Software reliability analysis models

by Mitsuru Ohba

This paper discusses improvements to conventional software reliability analysis models by making the assumptions on which they are based more realistic. In an actual project environment, sometimes no more information is available than reliability data obtained from a test report. The models described here are designed to resolve the problems caused by this constraint on the availability of reliability data. By utilizing the technical knowledge about a program, a test, and test data, we can select an appropriate software reliability analysis model for accurate quality assessment. The delayed S-shaped growth model, the inflection S-shaped model, and the hyperexponential model are proposed.

1. Introduction

Quality control is one of the key engineering technologies in today's industry. Some hardware oriented reliability theories and models, such as the exponential and the Weibull distribution models, have contributed to the high reliability of present-day electric parts and consumer products. Similarly, in the world of computer software, comprehensive software reliability theories and models can be expected to contribute towards achieving software quality objectives.

The first well-known proposal for software reliability analysis was made by Jelinski and Moranda [1]. Their model allows us to estimate the following reliability measures based on observing the results of a test, i.e., from the error detection (discovery) process. One measure is the number of errors that were initially present in a tested program. Another is the error detection rate, which indicates the efficiency of testing. From a mathematical point of view, many of the later models [e.g., the Musa execution time model and the Goel-Okumoto nonhomogeneous Poisson process (NHPP) model] are identical to the Jelinski-Moranda model. The models discussed in this paper are in the category of reliability growth models. The simplest form of such a model is the exponential model shown in **Figure 1**. As discussed in this paper, software reliability growth is defined by the mathematical relationship that exists between the time span of using (or testing) a program and the cumulative number of errors discovered.

In contrast to the above assumed exponential growth in software reliability, S-shaped software reliability growth is more often observed in real projects [2–5]. Curve (b) of Fig. 1 is a typical S-shaped reliability growth curve. In Japan, the Gompertz model and the logistic curve are used to represent S-shaped software reliability growth. Although it is quite practical to use the Gompertz model and the logistic curve, it is sometimes dangerous since these models may lead to a more optimistic assessment than other models [6, 7]. No interpretations of physical meaning are implied in the parameters of the Gompertz model. Although the parameters of the logistic model have some interpretations, these interpretations are not realistic in actual projects.

There are many reasons why observed software reliability growth curves often become S-shaped. As described in this paper, the S-shaped software reliability growth curve is typically caused by the definition of errors (i.e., failures or faults); more specifically, the problem is under what conditions test personnel decide that they have detected an error. The growth is also caused by the continuous test effort increase in which the test effort has been incrementally

increased through the test period. Some of these causative factors or influences can be described by making the basic assumptions of the exponential growth model more realistic. The delayed S-shaped software reliability growth model and the inflection S-shaped software reliability growth model described later have been developed incorporating the basic assumptions of the exponential software reliability growth model with new, more realistic assumptions.

In addition to the S-shaped software reliability growth curve, on rare occasions we also observe in real projects the hyperexponential reliability growth curve shown as curve (c) of Fig. 1. Based on our analysis of one of the IBM software error data sets, hyperexponential software reliability growth may be observed when the following conditions are satisfied: 1) The program under test consists of two or more modules which have different characteristics (i.e., different complexity, written in a different language, different error ratio, etc.); 2) the modules have been well tested prior to their integration, and not too many errors (faults) remain in the program; and 3) the software system is fairly large. Theoretically, if error detection rates of modules are different, the integrated reliability growth of those independent modules will become rapid linear growth with eventual saturation beyond a certain point. In such cases, it is not reasonable to apply the ordinary exponential growth model. It yields a quite pessimistic assessment in comparison with actual results because of the linear growth and the rapid saturation.

Therefore, the ordinary exponential growth model, the Gompertz model, and the logistic (curve) model are sometimes insufficient and inaccurate to analyze actual project error data for software quality assessment. In this paper, we propose three different software reliability growth models: the delayed S-shaped growth model, the inflection S-shaped growth model, and the hyperexponential growth model. Moreover, several examples of applying these models for analyzing actual project error data are given. Finally, the applicability for actual software reliability assessment of these models along with the ordinary exponential growth model is discussed.

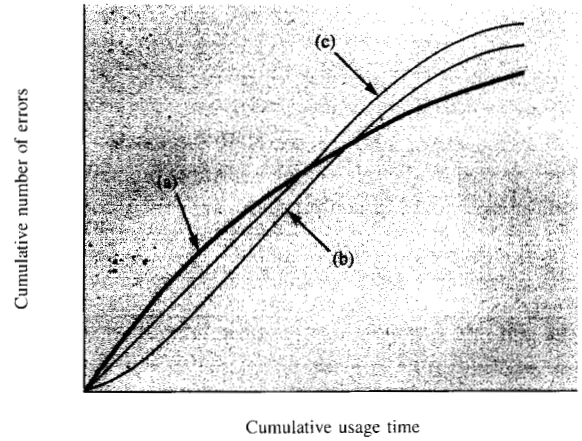

Figure 1

Software reliability growth curves. (a) Exponential software reliability growth; (b) typical S-shaped reliability growth; and (c) hyperexponential reliability growth.

2. Exponential growth models

A fault, in this paper, is defined as a cause of a failure. A fault is an erroneous statement or statements in a program which cause one or more failures. A failure is defined as an erroneous result or the malfunction of a program. During a test, we run test cases, observe results of runs, and eventually detect failures. After failure detection, failure analysis or fault isolation is performed. As a result of fault isolation, we find a fault and define a fix for the fault. The exponential software reliability growth models are designed to describe the failure detection process.

The models proposed by Musa [8], by Littlewood and Verrall [9], and by Goel and Okumoto [10] can be regarded as variants of the exponential growth models. In particular, the Musa execution time model and the Goel-Okumoto NHPP model are mathematically isomorphic. The Littlewood-Verrall model is another interpretation of the Jelinski-Moranda model based on a Bayesian approach.

The exponential growth models are based on the following assumptions:

1. All faults in a program are mutually independent from the failure detection point of view.
2. The number of failures detected at any time is proportional to the current number of faults in a program.
3. The probability of failure detection (proportionality) is constant.
4. The isolated faults are removed prior to future test occasions.

Musa's model has made a unique contribution to our understanding of the relationship between execution time and calendar time. The model allows us to predict a future reliability figure of a program based on the failure detection process of a test. The model is characterized by the following function:

$$m(t) = N\{1 - \exp(-Ct/MT)\}, \quad (1)$$

where t is execution time, that is, the total CPU time utilized to complete the test case runs up to a time of observation, C is the testing compression factor, N is the number of failures in the program, M is the total number of failures possible

during the maintained life of the program, T is the mean time to failure (MTTF) at the beginning of the test, and $m(t)$ is the number of failures discovered as a result of test case runs up to the time of observation.

Similarly, the Goel-Okumoto NHPP model is characterized by the following mean value function of a nonhomogeneous Poisson process:

$$m(t) = N\{1 - \exp(-\phi t)\}, \tag{2}$$

where t is the time of observation, ϕ is the failure detection rate, and $m(t)$ is the number of failures detected up to the time of observation. Obviously Eqs. (1) and (2) are isomorphic. The advantage of this model is a simple and compact algorithm of the parameter estimation method (see Appendix A). The exponential growth model gives a good estimate of the parameter N if the assumptions described above are satisfied (see Example 5 in Section 6).

The assumption of mutual independency of faults is equivalent to assuming that all faults in a program are randomly captured (failures occur randomly). Actually, faults are mutually dependent because of logical or functional dependencies that exist within a program (lattice structure). This mutual dependency of faults makes the observed software reliability growth curve S-shaped, because the number of detectable faults increases as the number of detected faults increases. During the early phase of a test, the growth is slow. The more faults are removed, the more (dependent) faults become detectable. Then the growth gradually goes up while the number of undetected faults which are detectable increases. The growth becomes slow again beyond this point, because the number of detectable faults gradually decreases. Thus, the growth of this failure detection process becomes S-shaped.

In addition, the proportionality of capturing the undetected failures changes from time to time during the test period in real projects. For example, when one observes a failure detection process based on calendar time, the test effort (i.e., the amount of work required for performing the test) is not homogeneously distributed. Thus, the observed software reliability growth may become S-shaped (if there is a peak of effort in the latter half of the test period, as in Example 2 in Section 6).

Moreover, if there were two or more different sources of failures and if they had different detection rates, the observed software reliability growth curve might become hyperexponential, as described in Section 5.

3. Delayed S-shaped growth model

The delayed S-shaped software reliability growth model has been developed for describing a software error removal phenomenon and its physical interpretation using a model that has a time delay function in the sense of control theory. Using this model, we can analyze a test process as a fault isolation process, not only a failure detection process. Fault isolation means that certain failures can be intentionally reproduced, leading to the identification of the fault and its removal. Sometimes error data of a test report are not failure detection data, but the fault isolation in an actual project (see Example 1 in Section 6). The model is designed for analysis of the fault isolation data.

The delayed S-shaped growth model is based on the following assumptions:

1. All faults in a program are mutually independent from the failure detection point of view.
2. The probability of failure detection at any time is proportional to the current number of faults in a program.
3. The proportionality of failure detection is constant.
4. The probability of fault isolation at any time is proportional to the current number of faults not isolated.
5. The proportionality of fault isolation is constant.
6. The detected faults can be entirely removed.

The model is characterized by the following delayed S-shaped reliability growth mean value function $g(t)$ of NHPP:

$$g(t) = N\{1 - (1 + \rho t)\exp(-\rho t)\}, \tag{3}$$

where t is time of observation, ρ is the fault removal (failure detection and fault isolation) rate parameter, and $g(t)$ is the number of faults isolated up to time t. The mean value function of Eq. (3) gives an S-shaped reliability growth curve.

The physical meaning of the fault isolation rate ρ is as follows. Let $F(\tau_i)$ be the p.d.f. (probability density function) of failure detection between observation of the $(i-1)$th and the ith failure detection. As with the exponential growth model, $F(\tau_i)$ is given by

$$F(\tau_i) = \phi\{N - (i-1)\}, \qquad 0 \le \tau_i < t_i - t_{i-1}, \tag{4}$$

where ϕ is the failure detection rate and t_i is the time when the ith failure is detected. Independently of the failure detection rate, suppose $G(\tau_j')$ is the p.d.f. of fault isolation between isolation of the $(j-1)$th and the jth fault. $G(\tau_j')$ is given by

$$G(\tau_j') = \lambda\{i - (j-1)\}, \qquad 0 \le \tau_j' < \min\{t_{i+1}, t_j' - t_{j-1}'\}, \tag{5}$$

where λ is the fault isolation rate and t_j' is the time when the jth fault is isolated. The relation between Eqs. (4) and (5) can be modeled as shown in **Figure 2**.

From another point of view, the model can be reformulated as simultaneous differential equations:

$$\frac{d}{dt} f(t) = \phi\{N - f(t)\}, \tag{6}$$

$$\frac{d}{dt} g(t) = \lambda\{f(t) - g(t)\}, \tag{7}$$

where $f(t)$ is the cumulative number of unique failures detected up to time t, with $f(t)$ corresponding to $F(\tau_i)$, and $g(t)$ is the cumulative number of unique faults isolated up to time t, with $g(t)$ corresponding to $G(\tau_j')$. By assuming $\phi \simeq \lambda \simeq \rho$, the above simultaneous differential equations can be solved in terms of $g(t)$. The results are identical to those of Eq. (3). The mean value function of Eq. (3) gives an S-shaped software reliability growth curve for the initial delay of fault isolation after the initial failure detection. As t becomes larger, the difference between the delayed S-shaped growth and the exponential growth curves becomes smaller. The parameters of Eq. (3) can be estimated by a method similar to that of the Goel-Okumoto NHPP model (see Appendix B).

The delayed S-shaped growth model gives a better estimate of the parameter N (the initial number of faults of a program) when the fault isolation data, which consist of pairs of the observation time (or cumulative effort index up to that time) and the cumulative number of faults isolated, are available, as in Example 1 in Section 6. Generally the fault isolation data are more accurate than the failure detection data because one fault may cause several different failures under different conditions if we can assume that new faults are not spawned by fixing faults. The model, however, does not fit observed data in the following cases:

1. When the time delay between failure detection and fault isolation is negligible.
2. When the test effort spent for failure detection and fault isolation is not constant.
3. When new faults are generated by fixing the inherent faults.

One of the advantages of using the fault isolation data is that there are some faults which are removed during the fault isolation process of other faults without failure detection by a test team. Sometimes the test team is responsible not only for detecting failures but also for analyzing failures or isolating failure conditions. In such cases, delayed S-shaped growth is frequently observed [2].

4. Inflection S-shaped growth model

The inflection S-shaped software reliability growth model has been developed to analyze the software failure detection process and its underlying reasons by modifying the logistic curve model which is widely used by Japanese computer makers for assessing the reliability growth of their software products. The underlying concept is that the observed software reliability growth becomes S-shaped if faults in a program are mutually dependent (some faults are not detectable before some other faults are removed).

The inflection S-shaped growth models are based on the following assumptions:

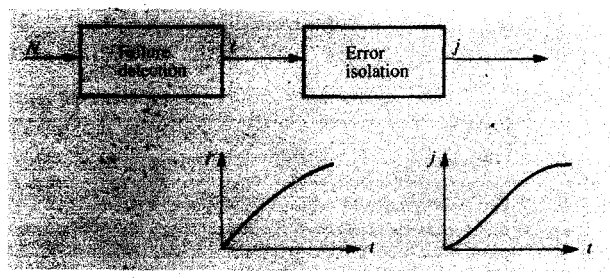

Figure 2

Model of the delayed reliability growth.

1. Some of the faults in a program are mutually dependent (there may be a set of faults which are not detectable).
2. The probability of failure detection at any time is proportional to the current number of detectable faults in a program.
3. The proportionality is constant.
4. The isolated faults can be entirely removed.

The model is characterized by the following inflection S-shaped growth mean value function $h(t)$ of NHPP:

$$h(t) = N \frac{1 - \exp(-\phi t)}{1 + \psi \cdot \exp(-\phi t)}, \tag{8}$$

where ϕ is the failure detection rate in the sense of the Jelinski-Moranda model, ψ is the inflection parameter, and $h(t)$ is the number of failures detected up to time t. The inflection parameter is defined for given r by the following equation:

$$\psi(r) = \frac{1 - r}{r}, \qquad r > 0, \tag{9}$$

where r is the inflection rate which indicates the ratio of the number of detectable faults to the total number of faults in the program. Basically the model rests on the assumption that the error discovery rate increases throughout a test period. The model becomes equivalent to the exponential growth model if the inflection rate equals 1, which is equivalent to assuming that all faults of a program are detectable from the beginning of a test. The model approaches the logistic curve model as the inflection rate tends towards zero, which is equivalent to assuming that only a few faults of a program are detectable at the beginning and faults rapidly become detectable.

The physical interpretation of the inflection rate is as follows. Let $H(\tau_i)$ be the p.d.f. of failure detection between the observation of the $(i-1)$th and the ith failure detection. As with the exponential growth model, $H(\tau_i)$ is given by the following equation:

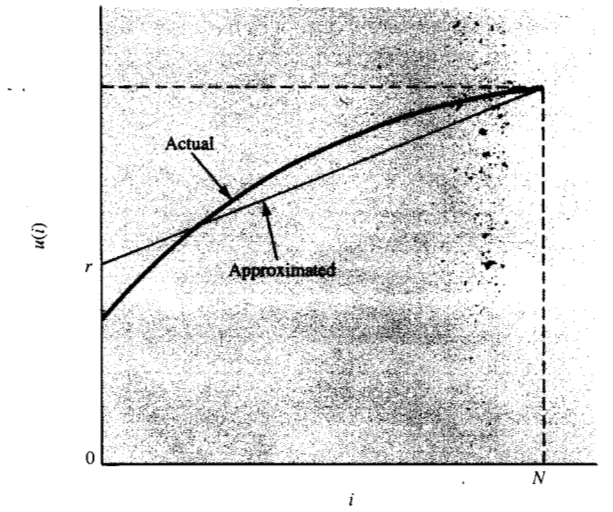

Figure 3

Number of detected failures i and the inflection rate function $u(i)$.

$$H(\tau_i) = \phi \cdot u(i) \cdot \{N - (i - 1)\}, \tag{10}$$

where $u(i)$ is the inflection rate function which satisfies the condition $0 \leq u(i)$, where $0 \leq i \leq N$. In this model, the inflection rate function is approximated by the following linear function:

$$u(i) = r + (1 - r)\frac{i}{N}. \tag{11}$$

Equation (11) means that the failure detection rate $H(\tau_i)$ is proportional to the current number of faults in the program and the proportionality is dependent upon the number of failures detected, in contrast to the total number of failures. The more failures we detect, the more undetected failures become detectable [11] (see **Figure 3**). There are two types of faults in a program:

1. Mutually independent.
2. Mutually dependent.

Faults of the first kind occur on different program paths, and faults of the second kind are on the same program paths [see **Figure 4(a)**]. Thus, the second or later faults on the same program path become detectable if and only if the preceding faults have been removed. Only faults on different paths are independent pairwise of one another. For example, every fault in **Figure 4(b)** has equal probability of being captured (i.e., the random capturing assumption). Faults in a program, however, are usually partially ordered by the execution paths of the program, as illustrated in **Figure 4(c)**. Where there exists a small number of faults in a program (the faults can be regarded as being mutually independent), we can assume the inflection rate to be 1. In an ideal case, the inflection rate function increases according to the order of the logarithmic function of the number of faults isolated, because of the lattice structure of the mutually dependent faults.

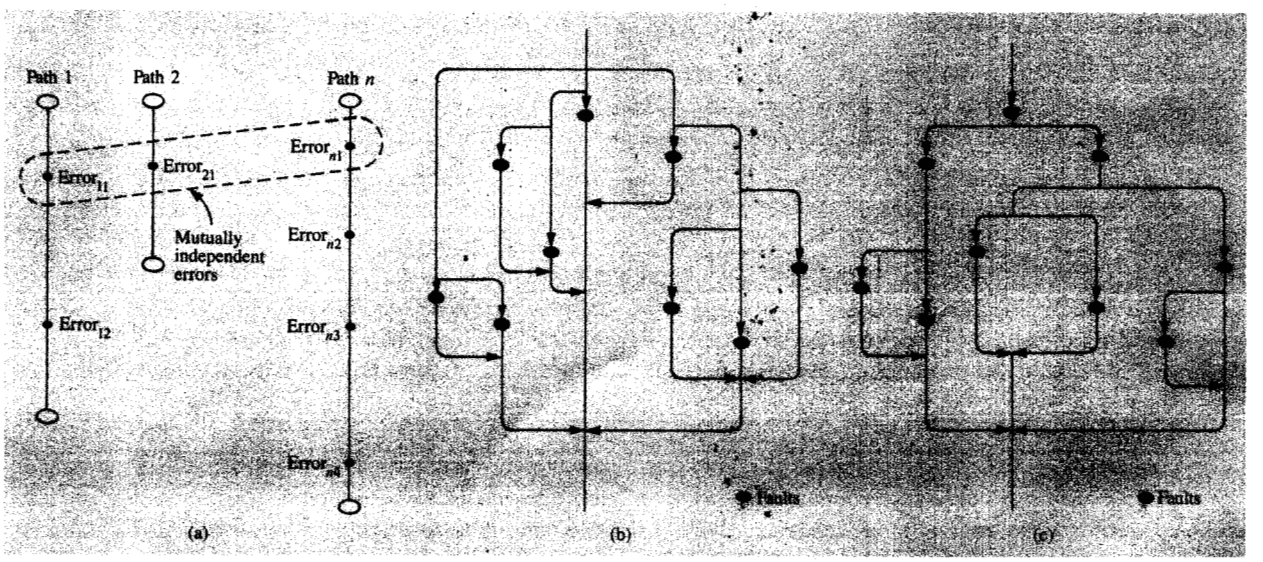

Figure 4

(a) Dependency of error occurrence; (b) independent faults in a program; and (c) dependent faults in a program.

The parameters of Eq. (8) can be estimated by a method similar to that used with the Goel-Okumoto NHPP model when the inflection rate is given (see Appendix C). Otherwise, the least squares method can be used. If the inflection rate is greater than or equal to 0.5, then the reliability growth curve does not have an inflection and becomes similar to, but not identical with, the exponential growth (see **Figure 5**). The reliability growth curve has an inflection point if the inflection rate is less than 0.5. In analysis of actual project data, not only by the mutual dependency of faults but also by the gradual increase in test effort, the inflection rate becomes smaller than 1.

The inflection S-shaped growth model gives a better estimate of the parameter N than the exponential growth model if one of the following conditions is satisfied:

1. The software is fairly large and contains a fairly large number of errors.
2. The testing effort increases (or decreases) throughout the test period.

The first condition is related to the mutual dependency of errors, and the second condition is related to the time distribution of the testing effort (see Example 2 in Section 6). The model does not fit observed behavior if the data are fault isolation data or new faults are generated by fixing the inherent faults.

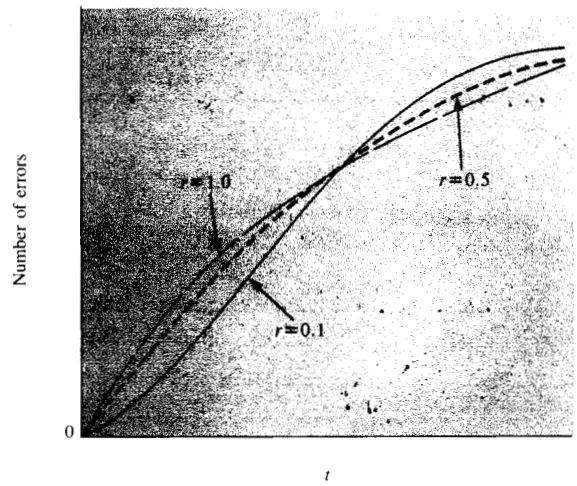

Figure 5

Parameter r and reliability growth.

5. Hyperexponential growth model

The sum of different exponential growth curves does not result in a new exponential growth curve, but in the hyperexponential growth curve. The hyperexponential software reliability growth model has been developed to analyze a failure detection process in module-structured software and its underlying causes on the basis of the ordinary exponential growth model. In particular, the model is used for analyzing a software failure detection process in which two or more different kinds of modules are tested. In other words, the model can be applied to software consisting of new modules and a set of reused (existing) modules.

In such a case, faults in these modules have different characteristics from the failure detection point of view; i.e., the proportionality of capturing undetected failures in the modules is completely different (the failure detection rate of the reused modules is smaller than that of new modules because those failures are usually difficult to detect).

The model is characterized by the following hyperexponential growth mean value function $y(t)$ of NHPP:

$$y(t) = \sum_{i=1}^{n} N_i\{1 - \exp(-\phi_i t)\}, \tag{12}$$

where n is the number of clusters of modules which have similar characteristics, N_i is the number of initial errors in cluster i, and ϕ_i is the failure detection rate for cluster i. The hyperexponential growth model is a simple sum of the exponential growth models of clusters that have different failure detection rates. The number of detected failures up to time t, $y(t)$, can be given by the following simple equation:

$$y(t) = N_0\{1 - \exp(-\phi_0 t)\}, \tag{13}$$

if and only if the failure detection rate of each cluster is nearly equal to ϕ_0; i.e., $\phi_0 \simeq \phi_i$ (for $i = 1, 2, \cdots, n$). This implies that the observed software reliability growth is not exponential, but hyperexponential, if a program under test consists of several different clusters that have different error characteristics. On the other hand, it is true that the ordinary exponential growth model can be used for analyzing the growth in reliability of module-structured software if the modules have similar characteristics, i.e., the same failure detection rate. The greater the number of clusters, theoretically the more the observed software reliability growth curve may differ from ordinary exponential growth (see **Figure 6**).

The parameters of Eq. (12) can be estimated by means of clustering the modules by characteristics, applying the ordinary exponential growth model (e.g., the Goel-Okumoto NHPP model) for the estimation of the parameters N_i and ϕ_i for each cluster i, and summing the estimation results into the final estimation. If the program consists of a single cluster of modules, then the method of estimating the exponential growth model parameters used for the Goel-Okumoto method can be used.

The hyperexponential growth model gives a better estimate of the number of faults than the ordinary

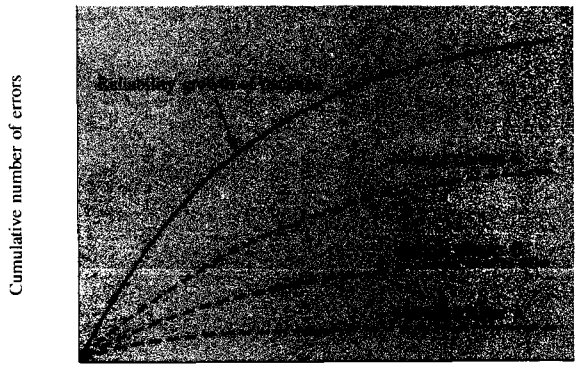

Figure 6

Hyperexponential software reliability growth.

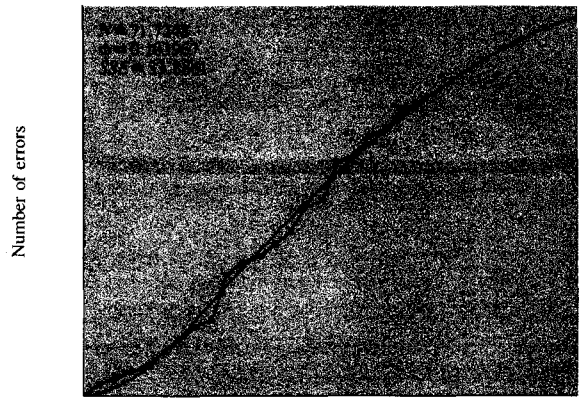

Figure 7

Small software package time/error data analysis by the delayed S-shaped model.

exponential growth model and the S-shaped software reliability growth models if one of the following conditions is satisfied:

1. The software is module-structured and the complexity of the modules is significantly different.
2. The software consists of a set of newly developed modules and a set of reused (existing) modules.
3. Some modules of the software interact directly with either the hardware devices or other system modules (which also contain some errors).

Although the observed software reliability growth curve of this hyperexponential model is similar to one of the ordinary exponential growth models, its physical interpretation is quite different. As a result, if the ordinary exponential growth model is inappropriately applied, the estimation results become unrealistic and inaccurate (typically N becomes infinite and ϕ becomes zero because the observed growth is linear before a saturation point).

6. Examples of application

Here we describe some experimental results of applying the delayed S-shaped, the inflection S-shaped, and the hyperexponential software reliability growth models. The following error data sets obtained from actual project reports are used:

1. On-line data entry software package test data (see Appendix D).
2. RADC Project A test data with calendar time [3].
3. RADC Project A test data with effort index [3].
4. RADC Project B test data with calendar time [3].
5. RADC Project B test data with effort index [3].
6. Hardware control program test data (see Appendix E).
7. PL/I application program test data (see Appendix F).

The small on-line data entry control software package has been available since 1980 in Japan. The size of the software is approximately 40 000 lines of code. The testing time (time axis) was measured on the basis of the number of shifts spent running test cases and analyzing the results. The number of persons on the test team was constant throughout the test period. **Figure 7** shows the analysis results using the delayed S-shaped growth model. The estimated mean value function is

$$g(t) = 71.7\{1 - (1 + 0.104t)\exp(-0.104t)\}. \quad (14)$$

The model fits the observed data well in this case in terms of the correlation coefficient (this is also supported by the chi square test and the Kolmogorov-Smirnov test). The test team reported a failure after it had reproduced it. Based on our field problem tracking data of this software for more than three years, the actual value of the parameter N (the number of initial faults) is 69 (the estimate is 72). In this case, the exponential growth model does not fit the observed data at all, and the estimated number of initial errors is infinite. Thus, the delayed S-shaped growth model provides a good fit to the data in this case.

The second example is Project 2 data of the Rome Air Development Center (RADC) project [3]. In this case, the test time is measured on the basis of calendar time (the

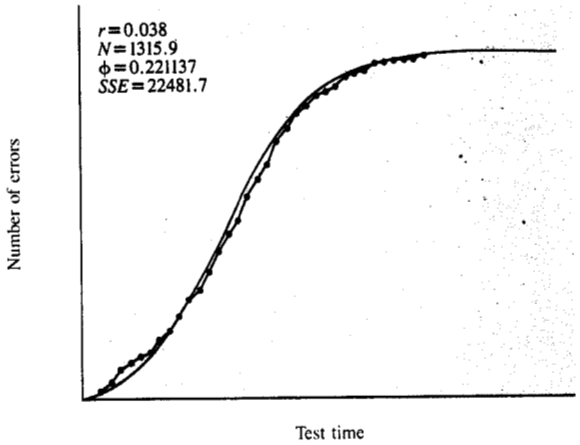

Figure 8

RADC Project A time/error data analysis by the inflection S-shaped model.

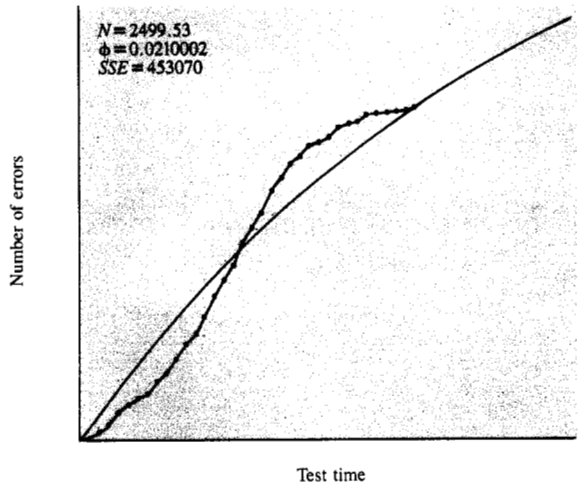

Figure 9

RADC Project A time/error data analysis by the exponential growth model.

monthly progress report). The size of the software is approximately 124 000 lines of code. It took thirty-five months to complete the test, and the test effort (the work load for the test) was not evenly distributed throughout the test period. The test effort gradually increased as the test proceeded. **Figure 8** shows the analysis results using the inflection S-shaped growth model where the inflection rate is 0.038; i.e., the observed growth in software reliability is almost equivalent to the logistic curve. Consequently, the logistic curve model may fit well in this case. The estimated mean value function obtained by the method of maximum likelihood is

$$h(t) = 1315.9 \frac{1 - \exp(-0.221t)}{1 + 25.3 \cdot \exp(-0.221t)}, \quad (15)$$

and the error sum of squares is 22 481.7. The inflection S-shaped growth model fits the observed data quite well, compared with the exponential growth model and the delayed S-shaped growth model. For example, in the case of the Goel-Okumoto NHPP model (**Figure 9**), the estimated mean value function is

$$m(t) = 2499.5\{1 - \exp(-0.021t)\}, \quad (16)$$

and the error sum of squares is 453 070 (twenty times larger than that of the inflection S-shaped growth model). In this case, the inflection S-shaped growth model provides a better fit to the data than the exponential model in terms of the correlation coefficient.

The third example is the test data of the same RADC project, but in this case the testing time is based on the testing effort index. The effort index was measured by

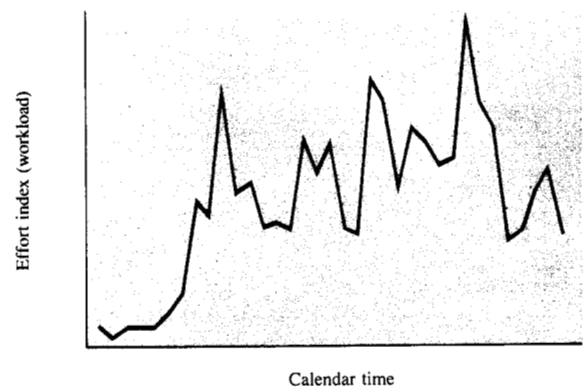

Figure 10

Time distribution of RADC Project A test effort.

observing the operation time spent for the test. As described previously, the testing effort was not homogeneously distributed, so that the effort index would be a more reasonable measure of the testing time. As illustrated in **Figure 10**, the testing effort has a peak in the latter half of the test period.

Figure 11 shows the analysis results using the inflection S-shaped growth model where the inflection rate is 0.23. This implies that the observed reliability growth curve is slightly

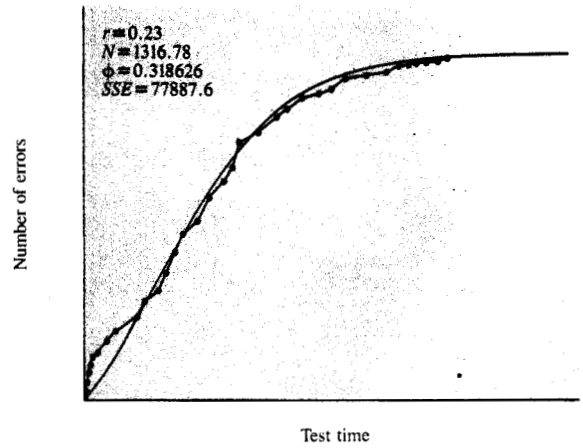

Figure 11

RADC Project A effort/error data analysis by the inflection S-shaped model.

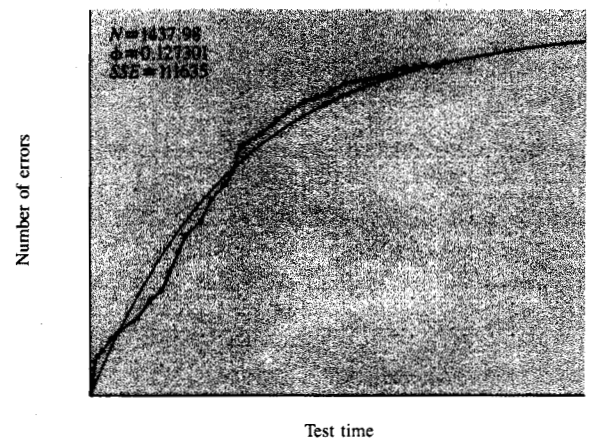

Figure 12

RADC Project A effort/error data analysis by the exponential growth model.

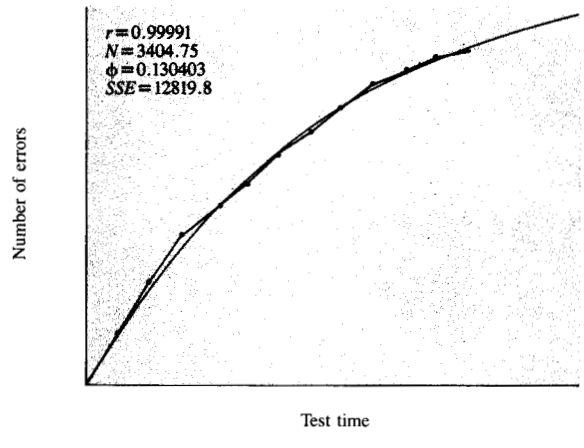

Figure 13

RADC Project B time/error data analysis by the inflection S-shaped model.

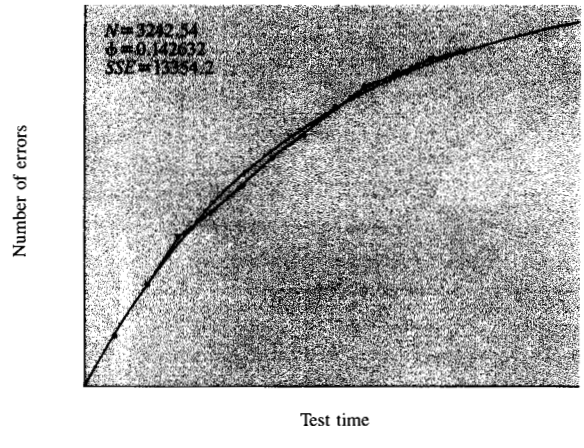

Figure 14

RADC Project B time/error data analysis by the exponential growth model.

S-shaped, reflecting the mutual dependency of faults. The strongly S-shaped growth curve of the calendar-time-based data can be regarded as being caused primarily by the nonhomogeneous time distribution of the testing effort. The estimated mean value function is

$$h(t) = 1316.8 \frac{1 - \exp(-0.319t)}{1 + 3.3 \cdot \exp(-0.319t)}, \quad (17)$$

and the error sum of squares is 77 888. The inflection S-shaped model gives a good fit to the observed data. In this case, the exponential growth model also gives a good fit (**Figure 12**). The estimated mean value function is

$$m(t) = 1438.0\{1 - \exp(-0.127t)\}, \quad (18)$$

and the error sum of squares is 111 650 (one and a half times larger than that of the inflection S-shaped growth model). In this case, the inflection S-shaped growth model provides a better fit to the data than the exponential model

in terms of the correlation coefficient. The estimated number of failures, N, indicated by the inflection S-shaped growth model compares well with the estimate of the inflection S-shaped growth model based on calendar time data (the estimated number of initial errors is 1316 based on the calendar time data and 1317 based on the effort index), but not as well with the exponential growth model. We also note that the model is quite adaptive.

The fourth example is Project 1 data of the RADC project [3]. In this case, the testing time is measured based on calendar time. The size of the software is approximately 1 317 000 lines of code. It took twelve months to complete the test, and the test effort was not homogeneously distributed throughout the test period. The testing effort gradually decreased as the test proceeded. **Figure 13** shows the analysis results using the inflection S-shaped growth model where the inflection rate is 0.99991. This implies that the observed software reliability growth is almost exponential. The estimated mean value function using the least squares method is

$$h(t) = 3404.8 \frac{1 - \exp(-0.130t)}{1 + 0.00009 \cdot \exp(-0.130t)}, \quad (19)$$

and the error sum of squares is 12 820. The inflection S-shaped model gives a good fit to the observed data. In this case, the exponential growth model also gives a good fit (**Figure 14**). The estimated mean value function is

$$m(t) = 3242.5\{1 - \exp(-0.143t)\}, \quad (20)$$

and the error sum of squares is 13 354 (which almost equals that of the inflection S-shaped growth model). In this case, both the exponential growth model and the inflection S-shaped growth model provide a good fit to the data. However, these estimated numbers are quite different from the numbers estimated based on the effort index as described below.

The fifth example is the test data from the same RADC project, but the testing time is based on the testing effort index. The effort index is based on the same definition as in the third example. As illustrated in **Figure 15**, the testing effort was not homogeneously distributed, so that the testing effort index would be more reasonable than the calendar time as the testing time unit. **Figure 16** shows the analysis results using the inflection S-shaped growth model where the inflection rate is 0.944. This implies that the observed reliability growth is slightly different from the exponential growth curve for the mutual dependency of errors. The estimated mean value function using the least squares method is

$$h(t) = 3540.9 \frac{1 - \exp(-0.135t)}{1 + 0.06 \cdot \exp(-0.135t)}, \quad (21)$$

and the error sum of squares is 19 663. The inflection S-shaped model gives a good fit to the observed data. In this

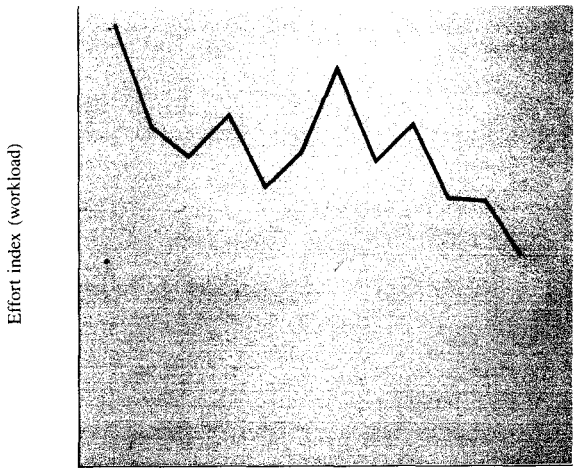

Figure 15

Time distribution of RADC Project B test effort.

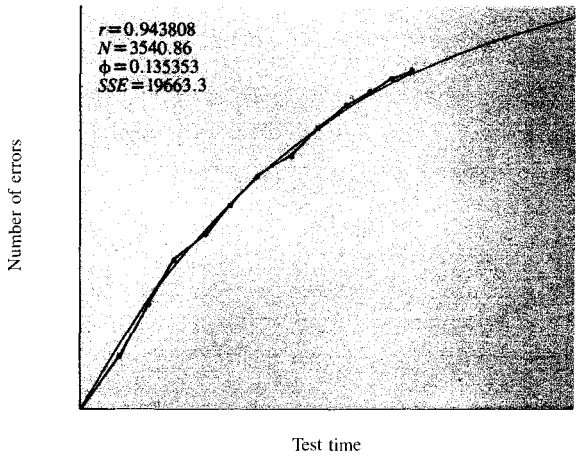

Figure 16

RADC Project B effort/error analysis by the inflection S-shaped model.

case, the exponential growth model also gives a good fit (**Figure 17**). The estimated mean value function is

$$m(t) = 3770.8\{1 - \exp(-0.117t)\}, \quad (22)$$

and the error sum of squares is 16 488 (which almost equals that of the inflection S-shaped growth model). In this case,

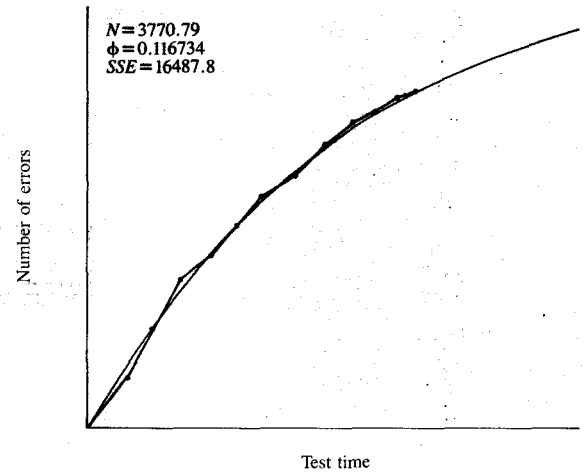

Figure 17

RADC Project B effort/error data analysis by the exponential growth model.

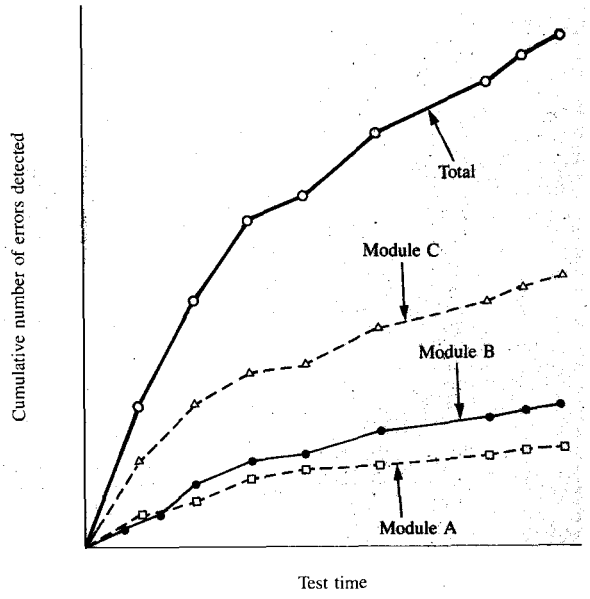

Figure 18

Observed hyperexponential growth curve.

the exponential growth model provides a better fit to the data than the inflection S-shaped growth model, but the estimated number of failures is quite large.

The sixth example is the test data of a hardware control program. The size of the software is approximately 35 000 lines of code. The testing time was measured based on calendar time. **Figure 18** shows the observed software reliability growth curves of each major module. The software consists of three major modules, i.e., the base module (Module A), the extended function module (Module B), and the newly added module (Module C). The reliability growth of the base module was relatively stable from the beginning of the test, that of the extended function module was similar, while the newly added module's reliability growth was unstable. Using the hyperexponential growth model, we obtain the analysis result shown in **Figure 19**. The estimated mean value functions are

$$m_A(t) = 58.9\{1 - \exp(-0.191t)\}, \quad (23)$$

$$m_B(t) = 94.9\{1 - \exp(-0.132t)\}, \quad (24)$$

$$m_C(t) = 164.4\{1 - \exp(-0.173t)\}. \quad (25)$$

Thus, the estimated mean value function of the total package is

$$m(t) = m_A(t) + m_B(t) + m_C(t). \quad (26)$$

At the final checkpoint, the estimated number of failures was 318 based on hyperexponential growth model analysis. The actual value after six months' testing beyond that checkpoint was 320. Thus, we can conclude that the hyperexponential growth model is quite accurate. In this case, the exponential growth model also fits the observed data because failure detection rates, ϕ_i, of each module are distributed in the range from 0.1 to 0.2 (the estimation by the hyperexponential growth model is better, but the difference is not significant).

The last example is the test data of a PL/I database application program. The size of the software is approximately 1 317 000 lines of code. The time axis is the execution time in this case. **Figure 20** shows the analysis results using the inflection S-shaped growth model where the inflection rate is 0.2. This implies that the observed reliability growth is slightly S-shaped for the mutual dependency of faults. The estimated mean value function is

$$h(t) = 347.2\,\frac{1 - \exp(-0.094t)}{1 + 4.0 \cdot \exp(-0.094t)}, \quad (27)$$

and the error sum of squares is 3406. The inflection S-shaped model gives a good fit to the observed data. In this case, the exponential growth model also gives a good fit (**Figure 21**). The estimated mean value function is

$$m(t) = 455.4\{1 - \exp(-0.027t)\}, \quad (28)$$

and the error sum of squares is 3932 (which almost equals that of the inflection S-shaped growth model). In this case, a total of 358 failures were observed. The inflection S-shaped

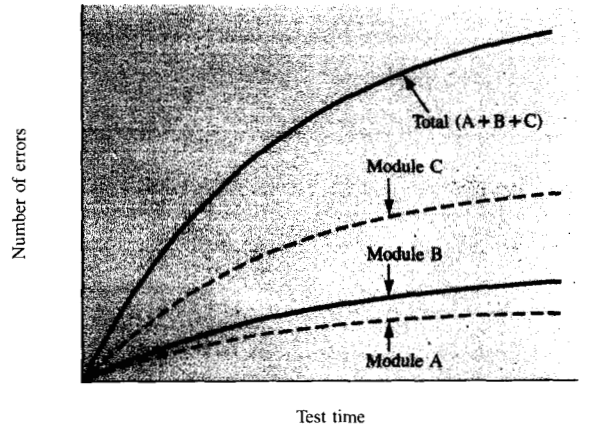

Figure 19

Hardware control program test data analysis by the hyperexponential growth model.

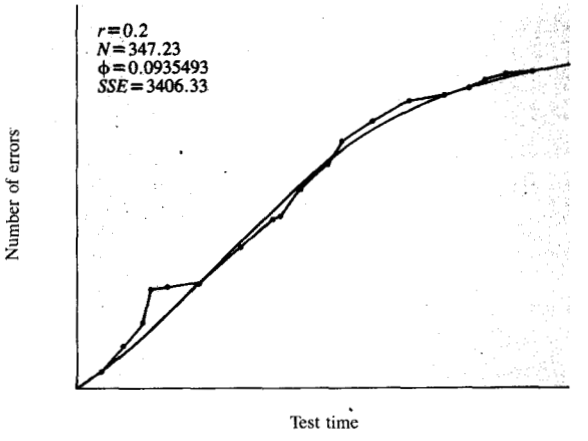

Figure 20

PL/I application program execution time data analysis by the inflection S-shaped growth model.

model provides a better fit to the data than the exponential growth model in terms of the correlation coefficient in this case.

7. Conclusion

For the effort index data or the execution-time-based data, the exponential growth model fits the observed data well, as shown by the fifth example of the previous section. This implies that the exponential growth model should be used where execution-time-based data are available [12]. There are, however, exceptions such that the observed software reliability growth is S-shaped, since the times recorded are the fault detection times (not the failure detection times) or the faults of the program are mutually dependent as in the seventh example of the previous section.

For the calendar-time-based data with nonhomogeneous time distribution of the testing effort, the exponential growth model does not fit the observed data, even if the observed reliability growth curve is similar to the exponential growth curve, as in the third example of the previous section (Fig. 12). Rather, the inflection S-shaped growth model fits well (Fig. 11). In particular, the inflection S-shaped growth model fits the calendar-time-based data which has an inflection point, as in the second example of the previous section (Fig. 8).

In the case in which the test team recorded the time when they isolated a failure symptom, the delayed S-shaped growth model gave a good fit to the data if the failure analysis time (or the fault isolation effort) was significant and the effort was homogeneously distributed, as in the first example of the previous section (Fig. 7).

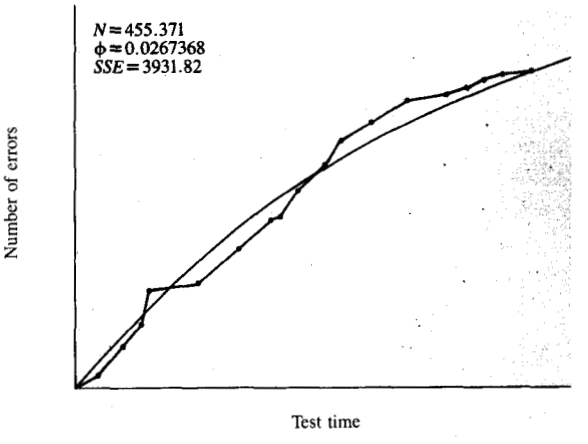

Figure 21

PL/I application program execution time data analysis by the exponential growth model.

In the case in which the errors of the tested program are not expected to be independent of one another (i.e., high program complexity, many initial errors, etc.), the inflection S-shaped growth model fits the observed data well, as in the seventh example of the previous section (Fig. 20).

In the case in which the software consists of two or more clusters of modules which have different characteristics [i.e., some existing (reused) modules and some newly developed

modules, or some modules written in high-level language and some modules written in basic assembler language, etc.], the hyperexponential growth model sometimes fits the observed data well, as in the sixth example of the previous section (Fig. 19).

Reliability growth analysis based on calendar-time data cannot be done with as much confidence as that based on the effort index or on execution time data. The reason is that the shape of the observed reliability growth curve is changeable, depending strongly upon the time distribution of the testing effort. The observed growth curve rapidly saturates once the testing effort (e.g., the number of persons on a test team) is reduced. Thus, the saturation of the observed reliability growth does not always indicate that software reliability has become stable, as in the fourth example of the previous section.

Some remaining problems concerning the software reliability growth model are still open. First, a reasonable criterion for evaluating the goodness of fit of an estimation has not been developed. The correlation coefficient is not sufficient for this problem. The chi square test is also inappropriate. For example, the estimation done for the seventh example by the exponential growth model is assured by the chi square test. Although the estimation done by the inflection S-shaped model is obviously better than that of the exponential growth model, it is not assured by the chi square test. The Kolgomorov-Smirnov test is better than the chi square test, but it still has a problem of sensitivity. Second, a reasonable method of determining the upper and lower bounds of an estimated reliability growth has not been developed. The theoretical upper and lower bounds are practically meaningless, typically in the case of the NHPP models. It only depends upon the estimated mean value function, not upon the goodness-of-fit index.

Appendix A: An algorithm for parameter estimation for the exponential growth model

The parameters of the exponential software reliability growth model are estimated as follows. Here we use the method proposed by Goel and Okumoto [10]. Suppose that the data shown in **Table 1**, pairs of the observation time and the cumulative number of failures observed, are available.

The number of failures observed up to time t, $M(t)$, is a random quantity. Assuming a Poisson distribution, the probability that $M(t)$ has the value z is given by

$$Pr\{M(t) = z\} = \frac{\{m(t)\}^z}{z!} \times e^{-m(t)}, \tag{29}$$

where $m(t)$ is a mean value function which is given by Eq. (2).

Suppose that z_i number of failures have been observed up to t_i and z_{i+1} number of failures have been observed up to t_{i+1}, where $t_{i+1} > t_i$ and $z_{i+1} > z_i$. The conditional probability of $M(t_{i+1}) = z_{i+1}$ given $M(t_i) = z_i$ is given by

$$\begin{aligned} Pr\{M(t_{i+1}) = z_{i+1} \mid M(t_i) = z_i\} &= Pr\{M(t_{i+1}) - M(t_i) = z_{i+1} - z_i\} \\ &= \frac{\{m(t_{i+1}) - m(t_i)\}^{z_{i+1}-z_i}}{(z_{i+1} - z_i)!} \times e^{-\{m(t_{i+1})-m(t_i)\}}. \end{aligned} \tag{30}$$

The joint probability that the pairs of data $\{t_i, z_i\}$ $(i = 1, 2, \cdots, n)$ are observed is therefore given by

$$\begin{aligned} &Pr\{M(0) = 0, M(t_1) = z_1, \cdots, M(t_n) = z_n\} \\ &\quad = \prod_{i=1}^{n} \frac{\{m(t_i) - m(t_{i-1})\}^{z_i - z_{i-1}}}{(z_i - z_{i-1})!} \times e^{-\{m(t_i)-m(t_{i-1})\}}. \end{aligned} \tag{31}$$

This joint probability may be used as the likelihood function for estimating the parameters N and ϕ of Eq. (2).

Estimates of N and ϕ can be found by maximizing the log-likelihood (logarithm of the likelihood) L:

$$\begin{aligned} L = &\sum_{i=1}^{n} (z_i - z_{i-1}) \ln \{m(t_i) - m(t_{i-1})\} \\ &- \sum_{i=1}^{n} \ln \{(z_i - z_{i-1})!\} - m(t_n), \end{aligned} \tag{32}$$

where

$$m(t_i) = N[1 - e^{-\phi \cdot t_i}]. \tag{33}$$

Taking the derivatives of L with respect to N and ϕ and setting them equal to zero, we obtain the equations

$$N = \frac{z_n}{1 - e^{-\phi \cdot t_n}}, \tag{34}$$

$$N \cdot t_n \cdot e^{-\phi \cdot t_n} = \sum_{i=1}^{n} (z_i - z_{i-1}) \left[\frac{t_i \cdot e^{-\phi \cdot t_i} - t_{i-1} \cdot e^{-\phi \cdot t_{i-1}}}{e^{-\phi \cdot t_{i-1}} - e^{-\phi \cdot t_i}} \right]. \tag{35}$$

Thus the estimate of ϕ is given as one of the solutions of the equation

$$\frac{t_n \cdot e^{-\phi \cdot t_n}}{1 - e^{-\phi \cdot t_n}} = \sum_{i=1}^{n} (z_i - z_{i-1}) \left[\frac{t_i \cdot e^{-\phi \cdot t_i} - t_{i-1} \cdot e^{-\phi \cdot t_{i-1}}}{e^{-\phi \cdot t_{i-1}} - e^{-\phi \cdot t_i}} \right]. \tag{36}$$

The above equation can be numerically solved with respect to ϕ.

Appendix B: An algorithm for parameter estimation for the delayed S-shaped growth model

The parameters for the delayed S-shaped software reliability growth model are estimated as follows. Here we use the method proposed by Ohba, Yamada, Takeda, and Osaki [2]. Assume that the data shown in Table 1 are available.

As with the exponential software reliability growth model, the joint probability that the pairs of data $\{t_i, z_i\}$ $(i = 1, 2, \cdots, n)$ are observed is given by

$$Pr\{M(0) = 0, M(t_1) = z_1, \cdots, M(t_n) = z_n\}$$

$$= \prod_{i=1}^{n} \frac{\{g(t_i) - g(t_{i-1})\}^{z_i - z_{i-1}}}{(z_i - z_{i-1})!} \times e^{-|g(t_i) - g(t_{i-1})|}, \quad (37)$$

where $g(t_i)$ is, from Eq. (3),

$$g(t_i) = N[1 - (1 + \phi \cdot t_i) \cdot e^{-\phi \cdot t_i}]. \quad (38)$$

This joint probability function may be used as the likelihood function for estimating the parameters N and ϕ.

Estimates of N and ϕ can be found by maximizing the log-likelihood L:

$$L = \sum_{i=1}^{n} (z_i - z_{i-1}) \ln \{g(t_i) - g(t_{i-1})\} - \sum_{i=1}^{n} \ln \{(z_i - z_{i-1})!\} - g(t_n). \quad (39)$$

Taking the derivatives of L with respect to N and ϕ and setting them equal to zero, we obtain the equations

$$N = \frac{z_n}{1 - (1 + \phi \cdot t_n) \cdot e^{-\phi \cdot t_n}}, \quad (40)$$

$$N \cdot \phi \cdot t_n^2 \cdot e^{-\phi \cdot t_n} = \sum_{i=1}^{n} \{(z_i - z_{i-1})\} \cdot \left[\frac{\phi \cdot t_i^2 \cdot e^{-\phi \cdot t_i} - \phi \cdot t_{i-1}^2 \cdot e^{-\phi \cdot t_{i-1}}}{(1 + \phi \cdot t_{i-1}) e^{-\phi \cdot t_{i-1}} - (1 + \phi \cdot t_i) e^{-\phi \cdot t_i}} \right]. \quad (41)$$

Thus, the estimate of ϕ is given as one of the solutions of the equation

$$\frac{z_n \cdot \phi \cdot t_n^2 \cdot e^{-\phi \cdot t_n}}{1 - (1 + \phi \cdot t_n) e^{-\phi \cdot t_n}} = \sum_{i=1}^{n} (z_i - z_{i-1}) \left[\frac{\phi \cdot t_i^2 \cdot e^{-\phi \cdot t_i} - \phi \cdot t_{i-1}^2 \cdot e^{-\phi \cdot t_{i-1}}}{(1 + \phi \cdot t_{i-1}) e^{-\phi \cdot t_{i-1}} - (1 + \phi \cdot t_i) e^{-\phi \cdot t_i}} \right]. \quad (42)$$

The above equation can be numerically solved with respect to ϕ.

Appendix C: An algorithm for parameter estimation for the inflection S-shaped growth model

The parameters of the inflection S-shaped software reliability growth model are estimated as follows. We use the method proposed by Ohba [15]. Assume that the data shown in Table 1 are available and the parameter r or ψ is given.

As with the exponential software reliability growth model, the joint probability that the pairs of data $\{t_i, z_i\}$ $(i = 1, 2, \cdots, n)$ are observed is given by

$$Pr\{M(0) = 0, M(t_1) = z_1, \cdots, M(t_n) = z_n\}$$

$$= \prod_{i=1}^{n} \frac{\{h(t_i) - h(t_{i-1})\}^{z_i - z_{i-1}}}{(z_i - z_{i-1})!} \times e^{-|h(t_i) - h(t_{i-1})|}, \quad (43)$$

where, from Eq. (8),

Table 1 Failure data.

Time of observation	*Cumulative number of failures*
t_1	z_1
t_2	z_2
•	•
•	•
•	•
t_n	z_n

$$h(t_i) = N \frac{1 - e^{-\phi \cdot t_i}}{1 + \psi \cdot e^{-\phi \cdot t_i}}. \quad (44)$$

This joint probability function may be used as the likelihood function for estimating the parameters N and ϕ.

Estimation of N and ϕ can be found by maximizing the log-likehood L:

$$L = \sum_{i=1}^{n} (z_i - z_{i-1}) \ln \{h(t_i) - h(t_{i-1})\} - \sum_{i=1}^{n} \ln \{(z_i - z_{i-1})!\} - h(t_n). \quad (45)$$

Taking the derivatives of L with respect to N and ϕ and setting them equal to zero, we obtain the equations

$$N = \frac{z_n \cdot (1 + \psi \cdot e^{-\phi \cdot t_n})}{1 - e^{-\phi \cdot t_n}}, \quad (46)$$

$$N \cdot t_n \cdot e^{-\phi \cdot t_n} \frac{1 - \psi + 2 \cdot \psi \cdot e^{-\phi \cdot t_n}}{(1 + \psi \cdot e^{-\phi \cdot t_n})^2} = \sum_{i=1}^{n} (z_i - z_{i-1}) \left[\frac{t_i \cdot e^{-\phi \cdot t_i} - t_{i-1} \cdot e^{-\phi \cdot t_{i-1}}}{e^{-\phi \cdot t_{i-1}} - e^{-\phi \cdot t_i}} + \frac{\psi \cdot t_i \cdot e^{-\phi \cdot t_i}}{1 + \psi \cdot e^{-\phi \cdot t_i}} + \frac{\psi \cdot t_{i-1} \cdot e^{-\phi \cdot t_{i-1}}}{1 + \psi \cdot e^{-\phi \cdot t_{i-1}}} \right]. \quad (47)$$

Thus the estimate of ϕ is given as one of the solutions of the equation

$$\frac{t_n \cdot z_n \cdot e^{-\phi \cdot t_n}}{1 - e^{-\phi \cdot t_n}} \cdot \frac{1 - \psi + 2\psi \cdot e^{-\phi \cdot t_n}}{1 + \psi \cdot e^{-\phi \cdot t_n}} = \sum_{i=1}^{n} (z_i - z_{i-1}) \left[\frac{t_i \cdot e^{-\phi \cdot t_i} - t_{i-1} \cdot e^{-\phi \cdot t_{i-1}}}{e^{-\phi \cdot t_{i-1}} - e^{-\phi \cdot t_i}} + \frac{\psi \cdot t_i \cdot e^{-\phi \cdot t_i}}{1 + \psi \cdot e^{-\phi \cdot t_i}} + \frac{\psi \cdot t_{i-1} \cdot e^{-\phi \cdot t_{i-1}}}{1 + \psi \cdot e^{-\phi \cdot t_{i-1}}} \right]. \quad (48)$$

The above equation can be numerically solved with respect to ϕ.

Appendix D: On-line data entry software package test data

The pairs of the observation time and the cumulative number of faults detected were as shown in **Table 2.**

Table 2 On-line data entry software package test data.

Time of observation (day)	*Cumulative number of faults*
1	2
2	3
3	4
4	5
6	9
7	11
8	12
9	19
10	21
11	22
12	24
13	26
14	30
15	31
16	37
17	38
18	41
19	42
20	45
21	46

Table 3 Hardware control program test data.

Time of observation (month)	*Cumulative number of failures*		
	Module A	*Module B*	*Module C*
1.0	—	10	—
1.5	17	—	47
2.0	—	18	—
3.0	24	34	77
4.5	37	47	95
6.0	42	51	99
8.0	44	63	119
11.0	50	71	133
12.0	53	75	141
13.0	54	78	147

Appendix E: Hardware control program test data

The pairs of the observation time and the cumulative number of unique failures detected were as shown in **Table 3**.

Appendix F: PL/I application program test data

The tuples of the observation time, the cumulative CPU execution time, and the cumulative number of unique failures detected were as shown in **Table 4**.

Table 4 PL/I application program test data.

Time of observation (week)	*Cumulative execution time*	*Cumulative number of failures*
1	2.45	15
2	4.90	44
3	6.86	66
4	7.84	103
5	9.52	105
6	12.89	110
7	17.10	146
8	20.47	175
9	21.43	179
10	23.35	206
11	26.23	233
12	27.67	255
13	30.93	276
14	34.77	298
15	38.61	304
16	40.91	311
17	42.67	320
18	44.66	325
19	47.65	328

Acknowledgments

The author thanks L. A. Belady of IBM, S. Osaki and S. Yamada of Hiroshima University, and H. Kobayashi of IBM. Osaki and Yamada have contributed to design of the delayed S-shaped growth model and to development of the parameter estimation method for the model. Belady has influenced our summarization of the work. In particular, the idea of the inflection S-shaped growth model was prompted by discussions with him. Kobayashi and Osaki encouraged the author to develop the hyperexponential growth model. Finally, the author especially expresses his thanks to all the referees for their comments and advice.

References

1. Z. Jelinski and P. B. Moranda, "Software Reliability Research," (*Statistical Computer Performance Evaluation*), W. Freiberger, Ed., Academic Press, Inc., New York, 1972.
2. M. Ohba, S. Yamada, K. Takeda, and S. Osaki, "S-Shaped Software Reliability Growth Curve: How Good Is It?", *Proceedings IEEE COMPSAC 82*, Chicago, 1982, pp. 38–44.
3. W. D. Brooks and R. W. Motley, "Analysis of Discrete Software Reliability Models," *Technical Report RADC-TR-80–84*, Rome Air Development Center, New York, 1980.
4. A. Kanno, "Software Engineering," (in Japanese), Union of Japanese Scientists and Engineers, Tokyo, 1979.
5. K. Sakata, "Formulation for Predictive Methods in Software Production Control: Static Prediction and Failure Rate Transition Model," (in Japanese), *Trans. IECE Japan* (Institute of Electronics and Communication Engineers of Japan) **57**, 277–283 (1974).
6. S. Yamada, M. Ohba, and S. Osaki, "S-Shaped Software Reliability Growth: Models and Comparisons," presented at the Summer Symposium on Operations Research, Daisen, Japan, 1983.
7. M. Kajiyama, S. Yamada, S. Osaki, and M. Ohba, "Comparisons of Software Reliability Growth Models," (in Japanese), *Proceedings*, Information Processing Society of Japan Spring Conference, Tokyo, Japan, 1982, pp. 401–402.
8. J. D. Musa, "Validity of Execution-Time Theory of Software Reliability," *IEEE Trans. Reliability* **R-28**, 181–191 (1979).
9. B. Littlewood and J. Verrall, "A Bayesian Reliability Growth Model for Computer Software," *Proceedings*, IEEE Symposium on Computer Software Reliability, New York, 1973, pp. 70–77.

10. A. L. Goel and K. Okumoto, "Time-Dependent Error Detection Rate Model for Software Reliability and Other Performance Measures," *IEEE Trans. Reliability* **R-28**, 206–211 (1979).
11. F. N. Parr, "An Alternative to the Rayleigh Curve for Software Development Effort," *IEEE Trans. Software Eng.* **SE-6**, 291–296 (1980).
12. P. N. Misra, "Software Reliability Analysis," *IBM Syst. J.* **22**, 262–270 (1983).
13. M. Ohba and M. Kajiyama, "A Software Reliability Growth Model with Learning Factor of Testing," (in Japanese), presented at the Information Processing Society of Japan Working Group Conference on Software Engineering, Sendai, Japan, 1983.
14. M. Ohba and N. Yonehara, "A Module-Structured Software Reliability Growth Model: Hyperexponential Model," (in Japanese), presented at the Information Processing Society of Japan Fall Conference, Fukuoka, Japan, 1982.
15. M. Ohba, "S-Shaped Software Reliability Growth Models," (in Japanese, abstract in English), presented at the Union of Japanese Scientists and Engineers Annual Reliability and Maintainability Symposium, Tokyo, 1983.

Received July 1, 1983; revised February 1, 1984

Mitsuru Ohba *Science Institute, IBM Japan, Ltd., Kowa Building No. 36, 5–19, Sanbancho, Chiyoda-ku, Tokyo 102, Japan.* Mr. Ohba is an advisory researcher in a software engineering group at the Science Institute. He is currently working on multinational language support of operating system messages, and on the software reliability assessment method and the software quality index. Mr. Ohba joined IBM at the Product Test Laboratory in Fujisawa in 1974. Since that time, he has worked to develop tools for the testing of remotely attached terminals and measurements of the performance of banking system terminals. Mr. Ohba holds B.S. and M.S. degrees in computer science from the Aoyama-Gakuin University, Tokyo. He is a member of the Information Processing Society of Japan.