TESTING-DOMAIN DEPENDENT SOFTWARE RELIABILITY MODELS

SHIGERU YAMADA
Department of Industrial and Systems Engineering, Hiroshima University
Higashi-Hiroshima 724, Japan

HIROSHI OHTERA
Information Processing Center, Okayama University of Science
Okayama 700, Japan

MITSURU OHBA
Tokyo Research Laboratory, IBM Japan, Ltd.
Tokyo 102, Japan

Abstract—An implemented software system is tested to detect and correct software errors during the testing phase in software development. The behavior of software error detection phenomenon in the testing phase can be described by a stochastic software reliability growth model. Software reliability is closely related to the quality and quantity of test-cases executed by software testing. Then, we discuss software reliability growth models based on testing-domain in a software system which is to cause the test-cases executed by testing. The models are formulated by nonhomogeneous Poisson processes. Further, we propose three kinds of different testing-domain functions. Finally, numerical illustrations of software reliability analysis for actual error data and comparison among existing models in terms of goodness-of-fit are presented.

1. INTRODUCTION

At present, software reliability measurement and assessment are of great importance to achieve a highly reliable computer system. A software development process is generally composed of four phases [1]: specification, design, coding and testing. Then, software failures are caused by software errors latent in the implemented software system. A software failure is said to occur when the software system cannot compute or perform its specified functions properly due to the errors. The software errors are injected in the software system by human work during specification, design and coding phases of the software development process. Synonymously, we may say that an error detection or isolation in the testing phase means a software failure occurrence. In order to prevent software errors in the software development process, many technologies have been studied in the field of software engineering, for example, structured programming, design review and documentation technology. However, the software developer cannot produce an error-free software system due to human error. Therefore, the testing phase in the software development process is of great importance to achieve a highly reliable software system.

Software errors are detected and corrected in the software testing phase. By the analysis of the results, the software managers have to assess the reliability of an implemented software system. The software error detection phenomenon can be regarded as a software reliability growth process. As a mathematical model for the reliability growth process, a software reliability growth model [2,3] has been studied by many researchers. The software reliability growth model describes the relationship between the number of detected software errors and the time span of software testing. The model enables us to estimate and predict software reliability measures such as the initial error content in a software system, the time-interval between software failures, the software reliability function and so on. Several leading software reliability growth models have been proposed by Goel and Okumoto [4], Jelinski and Moranda [5], Littlewood [6], Moranda [7], Musa et al. [8], Ohtera et al. [9] and Yamada and Osaki [10].
In this paper, we discuss three kinds of software reliability growth models, based on testing-domain and defined as a set of the modules and functions in a software system, which have been influenced by the testing-cases executed by software testing. Therefore, it is closely related to the time-dependent behavior of a software error detection phenomenon during the software testing. In Section 2, we propose three kinds of testing-domain functions representing the error-distribution in the software system. In Section 3, the models are formulated by nonhomogeneous Poisson processes by using the three testing-domain functions discussed in Section 2. Finally, numerical illustrations of software reliability analysis for actual error data and comparison among existing models in terms of goodness-of-fit are presented in Section 4.

2. SOFTWARE TESTING-DOMAIN

In the testing phase of a software development process, the software engineer applies various techniques to software testing in which the software errors are detected and corrected. The well-known techniques are white-box testing and black-box testing. The difference of the two techniques is a design method of test-cases which are pairs of input/output data to check software functions and specifications. In this paper, we suppose that a software error detection phenomenon doesn't relate to the method of test-case design, and consider the behavior of the error detection phenomenon for executed test-cases.

The software engineers detect and correct software errors latent in the implemented software system by executing many test-cases. Then, the functions in a software system have been influenced by the executed test-cases, i.e., the number of detectable errors by software testing is increased. The set of influenced functions is called a testing-domain in software system (see Figure 1). As the total number of executed test-cases is increased, the testing-domain is spreading over the software system. That is, the testing-domain growth rate is closely related to the quality and quantity of the executed test-cases. If software errors exist in the isolated testing-domain, then the errors influence on the output source. Thus, the software engineers wish to know the relationship between the testing-domain growth rate and the number of detected errors during software testing. However, it is difficult to directly observe the time-dependent behavior of the testing-domain growth rate. Therefore, in this paper we assume the following (see Figure 1):

1. The testing-domain growth rate is proportional to the number of errors existing in the isolated testing-domain.
2. For the isolated testing-domain, the number of errors detected in the testing time interval \((t, t + \Delta t]\) is proportional to the number of errors remaining in the software system outside of the isolated testing-domain.

From the above assumptions, we get the following differential equation:

\[
\frac{du_a(t)}{dt} = \lim_{\Delta t \to 0} \frac{u_a(t + \Delta t) - u_a(t)}{\Delta t} = v[a - u_a(t)], \quad (a > 0, \ v > 0),
\]

(1)
where $a$ is the initial error content in the software system, and $v$ the testing-domain growth rate. In equation (1), $u_a(t)$ is the total number of errors existing in the isolated testing-domain at testing time $t$. Under the boundary condition $u_a(0) = 0$ which means that the testing-domain at time zero is empty, solving (1) yields:

$$u_a(t) = a(1 - e^{-vt}),\quad (2)$$

(see Figure 2) where $(1 - e^{-vt})$ is the testing-domain coverage ratio to the final testing-domain to be covered. We call $u_a(t)$ in (2) a testing-domain function.

The equations (1) and (2) mean that the testing-domain growth rate is constant throughout the testing, i.e., the error distribution in the software system is uniform. However, it is true in the actual situation that the error distribution exhibits various patterns. Then, we consider the two kinds of error distribution as alternatives to the uniform error distribution of (2):

(Pattern 1) The pattern that the software errors concentrate in the testing-domain isolated during the initial stage of testing.

(Pattern 2) The pattern that the software errors concentrate in the testing-domain isolated during the middle and final stage of testing.

The error distributions above are related to the skill of test personnel. Then, we describe (Pattern 1) and (Pattern 2) by testing-domain functions $u_b(t)$ and $u_c(t)$, respectively. The testing-domain function $u_b(t)$ is described by solving (2) under the boundary condition $u_b(0) > 0$ which means that the testing-domain at time zero is not empty:

$$u_b(t) = a(1 - pe^{-vt}),\quad (1 > p > 0),\quad (3)$$

where $p$ is the parameter representing the error distribution patterns. We suppose that the initial testing-domain size is given by $a(1 - p)$. Equation (3) results in the uniform error distribution described by (2) when $p = 1$. Further, from Equation (1) the testing-domain function $u_c(t)$ is described by the following differential equations:

$$\frac{dz(t)}{dt} = v[a - z(t)],$$

$$\frac{du_c(t)}{dt} = v[z(t) - u_c(t)],\quad (4)$$

Figure 2. The time-dependent behaviors of testing-domain functions.
where \( z(t) \) is the error distribution specified by testing-domain growth at testing time \( t \). Solving (4) yields:

\[
uc(t) = a [1 - (1 + vt) e^{-vt}] .
\]

The quantities \( (1 - pe^{-vt}) \) in (3) and \( [1 - (1 + vt)e^{-vt}] \) in (5) are respectively the testing-domain coverage ratios to the final testing-domains to be covered (see Figure 2).

### 3. RELIABILITY GROWTH MODELS

In general, a software system is subject to software failures caused by the errors remaining in the system. We propose software reliability growth models based on the testing-domain discussed above, which describe software error detection phenomena during software testing. The models stand on the following fundamental assumptions:

1. A software failure is caused by an error.
2. Each time a failure occurs the error which caused it can be immediately removed.
3. A correction of detected errors doesn't introduce any new error.
4. The number of errors detected in the small time interval \((t, t + \Delta t)\) is proportional to the number of errors remaining in the isolated testing-domain at time \( t \).

Let \( \{N(t), t \geq 0\} \) denote a counting process representing the cumulative number of errors detected up to testing time \( t \) \((t \geq 0)\). Then, a software reliability growth model for such an error detection process can be described by a nonhomogeneous Poisson process (NHPP, see [3,11]) as

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

where \( m(t) \) is a mean value function representing the expected cumulative number of errors detected up to testing time \( t \), i.e., \( m(t) \) is the expected value of \( N(t) \) (see [4,9,10]). It is assumed that \( m(t) \) has a boundary condition \( m(0) = 0 \). From Assumption 4, since the number of errors detected in the small time interval \((t, t+\Delta t)\) is proportional to the detectable errors in the isolated testing-domain at time \( t \), we have the following differential equation:

\[
\frac{dm(t)}{dt} = \lim_{\Delta t \to 0} \frac{m(t + \Delta t) - m(t)}{\Delta t} = b[u(t) - m(t)], (b > 0),
\]

where \( b \) is the proportional constant, i.e., the error detection rate per error \((1 > b > 0)\). Solving (7) with respect to three kinds of \( u(t) \), i.e., \( u_a(t) \) in (2), \( u_b(t) \) in (3) and \( u_c(t) \) in (5), respectively:

\[
m_a(t) = a \left[ 1 - \frac{1}{v-b} (ve^{-bt} - be^{-vt}) \right], \quad (v \neq b),
\]

\[
m_b(t) = a \left[ 1 - \frac{1}{v-b} ((v-b+b) e^{-bt} - bpe^{-vt}) \right], \quad (v \neq b, 1 \geq p > 0),
\]

\[
m_c(t) = a \left[ 1 - \left( \frac{v}{v-b} \right)^2 e^{-bt} + \frac{b}{v-b} \left( vt + \frac{2v-b}{v-b} e^{-vt} \right) \right], \quad (v \neq b).
\]

Clearly, in the case of \( v = b \) for (8), we have the mean value function of a delayed S-shaped software reliability growth model (see [3,10,12]). Further, the intensity functions of NHPP models with (8)-(10) are, respectively:

\[
\lambda_a(t) \equiv \frac{dm_a(t)}{dt} = \frac{avb}{v-b} (e^{-bt} - e^{-vt}),
\]

\[
\lambda_b(t) \equiv \frac{dm_b(t)}{dt} = \frac{abp}{v-b} (be^{-bt} - ve^{-vt}),
\]

\[
\lambda_c(t) \equiv \frac{dm_c(t)}{dt} = \frac{abv^2}{v-b} \left[ \frac{1}{v-b} e^{-bt} - \left( \frac{1}{v-b} + t \right) e^{-vt} \right].
\]
The reliability growth parameters $a$, $b$, $v$, and $p$ in the mean value functions $m_a(t)$, $m_b(t)$, and $m_c(t)$ can be estimated by a method of maximum-likelihood [3]. Suppose that the data on the cumulative number of detected errors $y_k$ in a given time interval $[0, t_k]$ ($k = 1, 2, \ldots, n$; $0 < t_1 < t_2 < \cdots < t_n$) are observed. Then, based on an NHPP model described by (6), the joint probability mass function, i.e., the likelihood function, for the observed data, is given by

$$L \equiv \Pr\{N(t_1) = y_1, N(t_2) = y_2, \ldots, N(t_n) = y_n\} = \prod_{k=1}^{n} \frac{[m(t_k) - m(t_{k-1})]^{y_k - y_{k-1}}}{(y_k - y_{k-1})!} \cdot \exp[-m(t_n)],$$

where $t_0 = 0$ and $y_0 = 0$. Therefore, substituting $m_a(t)$ in (8), $m_b(t)$ in (9), and $m_c(t)$ in (10) for $m(t)$ in (13), the reliability growth parameters $a$, $b$, $v$, and $p$ can be estimated by maximizing the likelihood function in (13).

4. NUMERICAL EXAMPLES

In this section, we analyze actual software error data by using software reliability growth models with mean value functions $m_a(t)$, $m_b(t)$, and $m_c(t)$. We apply these models to three data sets of software error: DS1, DS2, and DS3. DS1 is in the form $(t_k, y_k)$ ($k = 1, 2, \ldots, 12; t_k\text{ (months)}$) and DS2 in the form $(t_k, y_k)$ ($k = 1, 2, \ldots, 35; t_k\text{ (months)}$), which were cited by Brooks and Motley [13]. DS3 is in the form $(t_k, y_k)$ ($k = 1, 2, \ldots, 19; t_k\text{ (weeks)}$) which was cited by Ohba [12].

![Figure 3. The estimated mean value function $m_a(t)$ for DS3.](image)

Using the method of maximum-likelihood, the reliability growth parameters $a$, $b$, $v$, and $p$ in the mean value functions $m_a(t)$ in (8), $m_b(t)$ in (9), and $m_c(t)$ in (10) for the data sets DS1–DS3, are estimated:

$$m_a(t) = a(1 - e^{-vt}),$$

$$DS1: \quad \dot{a} = 3210.9, \quad \dot{b} = 0.1473, \quad \dot{v} = 14.33,$$

$$DS2: \quad \dot{a} = 1478.8, \quad \dot{b} = 0.1051, \quad \dot{v} = 0.1039,$$

$$DS3: \quad \dot{a} = 459.1, \quad \dot{b} = 0.06818, \quad \dot{v} = 1.653.$$
\( m_b(t) \) \( (u_b(t) = a(1 - pe^{-vt}))): \\
DS1: \( a = 3210.9, \quad \hat{\beta} = 0.1473, \quad \hat{\nu} = 14.33, \quad \hat{p} = 1.0, \)
DS2: \( a = 1478.1, \quad \hat{\beta} = 0.10461, \quad \hat{\nu} = 0.10460, \quad \hat{p} = 1.0, \)
DS3: \( a = 381.1, \quad \hat{\beta} = 0.1699, \quad \hat{\nu} = 0.1694, \quad \hat{p} = 0.7753. \)

\( m_c(t) \) \( (u_c(t) = a[1 - (1 + vt)e^{-vt}])): \\
DS1: \( a = 3210.9, \quad \hat{\beta} = 0.1473, \quad \hat{\nu} = 28.58, \)
DS2: \( a = 1371.4, \quad \hat{\beta} = 0.1856, \quad \hat{\nu} = 0.1757, \)
DS3: \( a = 464.1, \quad \hat{\beta} = 0.06646, \quad \hat{\nu} = 3.748. \)

Then, for DS3, the estimated mean value functions \( \hat{m}_a(t), \hat{m}_b(t), \) and \( \hat{m}_c(t) \) are plotted in Figures 3-5 along with the actual software error data, respectively. Also, the estimated testing-domain functions \( \hat{u}_a(t), \hat{u}_b(t), \) and \( \hat{u}_c(t) \) are plotted in Figure 6. Further, the Kolmogorov-Smirnov goodness-of-fit test (see [3]) can show that the three kinds of software reliability growth models with estimated mean value functions \( \hat{m}_a(t), \hat{m}_b(t), \) and \( \hat{m}_c(t) \) are well-fitted to DS1-DS3.

![Diagram](image)

Figure 4. The estimated mean value function \( m_b(t) \) for DS3.

Now, by using DS1-DS3, let us compare the three estimated models with two NHPP models, i.e., an exponential software reliability growth model proposed by Goel and Okumoto [4] and a delayed S-shaped software reliability growth model proposed by Yamada and Osaki [10], with respect to estimation accuracy. As the criterion for comparison, we choose the sums of squares of the differences between the actual cumulative number of errors \( y_k \) detected by testing time \( t_k \) \( (k = 1, 2, \ldots, n) \) and its estimated value \( \hat{m}(t_k) \):

\[
SSE = \sum_{k=1}^{n} [y_k - \hat{m}(t_k)]^2.
\]

Table 1 shows the results of comparison. From Table 1, we find that the NHPP models with mean value functions \( m_a(t), m_b(t), \) and \( m_c(t) \) fit to the actual data sets better than the exponential and delayed S-shaped software reliability growth models based on an NHPP, i.e., the NHPP model with \( m_b(t) \) fits best to DS3, and the NHPP with \( m_c(t) \) to DS2.
5. CONCLUSION

In this paper, we have discussed three kinds of software reliability growth models based on the testing-domain, which describe the time-dependent behavior of influence of executing test-cases and skill of test personnel. Also, numerical examples of software reliability analysis for the actual data have been presented and the comparison among the models shown. From the results of comparison, we have found that the time-dependent behaviors of error-detection are closely related to that of testing-domain during the software testing phase.
Table 1. A Summary of goodness-of-fit with respect to the sum of square errors.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>(m_a^{(1)})</th>
<th>(m_b^{(1)})</th>
<th>(m_c^{(1)})</th>
<th>exponential</th>
<th>delayed</th>
<th>S-shaped</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS1</td>
<td>14579</td>
<td>14579</td>
<td>14579</td>
<td></td>
<td>13354</td>
<td>245071</td>
</tr>
<tr>
<td>DS2</td>
<td>147667</td>
<td>147667</td>
<td>63930</td>
<td></td>
<td>453069</td>
<td>147704</td>
</tr>
<tr>
<td>DS3</td>
<td>3499</td>
<td>2481</td>
<td>3559</td>
<td>4220</td>
<td></td>
<td>3588</td>
</tr>
</tbody>
</table>

In the future, we are planning to investigate the relationship of the testing-domain and the quality and quantity of executed test-cases.

REFERENCES