SRMs from the empirical conclusions as well as the theoretical view points.

Finally, we estimate the software reliability given in Figs. 9 $\sim$ 12. As mentioned in Eq. (13), the software reliability is defined as the probability that the software system does not fail during the time period $[t, t + x)$, provided that it is operative at time t. From these results, it is observed that Rayleigh SRM and Power Law SRM give the upper and lower bounds of software reliability for six SRMs, respectively. Since the model with smaller information criterion is regarded as the better SRM, it can be expected that the software reliability based on the best SRM is most reliable. In Fig. 9, the software reliabilities evaluated by Gamma and Gompertz SRMs are larger than that evaluated by the Weibull SRM. On the other hand, in Figs. 10 and 12, Gompertz SRM underestimates the software reliability to Gamma SRM, but the difference is not so significant in the range greater than $R(x \mid t) = 0.95$.

6 Conclusions

In this paper, we have tried to propose a unified modeling framework of the SRMs based on the extreme-value theory and developed the Gompertz SRM as one of the representative SRMs. As an effective parameter estimation algorithm, we have developed the EM algorithm to calculate the maximum likelihood estimates of model parameters involved in the Gompertz SRM. In

numerical examples, we have checked the convergence properties of the EM algorithm and performed the goodness-of-fit test to the actual software fault-detection data. From some numerical observations, it can be concluded that the proposed Gompertz SRM is rather attractive comparing with the existing SRMs, and should be treated as one of the representative NHPP-based models.

We believe that the Gompertz SRM developed in this paper could answer the questions from practitioners faced in the software quality management; what can we do with the classical trend-curve modeling approach. In future, we will examine the usefulness for the Gompertz SRM in software testing practice. In the Japanese software development companies, the estimation results on software reliability for several actual projects should be compared based on the classical deterministic approach and the stochastic one proposed here.

References

[1] M. R. Lyu (Ed.), Handbook of Software Reliability Engineering, McGraw-Hill, New York, 1996.

[2] J. D. Musa, Software Reliability Engineering, McGraw-Hill, New York, 1999.

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

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

[5] M. Xie, Software Reliability Modelling, World Scientific, Singapore, 1991.

[6] A. L. Goel, K. Okumoto, Time-dependent error-detection rate model for software reliability and other performance measures, IEEE Trans. Reliab. R-28 (1979) 206–211.

[7] S. Yamada, M. Ohba, S. Osaki, S-shaped reliability growth modeling for software error detection, IEEE Trans. Reliab. R-32 (1983) 475–478.

[8] A. L. Goel, Software reliability models: Assumptions, limitations and applicability, IEEE Trans. Software Eng. SE-11 (1985) 1411–1423.

[9] K. Sakata, Formulation for predictive methods in software production control – static prediction and failure rate transition model – (in Japanese), Trans. on IECE of Japan 57-D (1974) 277–283.

[10] M. Ohba, Inflection S-shaped software reliability growth model, in: S. Osaki, Y. Hatoyama (Eds.), Stochastic Models in Reliability Theory, Springer-Varlag, Berlin, 1984, pp. 144–165.

[11] N. Langberg, N. D. Singpurwalla, Unification of some software reliability models, SIAM J. Sci. Comput. 6 (3) (1985) 781–790.

[12] D. R. Miller, Exponential order statistic models of software reliability growth, IEEE Trans. Software Eng. SE-12 (1986) 12–24.

[13] S. S. Gokhale, K. S. Trivedi, Log-logistic software reliability growth model, in: Proc. 3rd IEEE Int'l. High-Assurance Systems Eng. Sympo. (HASE-1998), IEEE CS Press, 1998, pp. 34–41.

[14] H. Okamura, T. Dohi, S. Osaki, EM algorithms for logistic software reliability models, in: Proc. 7th IASTED Int'l Conf. on Software Eng., 2004, pp. 263–268.

[15] B. Gompertz, On the nature of the function expressive of the law of human mortality, and on a new mode of determining the value of life contingencies, Phil. Trans. Roy. Soc. London 123 (1832) 513–585.

[16] E. P. Virene, Reliability growth and its upper limit, in: Proc. 1968 Annual Sympo. on Reliab., 1968, pp. 265–270.

[17] A. P. Dempster, N. M. Laird, D. B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, J. Roy. Statist. Soc. B B-39 (1977) 1–38.

[18] C. F. J. Wu, On the convergence properties of the EM algorithm, Ann. Statist. 11 (1983) 95–103, convergence for EM algorithm.

[19] S. A. Hossain, R. C. Dahiya, Estimating the parameters of a non-homogeneous Poisson-process model for software reliability, IEEE Trans. Reliab. 42 (1993) 604–612.

[20] G. Knafl, J. Morgan, Solving ML equations for 2-parameter Poisson-process model for ungrouped software failure data, IEEE Trans. Reliab. 45 (1996) 42–53.

[21] H. Okamura, A. Murayama, T. Dohi, EM algorithm for discrete software reliability models: a unified parameter estimation method, in: Proc. 8th IEEE Int'l Sympo. on High Assurance Systems Eng. (HASE-2004), IEEE CS Press, 2004, pp. 219–228.

[22] H. Okamura, Y. Watanabe, T. Dohi, Estimating mixed software reliability models based on the EM algorithms, in: Proc. of 2002 Int'l Sympo. on Empirical Software Eng. (ISESE-2002), IEEE CS Press, 2002, pp. 69–78.

[23] H. Okamura, Y. Watanabe, T. Dohi, An iterative scheme for maximum likelihood estimation in software reliability modeling, in: Proc. of 14th Int'l Sympo. on Software Reliab. Eng. (ISSRE-2003), IEEE CS Press, 2003, pp. 246–256.

[24] D. Satoh, A discrete gompertz equation and a software reliability growth model, IEICE Trans. on Information and Systems (D) E83 (2000) 1508–1513.

[25] D. Satoh, S. Yamada, Discrete equations and software reliability growth models, in: Proc. 12th Int'l Sympo. on Software Reliab. Eng., 2001, pp. 176–184.

[26] S. Yamada, A stochastic software reliability growth model with gompertz curve (in Japanese), Trans. IPSJ 33 (1992) 964–969.

[27] K. Shima, S. Takada, K. Matsumoto, K. Torii, A study on the failure intensity of different software faults, in: Proc. 19th Int'l Conf. on Software Eng., 1997, pp. 86–94.

[28] M. Ohba, Software reliability analysis, IBM. J. Research & Development 28 (1984) 428–443.

[29] Z. Jelinski, P. B. Moranda, Software reliability research, in: W. Freiberger (Ed.), Statistical Computer Performance Evaluation, Academic Press, New York, 1972, pp. 465–484.

[30] M. Zhao, M. Xie, On maximum likelihood estimation for a general non-homogeneous Poisson process, Scand. J. Statist. 23 (1996) 597–607.

[31] B. V. Gnedenko, Sur la distribution limitee du terme maximum d'une serie aleatoire, Ann. Math. 44 (1943) 423–453.

[32] E. J. Gumbel, Statistics of Extremes, Columbia Press, New York, 1958.

[33] E. Castillo, Extreme Value Theory in Engineering, Academic Press, San Diego, 1988.

[34] J. Galambos, The Asymptotic Theory of Extreme Order Statistics, Robert E. Krieger Publishing Company, Florida, 1987.

[35] L. M. Kaufman, J. B. Dugan, B. W. Johnson, Using statistics of the extremes for software reliability analysis of safety critical systems, in: Proc. 9th Int'l Sympo. Software Reliab. Eng., 1998, pp. 355–363.

[36] L. M. Kaufman, J. B. Dugan, B. W. Johnson, Using statistics of the extremes for software reliability analysis, IEEE Trans. Reliab. R-48 (1999) 292–299.

[37] D. R. Miller, Order statistics, poisson processes and repairable systems, J. Appl. Probab. 13 (1976) 519–529.

[38] E. Cretois, O. Gaudoin, New results on goodness-of-fit tests for the power-law process and application to software reliability, Int'l J. Reliab. Qual. Safe. Eng. 5 (1998) 249–267.

[39] R. Fletcher, A new approach to variable metric algorithms, The Computer Journal 13 (1970) 317–322.

[40] H. Akaike, Information theory and an extension of the maximum likelihood principle, in: B. N. Petrov, F. Csaki (Eds.), Proc. 2nd Int'l Sympo. on Information Theory, Akademiai Kiado, 1973, pp. 267–281.

[41] G. Schwarz, Estimating the dimension of a model, Ann. Statist. 6 (1978) 461–464.

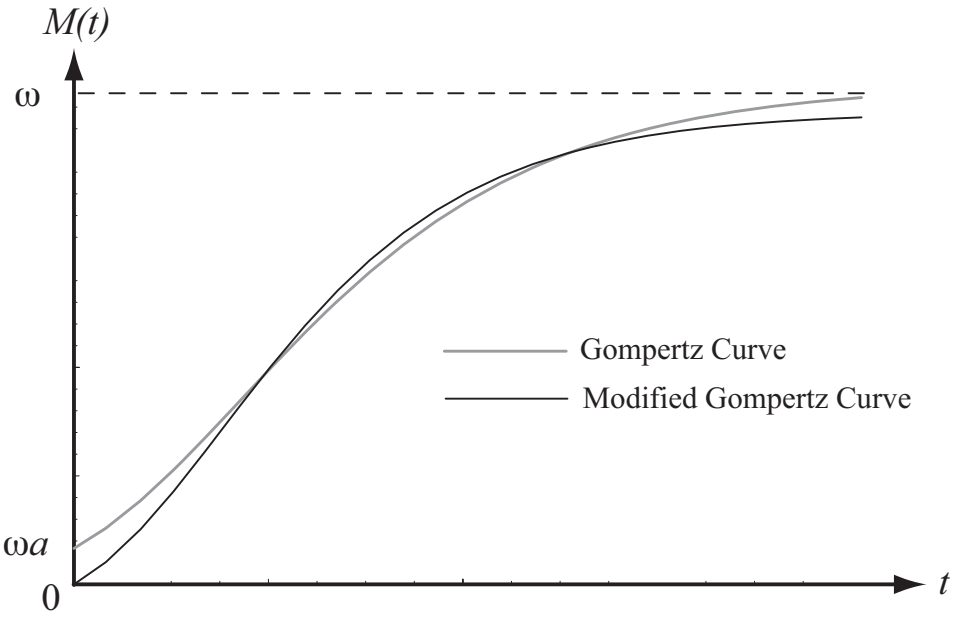

Fig. 1. Schematic illustration of the Gompertz curves.

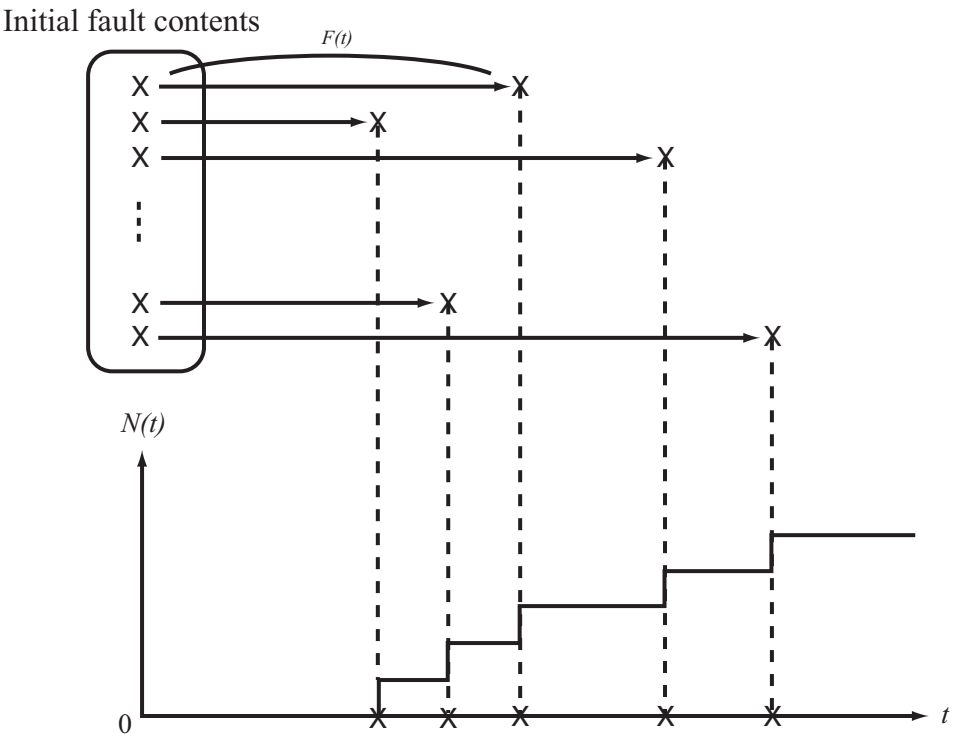

Fig. 2. Configuration of software debugging theory.

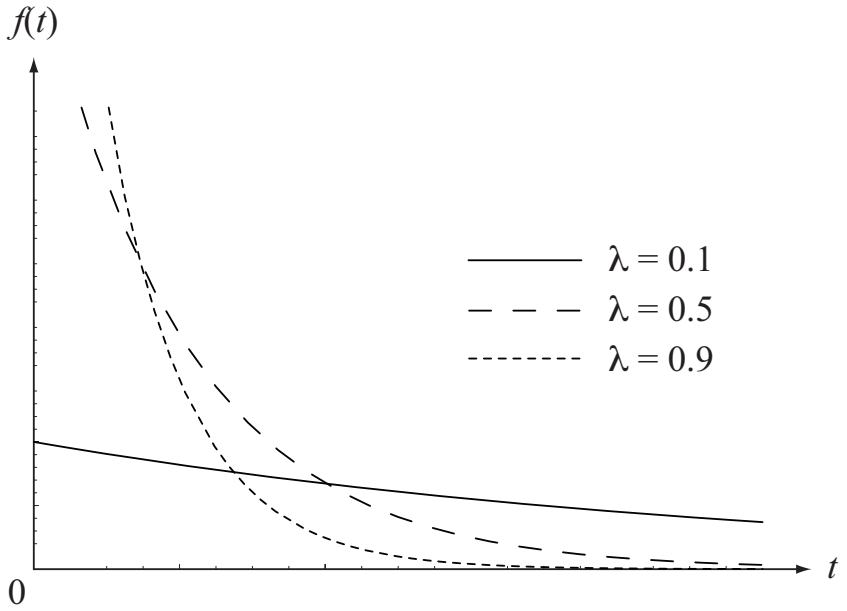

Fig. 3. Gompertz distribution.

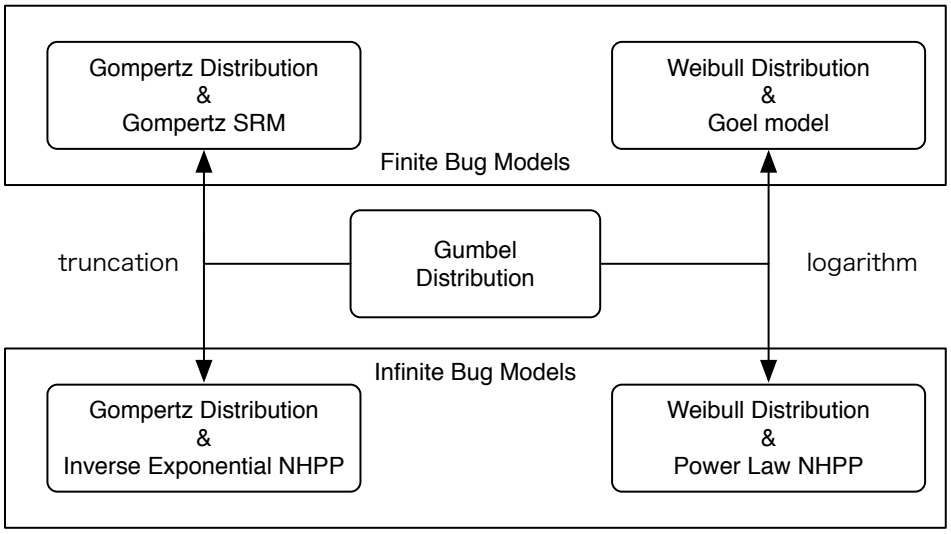

Fig. 4. Relationship among the extremal-type NHPP models.

Table 1
Comparison between EM algorithm and quasi-Newton method (Case 1: $\alpha = 1$ and $\lambda = 1$).

method	parameter	ROC	MRD	VRD
EM(10)	ω	100.0%	0.125	1.14E-01
	α	100.0%	0.285	3.50E-02
	λ	100.0%	0.197	4.25E-02
EM(100)	ω	100.0%	0.005	4.26E-06
	α	100.0%	0.100	1.55E-03
	λ	100.0%	0.048	3.39E-04
QN(10)	ω	84.0%	14.319	4.95E+04
	α	84.0%	0.208	1.38E-01
	λ	84.0%	0.606	1.78E+02
QN(100)	ω	78.5%	6.687	5.58E+03
	α	78.5%	0.175	1.32E-01
	λ	78.5%	0.114	7.86E-02

Table 2
Comparison between EM algorithm and quasi-Newton method (Case 2: $\alpha = 0.1$ and $\lambda = 1$).

method	parameter	ROC	MRD	VRD
EM(10)	ω	100.0%	0.444	4.01E-01
	α	100.0%	0.354	8.27E-02
	λ	100.0%	0.327	1.35E-01
EM(100)	ω	100.0%	0.131	7.79E-01
	α	100.0%	0.186	1.13E-01
	λ	100.0%	0.226	1.11E-01
QN(10)	ω	98.1%	0.164	2.46E-01
	α	98.1%	0.094	5.56E-02
	λ	98.1%	0.041	1.44E-02
QN(100)	ω	98.0%	0.248	1.91E+01
	α	98.0%	0.011	7.59E-03
	λ	98.0%	0.007	5.04E-03

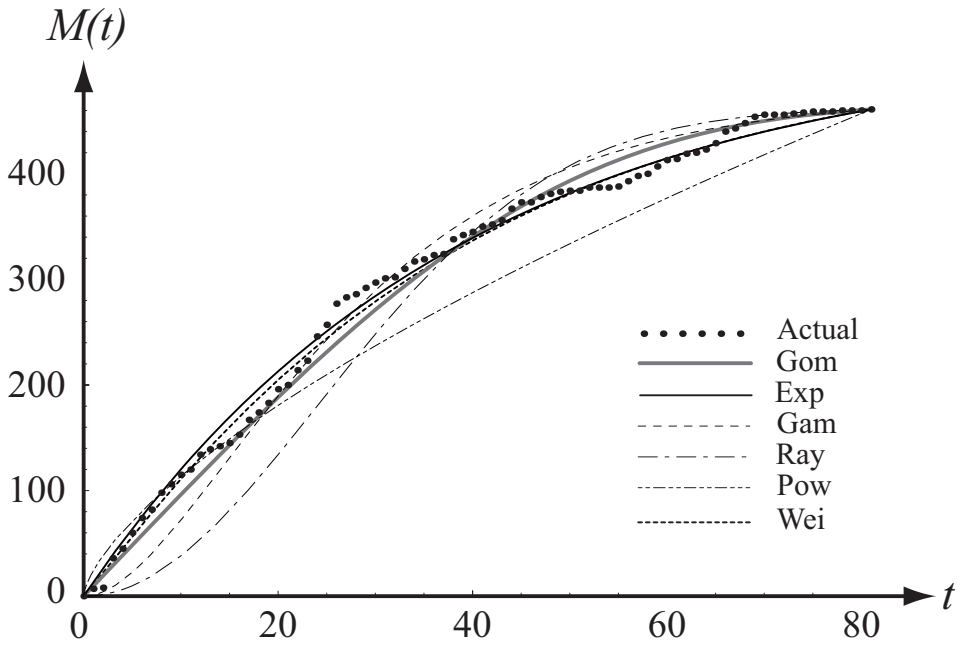

Fig. 5. Behavior of the mean value functions of the NHPP-based SRMs (DS-1).

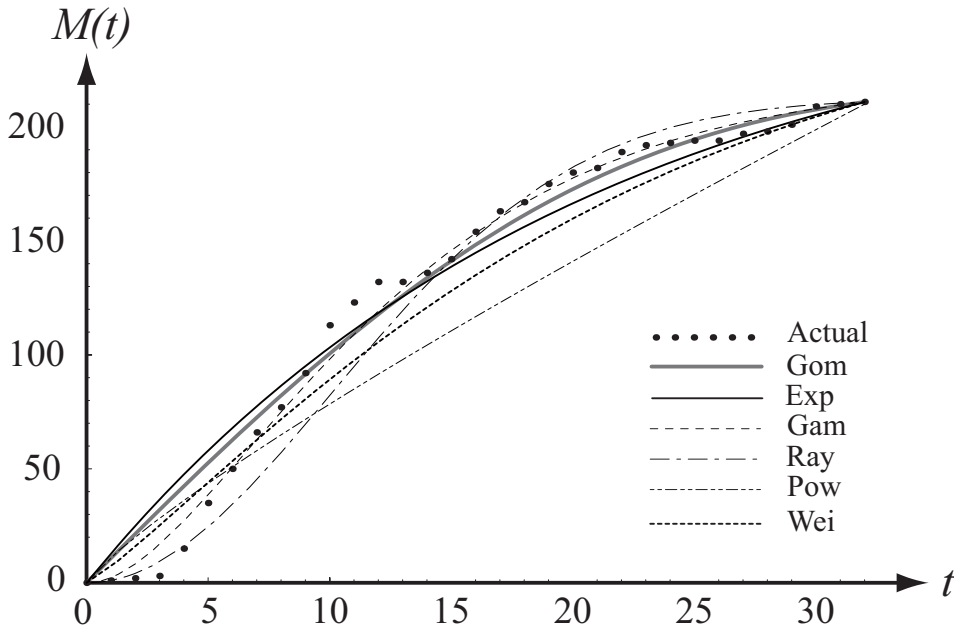

Fig. 6. Behavior of the mean value functions of the NHPP-based SRMs (DS-2).

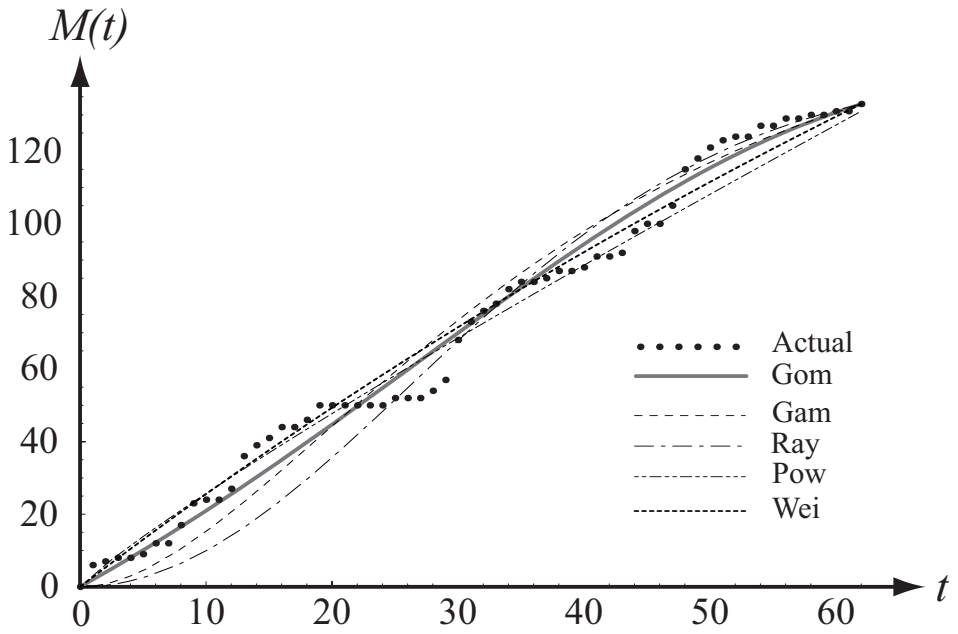

Fig. 7. Behavior of the mean value functions of the NHPP-based SRMs (DS-3).

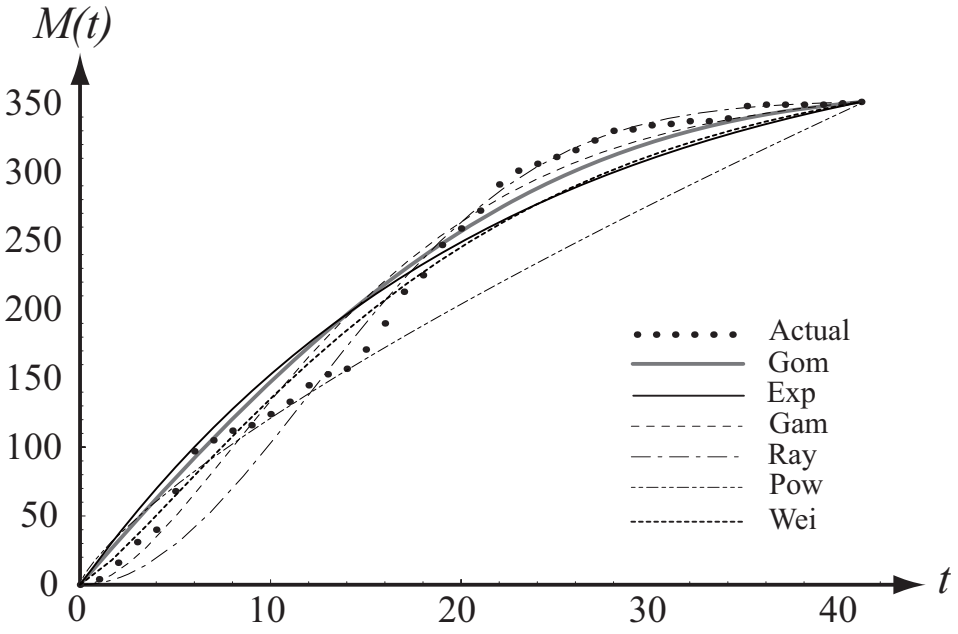

Fig. 8. Behavior of the mean value functions of the NHPP-based SRMs (DS-4).

Table 3
Goodness-of-fit test result (DS-1).

model	AIC	BIC	MSE
Gom	505.53	512.72	205.55
Exp	500.19	504.97	114.31
Gam	555.84	560.63	416.39
Ray	689.53	694.32	1676.30
Pow	569.61	574.42	1533.35
Wei	496.66	501.48	101.85

Table 4
Goodness-of-fit test result (DS-2).

model	AIC	BIC	MSE
Gom	231.23	235.62	112.41
Exp	241.23	244.16	177.16
Gam	193.24	196.17	37.54
Ray	211.52	214.45	150.89
Pow	275.73	278.72	720.43
Wei	227.34	230.33	226.94

Table 5
Goodness-of-fit test result (DS-3).

model	AIC	BIC	MSE
Gom	287.74	294.12	25.62
Exp	288.14	292.40	27.51
Gam	312.19	316.45	52.17
Ray	326.45	330.70	83.30
Pow	288.09	292.38	34.59
Wei	288.16	292.44	28.76

Table 6
Goodness-of-fit test result (DS-4).

model	AIC	BIC	MSE
Gom	297.04	302.18	285.96
Exp	325.50	328.93	488.52
Gam	291.44	294.87	232.98
Ray	314.62	318.05	314.11
Pow	390.60	394.07	1623.39
Wei	297.80	301.27	285.96

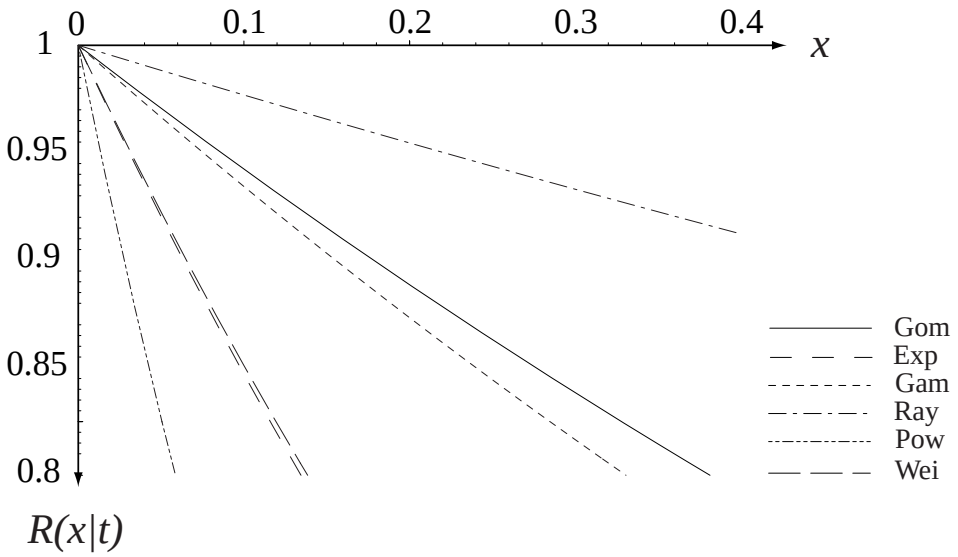

Fig. 9. Behavior of the software reliability (DS-1).

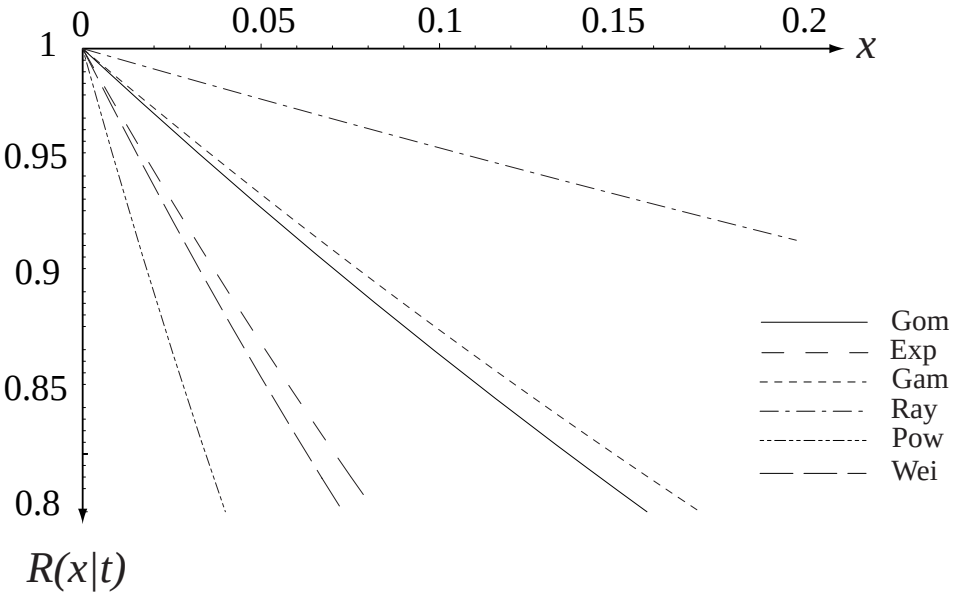

Fig. 10. Behavior of the software reliability (DS-2).

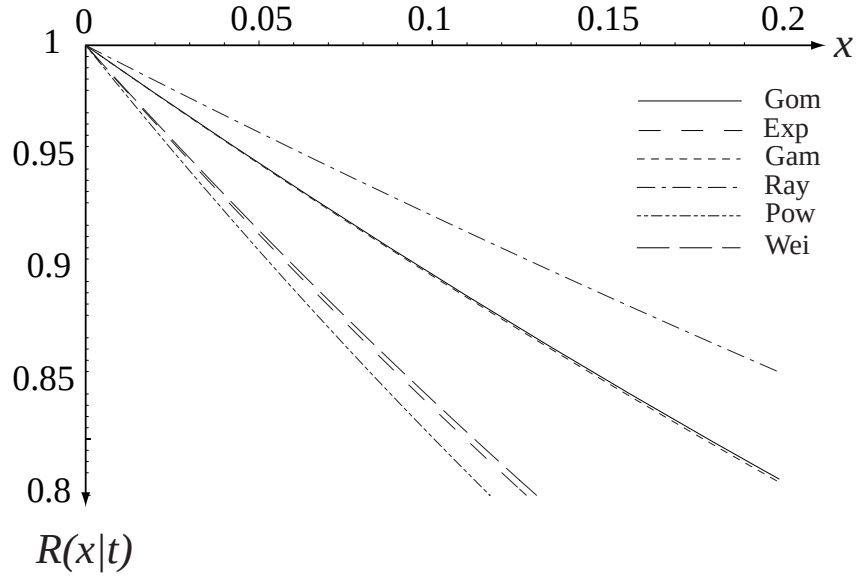

Fig. 11. Behavior of the software reliability (DS-3).

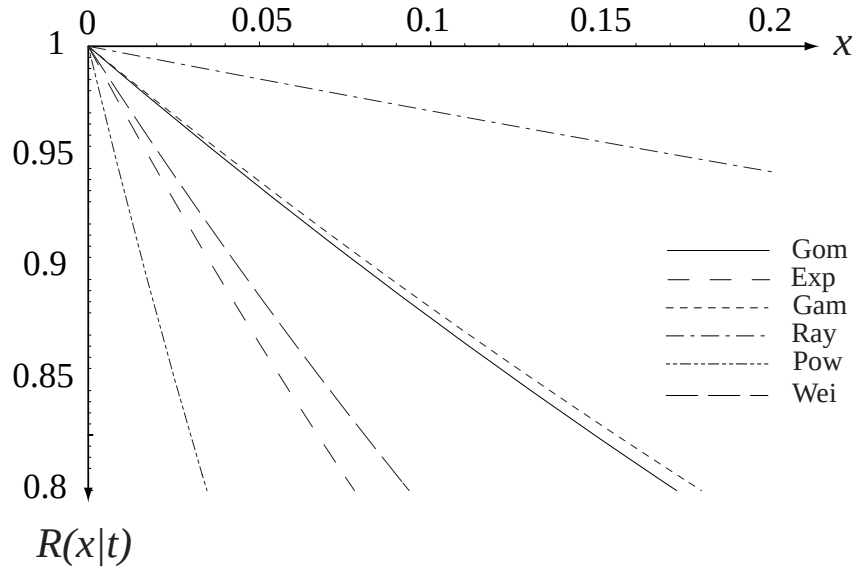

Fig. 12. Behavior of the software reliability (DS-4).