Comparison of Two Estimation Methods of the Mean Time-Interval between Software Failures

Jun Hisitani, Shigeru Yamada, and Shunji Osaki

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

ABSTRACT

It is of great importance to assess software reliability quantitatively by using failure time data observed during software testing in the software development. Based on two models described by nonhomogeneous Poisson processes, i.e., exponential and delayed S-shaped software reliability growth models, we adopt mean time between software failures as a reliability assessment measure and propose a method of software reliability assessment. Since the distribution of the time-interval between software failures for the two models are improper, the means are obtained by standardization of the distributions. Further, applying the two models to actual data, numerical examples of the mean time between software failures are shown.

1. Introduction

It is of great importance to develop a highly reliable software system to increase the reliability of a computer system since once it is broken down by software failures, it might give rise to social problems. In general, a software system is developed through successive four phases: specification, design, coding, and testing. Software reliability in the testing phase which is the last stage of software development process is often discussed. The purpose of software testing is to detect and remove software faults latent in the software system. During the testing phase, many software failures are observed and recorded. A software failure is defined as an unacceptable departure of program operation caused by a fault in the software system. We assume that all detected faults are removed and no new faults are introduced into the program. Under this assumption, the cumulative number of detected faults is increasing as these are corrected, and the time-interval between software failures becomes longer. This means that the probability of software failure occurrence is decreasing, i.e., the software reliability is increasing, with testing time. A mathematical tool which treats such a software failure occurrence phenomenon during testing phase is a software reliability growth model (see, Musa et al. [1], Ramamoorthy and Bastani [2] and Yamada [3]). Many software reliability growth models have been developed for describing the software failure occurrence phenomenon and assessing the software reliability (e.g., Abdel-Ghaly et al. [4], Goel and Okumoto [5], Jelinski and Moranda [6], Littlewood [7], Musa[8], Musa and Okumoto [9], and Yamada and Osaki [10]). As quantitative measures for software reliability assessment, we can use the expected number of remaining faults in the system, the mean time interval between software failures, the software reliability function, and so on, which are derived from the software reliability growth models.

In this paper, we analyze software failure time data observed in software testing by using software reliability growth models based on a nonhomogeneous Poisson process (NHPP, see Ascher and Feingold [11]). In particular, we discuss the estimation methods of the mean time-interval between software failures as a quantitative measure, based on exponential software reliability growth model (Goel and Okumoto [5]) and delayed S-shaped software reliability growth model (Yamada et al. [12]). Since the distributions of software failure time in these models are improper, the approximation and standardization methods to obtain the mean time-interval between software failures are proposed. First, we summarize exponential and delayed S-shaped software reliability growth models, and discuss the software reliability analysis based on the method of maximum likelihood. Next, we investigate the distribution of software failure time in a software reliability growth model based on an.
NHPP and the estimation methods of the mean time-interval between software failures for the two models above. Finally, we compare two methods proposed in this paper in terms of the accuracy of estimation by using actual software failure time data.

2. Software Reliability Growth Model

2.1 Model description

During testing phase in the software development, a lot of testing resources are consumed to detect and correct software faults. A software system is subject to software failures caused by the faults remaining in the system during the testing phase. A software failure is defined as an unacceptable departure of program operation caused by a software fault in the system. Then, test data on such as software failure occurrence times or number of detected faults be observed. Using the test data, we can describe a software failure occurrence phenomenon and assess the software reliability, based on software reliability growth models. The following are assumed:

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

Let \( N(t), t \geq 0 \) denote a counting process representing the cumulative number of faults detected up to testing time \( t \). Therefore, a software reliability growth model for a failure occurrence phenomenon can be described by an NHPP as (see Goel and Okumoto [5] and Yamada and Osaki [10]):

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

where \( H(t) \) is a mean value function which indicates the expected cumulative number of faults detected up to testing time \( t \) and \( h(t) \) an intensity function which indicates the fault detection rate at testing time \( t \). Defining \( a \) be the expected initial fault content or the expected cumulative number of faults to be eventually detected, we usually assume that \( H(0) = a \).

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

In this paper, we discuss two models based on NHPPs'. The one is exponential software reliability growth model with mean value function \( m(t) \) proposed by Goel and Okumoto [5]:

\[
H(t) = m(t) = a (1 - e^{-bt}) \quad (a>0, b>0),
\]

where \( a \) is the total expected number of faults to be eventually detected and \( b \) is the fault detection rate at arbitrary testing time. This model is often applied to describe a software failure occurrence phenomenon. The other is delayed S-shaped software reliability growth model with mean value function \( M(t) \) proposed by Yamada et al. [12]:

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

where \( a \) is the total expected number of faults to be eventually detected and \( b \) is the fault detection rate per fault in the steady-state. This model is well-fitted to describe a fault removal process in the testing.

2.2 Estimation of parameters

Let \( X \) denote a random variable representing the time-interval between \((k-1)\)-st and \( k \)-th failures \((k = 1, 2, \ldots)\). Then,

\[ S_k = \sum_{i=1}^{k} X_i \]

is a random variable representing the \( k \)-th failure occurrence time where \( X_k = s_k - S_{k-1} \) \((k = 1, 2, \ldots, n; S_0 = 0)\). The joint probability density function of \( \{ S_1, S_2, \ldots, S_n \} \) for an NHPP with \( H(t) \) in (2) is given by

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

where \( 0 \leq s_1 \leq s_2 \leq \cdots \leq s_n < \infty \). Suppose that the data on failure occurrence times \( s_k \) \((k = 1, 2, \ldots, n)\) are observed during the testing phase. Then, the likelihood function for the unknown parameters in an NHPP model with \( H(t) \) is given by (5). Denoting the likelihood function in (4) by \( L \) and taking the natural logarithm of \( L \) yields

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

Then, solving the likelihood equations obtained from (6), we can estimate the parameters of a fitted software reliability growth model based on an NHPP by a method of maximum likelihood.

For the exponential software reliability growth model substituting (3) in (6), equation (6) is given by

\[
\ln L = -a (1 - e^{-bt}) + n (\ln a + \ln b) - b \sum_{k=1}^{n} s_k.
\]

We can obtain the maximum likelihood estimates of the unknown parameters \( a \) and \( b \) by solving the simultaneous
likelihood equations: \[ \frac{\partial \ln L}{\partial a} = \frac{\partial \ln L}{\partial b} = 0. \] Thus, we have

\[ \frac{n}{a} = 1 - e^{-bs_k}, \]  
\[ \frac{n}{b} = \sum_{k=1}^{n} s_k + a s_n e^{-bs_k}, \]

which can be solved numerically. In the same way, we obtain the maximum likelihood estimates \( \hat{a} \) and \( \hat{b} \) for delayed S-shaped software reliability growth model by solving the following simultaneous equations numerically:

\[ \frac{n}{a} = 1 - (1 + bs_n) e^{-bs_k}, \]
\[ 2n/b = \sum_{k=1}^{n} s_k + ab s_n^2 e^{-bs_k}. \]

3. Mean Time Between Failures

3.1 Failure time distribution

We can estimate the mean value functions \( m(t) \) in (3) and \( M(t) \) in (4) by using the estimated model parameters discussed above. It is useful to estimate and predict software reliability measures derived from the estimated models in order to assess software reliability quantitatively. In this paper, we adopt mean time between software failures in the testing phase or the operational phase as the measure, which is often measured in CPU time (see [8]).

From (5), considering the marginal density of the \( k \)-th software failure time \( S_k \) \( (k=1,2,...,n) \), we have the probability density function of \( S_k \):

\[ f_s(t) = \frac{h(t)1-H(t)}{\Gamma(k)} \exp[-H(t)] \quad (k=1,2,...,n), \]

where \( \Gamma(k) \) denotes a Gamma function. From (12), the cumulative distribution function is given by

\[ F_s(t) = \frac{1}{\Gamma(k)} \int_{0}^{t} u^{k-1} e^{-u} du. \]

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

\[ F_s(\infty) = 1 - \frac{1}{\Gamma(k)} \int_{a}^{\infty} u^{k-1} e^{-u} du < 1, \]

where \( H(\infty) = a \), e.g., \( m(\infty) = a \) in (3) and \( M(\infty) = a \) in (4). The equation (14) implies that there does not exist the mean of the distribution of \( S_k \). Then, the joint distribution function of \( \{ S_1, S_2, \ldots, S_n \} \) is also improper. Consequently, there does not exist the mean time between failures, which is given by

\[ E[S_k] = E[S_k] - E[S_{k,1}] \quad (k=1,2,...,n). \]

However, the mean time between failures can be obtained approximately by calculating the inverse transformation of the mean value function (see Goel and Okumoto [5]). That is, for the exponential and delayed S-shaped software reliability growth models, we have the mean time to \( k \)-th software failure, \( E[S_k] \), by solving the following equations in terms of \( s_k \):

\[ k = m(s_k) \quad \text{and} \quad k = M(s_k). \]

Then, we have

\[ \hat{s}_k = m^{-1}(k) = \frac{1}{\hat{b}} \ln(1 - \frac{k}{a}) \quad (k=1,2,...,n), \]

for the exponential software reliability growth model and

\[ \hat{s}_k = M^{-1}(k) = \sqrt{\frac{2}{\hat{b}}} \ln \left( \frac{\hat{b}}{k} \right) \quad (k=1,2,...,n), \]

for the delayed S-shaped software reliability growth model (see equation (27)). Thus, using these approximation methods, we can estimate mean time between failures as \( \hat{s}_k = \hat{s}_k - \hat{s}_{k-1} \) \( (k = 1, 2, \ldots, n) \).

3.2 Standardization of distribution

In this paper, to get the accurate mean time between software failures, we standardize the failure time distribution \( F_s(t) \) in (13) as

\[ G_s(0) = \frac{F_s(0)}{F_s(\infty)} = \frac{F_s(0)}{\int_{a}^{\infty} u^{k-1} e^{-u} du / \Gamma(k)}. \]

Then, it is shown that \( G_s(0) = 0 \) and \( G_s(\infty) = 1 \), i.e., \( G_s(t) \) is proper. The probability density function is given by

\[ g_s(t) = \frac{f_s(t)}{F_s(\infty)} = \frac{f_s(t)}{\int_{a}^{\infty} u^{k-1} e^{-u} du / \Gamma(k)}. \]

And the mean time to \( k \)-th software failure is given by

\[ E[S_k] = \int_{a}^{\infty} t g_s(t) dt \quad (k=1,2,...,n). \]

From (20) and (21), the mean time between software failures is given by

\[ E[S_k] = \int_{a}^{\infty} F_s^{-1}(u) u^{k-1} e^{-u} du / \int_{a}^{\infty} u^{k-1} e^{-u} du. \]

Thus, the mean time between software failures can be obtained from (15) and (22).

For the exponential software reliability growth model with \( m(t) \) in (3), the mean time to \( k \)-th software failure is calculated as
Then, the mean time between failures $E[S_k] (k=1,2,\ldots,n)$ for this model can be obtained by

$E[S_k] = \frac{1}{b} \int_a^b \frac{\ln(a - u) \ e^{ku} \ du}{\int_u^b \ e^{ku} \ du} \int_a^u \ e^{ku} \ du$

$= \frac{1}{b} \int_a^b \ln(a - u) \ e^{ku} \ du \int_u^b \ e^{ku} \ du$

(23)

since

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

(24)

Then, the mean time between failures $E[S_k] (k=1,2,\ldots,n)$ for this model can be obtained by (15).

For the delayed S-shaped software reliability growth model with $M(t)$ in (4), we need some tricky manipulations. Letting $u = M(t)$ yields

$\ln(1 + bt) - bt = \ln\left(\frac{a \cdot u}{a}\right)$.

(25)

If $bt = 1$ then equation (25) is

$\ln(1+b) - bt \sim (bt)^2/2$.

(26)

Then, we have

$H^{-1}(u) = M^{-1}(u) = \frac{1}{b} \sqrt{2 \ln \left(\frac{a}{a-u}\right)}$.

(27)

by Taylor expansion. Therefore, the mean time to $k$-th software failure is calculated as

$E[S_k] = \frac{1}{b} \int_a^b \sqrt{2 \ln \left(\frac{a}{a-u}\right)} \ e^{ku} \ du \int_u^b \ e^{ku} \ du$.

(28)

Then, the mean time between failures $E[S_k] (k=1,2,\ldots,n)$ for this model can be obtained by (15).

**4. Analysis of Failure Time Data**

We analyze actual software fault data observed during testing phase to show numerical examples for application of the method described above. Two data sets are analyzed here.

The first data set (DATA 1) is available in the form $x_k$ ($k=1,2,\ldots,26$) (days), which was cited by Goel and Okumoto [5]. It is difficult to judge that the relationship between the testing time and the cumulative number of detected faults shows an exponential growth curve or S-shaped growth curve (see Fig.1). Then, we analyze DATA 1 by applying both exponential and delayed S-shaped software reliability growth models. The model parameters of each model can be estimated by the method of maximum likelihood. For the exponential software reliability growth model with $m(t)$ in (3), we obtained the estimated parameters as $\hat{a} = 33.99$ and $\hat{b} = 0.00579$, i.e.,

$\hat{m}(t) = 33.99 \left(1 - e^{-0.00579t}\right)$.  

(29)

For the delayed S-shaped software reliability growth model with $M(t)$ in (4), we obtained $\hat{a} = 27.49$ and $\hat{b} = 0.0186$, i.e.,

$\hat{M}(t) = 27.49 \left[1 - (1+0.0186t) e^{-0.0186t}\right]$.  

(30)

Fig.1 shows the estimated mean value functions in (29) and (30) along with the actual data. The Kolmogorov-Smirnov goodness-of-fit test (see Yamada [2]) can show that the NHPP models with mean value function $\hat{m}(t)$ in (29) and $\hat{M}(t)$ in (30) fit DATA 1 statistically.

Based on the estimation results above, we calculate the mean times between failures by using the approximation and standardization methods discussed in 3.1 and 3.2, respectively. Table 1 shows the estimation results of the mean times between failures for the exponential and delayed S-shaped software reliability growth models. From fault tracking research after the test, we have found that the estimated initial fault content of the exponential software reliability growth model is more accurate than that of the delayed S-shaped software reliability growth model. Therefore, we conclude that the exponential software reliability growth model is fitted to DATA 1 better. In terms of estimation accuracy for the fitted exponential software reliability growth model, we calculate the sum of square errors between the actual and estimated mean times between failures to compare the approximation method in 3.1 and standardization method in 3.2. The sum of square errors for the $n$ actual data is given by

\[
\sum_{i=1}^{n} (y_i - \hat{y}_i)^2
\]
The estimation results of mean time between software failures for DATA 1.

<table>
<thead>
<tr>
<th>Failure No.</th>
<th>Exponential Reliability Growth Model</th>
<th>Delayed S-shaped Reliability Growth Model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Approximation $\widehat{X}_k$</td>
<td>Standardization $\hat{E}[X_k]$</td>
</tr>
<tr>
<td>1</td>
<td>5.16</td>
<td>5.25</td>
</tr>
<tr>
<td>2</td>
<td>5.32</td>
<td>5.40</td>
</tr>
<tr>
<td>3</td>
<td>5.48</td>
<td>5.59</td>
</tr>
<tr>
<td>4</td>
<td>5.66</td>
<td>5.79</td>
</tr>
<tr>
<td>5</td>
<td>5.86</td>
<td>6.00</td>
</tr>
<tr>
<td>6</td>
<td>6.06</td>
<td>6.22</td>
</tr>
<tr>
<td>7</td>
<td>6.28</td>
<td>6.47</td>
</tr>
<tr>
<td>8</td>
<td>6.52</td>
<td>6.73</td>
</tr>
<tr>
<td>9</td>
<td>6.78</td>
<td>7.03</td>
</tr>
<tr>
<td>10</td>
<td>7.05</td>
<td>7.35</td>
</tr>
<tr>
<td>11</td>
<td>7.35</td>
<td>7.70</td>
</tr>
<tr>
<td>12</td>
<td>7.68</td>
<td>8.09</td>
</tr>
<tr>
<td>13</td>
<td>8.04</td>
<td>8.54</td>
</tr>
<tr>
<td>14</td>
<td>8.43</td>
<td>9.03</td>
</tr>
<tr>
<td>15</td>
<td>8.86</td>
<td>9.60</td>
</tr>
<tr>
<td>16</td>
<td>9.34</td>
<td>10.24</td>
</tr>
<tr>
<td>17</td>
<td>9.88</td>
<td>10.98</td>
</tr>
<tr>
<td>18</td>
<td>10.47</td>
<td>11.80</td>
</tr>
<tr>
<td>19</td>
<td>11.15</td>
<td>12.71</td>
</tr>
<tr>
<td>20</td>
<td>11.92</td>
<td>13.71</td>
</tr>
<tr>
<td>21</td>
<td>12.81</td>
<td>14.75</td>
</tr>
<tr>
<td>22</td>
<td>13.83</td>
<td>15.79</td>
</tr>
<tr>
<td>23</td>
<td>15.04</td>
<td>16.80</td>
</tr>
<tr>
<td>24</td>
<td>16.47</td>
<td>17.72</td>
</tr>
<tr>
<td>25</td>
<td>18.21</td>
<td>18.49</td>
</tr>
<tr>
<td>$F$</td>
<td>7180</td>
<td>7062</td>
</tr>
</tbody>
</table>

where $x_k$ is the actual time-interval between $(k-1)$-st and $k$-th failures calculated by $x_k = s_k - s_{k-1}$ $(k=1, 2, \ldots, n)$, and $\hat{E}[X_k]$ is the estimated mean time between failures. From Table 1, we have $F=7180$ by the approximation method and $F=7062$ by the standardization method for the fitted exponential software reliability growth model. This means that the standardization method calculates mean time between failures more accurately than the approximation method. Fig. 2 shows the actual and estimated mean times between failures based on the standardization method for DATA 1.

The second data set (DATA 2) is available in the form $t_k$ $(k=1, 2, \ldots, 19)$ (hours), which was cited by Yamada and Osaki [13]. For the exponential software reliability model, we can not obtain the finite estimated values of model parameters, $a$ and $b$. Then, it is not appropriate to apply the exponential software reliability growth model to DATA 2. For the delayed S-shaped software reliability growth model, we obtained $\widehat{a} = 45.20$ and $\widehat{b} = 0.1014$, i.e.,

$$M(t) = 45.20 \left[ 1 - (1 + 0.1014t)^e \right]^{-0.1014}.$$  

Fig. 3 shows the estimated mean value function in (32) along with the actual data. The Kolmogorov-Smirnov goodness-of-fit test can show that the NHPP model with mean value function $M(t)$ in (32) fits DATA 2 statistically.

Table 2 shows the estimation results of the mean times between failures based on this estimation results. From Table 2, we have $F=11.17$ by the approximation method and $F=10.41$ by the standardization method for the delayed S-shaped software reliability growth model. It means that the standardization method calculates mean time between failures more accurately than the approximation method. Fig. 4 shows the actual and estimated mean times between failures based on the standardization method for DATA 2.
of failure time distribution. Concretely, we have discussed the mean time between failures for two models, i.e., the exponential and delayed S-shaped software reliability growth models based on NHPPs. Generally, we can not obtain the mean time to software failure because the failure times in these models have improper distributions. Then, we have proposed the approximation and standardization methods to obtain mean time between failures. Further, we have shown numerical illustrations of the mean time between software failures by analyzing actual software failure time data based on these NHPP models and compared the approximation and standardization methods in terms of estimation accuracy. As the result, we have confirmed that the standardization method is well-fitted to the actual data set.

In future, we are going to analyze more actual data sets on software failure time and study the applicability of the standardization method.

REFERENCES


