Relationship between Software Availability Measurement and the Number of Restorations with Imperfect Debugging

K. TOKUNO AND S. YAMADA
Department of Social Systems Engineering, Faculty of Engineering
Tottori University
4 101, Koyama, Tottori chi, 680 8552, Japan
<toku><yamada>@esse.tottori-u.ac.jp

Abstract—In this paper, we propose a software availability model considering the number of restoration actions. We correlate the failure and restoration characteristics of the software system with the cumulative number of corrected faults. Furthermore, we consider an imperfect debugging environment where the detected faults are not always corrected and removed from the system. The time-dependent behavior of the system alternating between up and down states is described by a Markov process. From this model, we can derive quantitative measures for software availability assessment based on the number of restoration actions. Finally, we show numerical examples of software availability analysis. © 2003 Elsevier Ltd. All rights reserved.

Keywords—Software availability, Imperfect debugging, Software reliability growth, Markov process, Quantitative assessment.

1. INTRODUCTION

Many methodologies for software reliability measurement and assessment have been discussed for the last few decades [1–4]. A mathematical software reliability model is often called a software reliability growth model (SRGM); this describes a software fault-detection or a software failure-occurrence phenomenon during the testing phase of software development process and the operation phase. A software failure is defined as an unacceptable departure from program operation caused by a fault remaining in the software system. This model is available for measuring and assessing the degree of achievement of software reliability, deciding the time to software release for operational use, and estimating the maintenance cost for faults undetected during the testing phase.

Most of SRGMs so far provide quantitative software reliability measures for developers. However, it begins to be necessary to assess software systems from the viewpoint of customers. In particular, recent systems are required nonstop operation and utilities. One of the customer-oriented attributes is software availability [5–7]; this is defined as the attribute that the software...
systems are performing at a given time point, according to the specification, under the specified environment. In other words, it represents the property that the systems are in available states whenever the customers want to use them. Few mathematical models for evaluating software availability are proposed.

In this paper, we construct a software availability model. Quantitative measures on reliability derived from previous SRGMs, such as the mean time between software failures and the software reliability representing the probability that the system can continue to operate for a given time period, are often provided as the functions of the number of software failures or fault detections, and useful for seeking the relationship between the number of detected faults and software reliability growth. On the other hand, there are scarcely software availability measures for explicitly understanding the relation with the number of restoration actions. Here, we discuss software availability measurement considering the number of restoration actions. The time-dependent behavior of the software system is described by a Markov process [8]. The software failure and the restoration characteristics are correlated with the cumulative number of corrected faults. Furthermore, we also describe the imperfect debugging environment where the debugging activities are not always performed for certain [9]. The assumptions and modeling are detailed in Section 2. Derivation of the stochastic quantities for software availability measurement is presented in Section 3. Numerical illustrations of software availability analysis are shown in Section 4. Finally, concluding remarks of this paper are summarized in Section 5.

2. MODEL DESCRIPTION

The following assumptions are made for software availability modeling.

(A1) The software system is unavailable and starts to be restored as soon as a software failure occurs, and the system cannot operate until the restoration action is complete.

(A2) The restoration action implies the debugging activity; this is performed perfectly with probability \( a_n \) (\( 0 < a_n \leq 1 \)) and imperfectly with probability \( b_n \) (\( = 1 - a_n \)). We call \( a_n \) the perfect debugging rate. \( a_n \) is a decreasing function of \( n \) [9]. One fault is corrected and removed when a debugging activity is perfect.

(A3) The next time intervals of software failures and restorations when \( n \) faults have already been corrected from the system follow exponential distributions with means \( 1/\lambda_n \) and \( 1/\mu_n \), respectively.

(A4) The probability that two or more software failures occur simultaneously is negligible.

Consider a stochastic process \( \{X(t), \ t \geq 0\} \) whose state space is \( (W, R) \), where up state vector \( W = \{W_n; \ n = 0, 1, 2, \ldots\} \) and down state vector \( R = \{R_n; \ n = 0, 1, 2, \ldots\} \) [10]. Then the events \( \{X(t) = W_n\} \) and \( \{X(t) = R_n\} \) mean that the system is operating and inoperable due to the restoration action at time point \( t \), when \( n \) faults have already been corrected, respectively.

From Assumption (A2), if the restoration action has been complete in \( \{X(t) = R_n\} \), then

\[
X(t) = \begin{cases} 
W_n, & \text{with probability } b_n, \\
W_{n+1}, & \text{with probability } a_n.
\end{cases}
\]

(1)

In general, the faults detected later tend to have higher complexity [11]. That is, the certainty of debugging becomes smaller with progress of debugging. For instance, we may describe the perfect debugging rate \( a_n \) as

\[
a_n = \nu w^n + \alpha, \quad n = 0, 1, 2, \ldots; \quad 0 < \nu, \ w, \ \alpha, \ \nu + \alpha \leq 1,
\]

(2)

where \( \nu + \alpha, \ w, \) and \( \alpha \) mean the initial perfect debugging rate, the decreasing ratio of the perfect debugging rate, and the stationary perfect debugging rate, respectively. In the special case of \( \nu + \alpha = \nu = 1 \), equation (2) describes the perfect debugging environment where any faults can be removed and the hazard rate always decreases when a debugging is performed.
We use Moranda's model [12] to describe the software failure-occurrence phenomenon, i.e., when \( n \) faults have been corrected, the hazard rate \( \lambda_n \) is given by

\[
\lambda_n = Dk^n, \quad n = 0, 1, 2, \ldots; \quad D > 0, \quad 0 < k < 1,
\]

where \( D \) and \( k \) are the initial hazard rate and the decreasing ratio of the hazard rate, respectively. The expression of (3) comes from the viewpoint that software reliability depends on the debugging efforts, not the residual fault content. We do not note how many faults remain in the software system. Equation (3) describes a software reliability growth process where earlier perfect debugging activities have larger impact on software reliability growth than later ones [4,9].

Early software availability models such as those of Okumoto and Goel [13] and Kim et al. [14] often assume that the hazard rate is proportional to the residual fault content and decreases by a constant amount with the perfect debugging, i.e., \( \lambda_n \) is described as

\[
\lambda_n = \phi(N - n), \quad n = 0, 1, 2, \ldots, N - 1; \quad N > 0, \quad \phi > 0,
\]

where \( N \) and \( \phi \) are the initial fault content and the hazard rate per fault remaining in the system, respectively [15].

Next, we describe the time-dependent behavior of the restoration action. The restoration action for software systems includes not only the data recovery and the program reload, but also the debugging activities for manifested faults. There often exist the cases where the later restoration actions tend to take longer times to isolate the positions of the faults and to check the fault corrections [1,11]. Describing such situations, we express the restoration rate \( \mu_n \) as follows:

\[
\mu_n = Er^n, \quad n = 0, 1, 2, \ldots; \quad E > 0, \quad 0 < r < 1.
\]

where \( E \) and \( r \) are the initial restoration rate and the decreasing ratio of the restoration rate, respectively. In the case of \( r = 1 \), \( \mu_n = E \) being constant, we can give the interpretation that any faults are homogeneous from the viewpoint of difficulty in debugging [10].

Let \( Q_{A,B}(\tau) \ (A, B \in (W, R)) \) denote the one-step transition probability that after making a transition into state \( A \), the process \( \{X(t), t \geq 0\} \) makes a transition into state \( B \) by time \( \tau \). The expressions for \( Q_{A,B}(\tau) \)'s are given as follows:

\[
Q_{W_n,R_n}(\tau) = 1 - e^{-\lambda_n \tau},
\]

\[
Q_{R_n,W_{n+1}}(\tau) = a_n \left(1 - e^{-\mu_n \tau}\right),
\]

\[
Q_{R_n,W_{\infty}}(\tau) = b_n \left(1 - e^{-\gamma_n \tau}\right).
\]

Figure 1 illustrates the sample state transition diagram of \( X(t) \).

![Figure 1. A diagrammatic representation of state transitions between \( X(t) \)s.](image-url)
3. DERIVATION OF SOFTWARE AVAILABILITY MEASURES

3.1. Distribution of Transition Time between up States

Let \( S_{i,n} \) \((i \leq n)\) be the random variable representing the transition time from state \( W_i \) to state \( W_n \), and \( G_{i,n}(t) \) be the distribution function of \( S_{i,n} \), respectively. Then we obtain the following renewal equation with respect to \( G_{i,n}(t) \):

\[
G_{i,n}(t) = Q_{W_i,R_i} * G_{i+1,n}(t) + Q_{W_i,R_i} * G_{i+1,n}(t), \tag{9}
\]

where \( * \) denotes a Stieltjes convolution and \( G_{n,n}(t) = 1(t) \) \((\text{step function}, n = 0, 1, 2, \ldots)\).

By applying the Laplace-Stieltjes (L-S) transform \([8]\) to (9), we can obtain the distribution function of \( S_{i,n} \):

\[
G_{i,n}(t) \equiv \Pr\{S_{i,n} \leq t\} = 1 - \sum_{m=1}^{n-1} \left[ A_{i,n}^1(m)e^{-x_m t} + A_{i,n}^2(m)e^{-y_m t} \right], \quad n = 1, 2, \ldots; \quad i = 0, 1, 2, \ldots, n, \tag{10}
\]

where

\[
x_i = \frac{1}{2} \left[ (\lambda_i + \mu_i) \pm \sqrt{(\lambda_i + \mu_i)^2 - 4a_i\lambda_i\mu_i} \right] \quad \text{(double signs in same order)}, \tag{11}
\]

and constant coefficients \( A_{i,n}^1(m) \) and \( A_{i,n}^2(m) \) are given by

\[
A_{i,n}^1(m) = \frac{\prod_{j=1}^{n-1} x_j y_j}{x_m \prod_{j \neq m} (x_j - x_m) \prod_{j=1}^{n-1} (y_j - x_m)}, \quad m = i, i + 1, \ldots, n - 1, \tag{12}
\]

\[
A_{i,n}^2(m) = \frac{\prod_{j=1}^{n-1} x_j y_j}{y_m \prod_{j \neq m} (y_j - y_m) \prod_{j=1}^{n-1} (x_j - y_m)}, \quad m = i, i + 1, \ldots, n - 1, \tag{13}
\]

respectively. It is noted that

\[
\sum_{m=1}^{n-1} \left[ A_{i,n}^1(m) + A_{i,n}^2(m) \right] = 1. \tag{14}
\]

Furthermore, the expectation and the variance of \( S_{i,n} \) are given by

\[
\mathbb{E}[S_{i,n}] = \sum_{m=1}^{n-1} \left( \frac{1}{x_m} + \frac{1}{y_m} \right), \tag{15}
\]

\[
\text{Var}[S_{i,n}] = \sum_{m=1}^{n-1} \left( \frac{1}{x_m^2} + \frac{1}{y_m^2} \right), \tag{16}
\]

respectively.

3.2. State Occupancy Probability

Let \( P_{A,B}(t) \) \((A, B \in (W, R))\) be the state occupancy probability that the system is in state \( B \) at time point \( t \) on the condition that the system was in state \( A \) at time point \( t = 0 \), i.e.,

\[
P_{A,B}(t) = \Pr\{X(t) = B | X(0) = A\}, \quad A, B \in (W, R). \tag{17}
\]
We obtain the following renewal equations with respect to $P_{W_i,w_n}(t)$:

\[ P_{W_i,w_n}(t) = G_{i,n} \ast P_{w_n,w_n}(t), \]
\[ P_{w_n,w_n}(t) = e^{-\lambda_n t} + Q_{W_n,R_n} \ast Q_{R_n,W_n} \ast P_{w_n,w_n}(t). \]

Substituting the L-S transform of (19) into that of (18) yields

\[ \tilde{P}_{W_i,w_n}(s) = \frac{s \tilde{G}_{i,n+1}(s)}{a_n \lambda_n} + \frac{s^2 \tilde{G}_{i,n+1}(s)}{a_n \lambda_n \mu_n}. \]

(20)

By inverting (20), $P_{W_i,w_n}(t)$ is obtained as

\[ P_{W_i,w_n}(t) = \Pr\{X(t) = W_n \mid X(0) = W_i\} = \frac{g_{i,n+1}(t)}{a_n \lambda_n} + \frac{g'_{i,n+1}(t)}{a_n \lambda_n \mu_n}, \]

(21)

where $g_{i,n}(t)$ is the probability density function associated with $G_{i,n}(t)$ and $g'_{i,n}(t) \equiv \frac{d g_{i,n}(t)}{dt}$.

Similarly, we obtain the following renewal equations with respect to $P_{W_i,R_n}(t)$:

\[ P_{W_i,R_n}(t) = G_{i,n} \ast Q_{w_n,R_n} \ast P_{R_n,R_n}(t), \]
\[ P_{R_n,R_n}(t) = e^{-\mu_n t} + Q_{R_n,w_n} \ast Q_{w_n,R_n} \ast P_{R_n,R_n}(t). \]

Substituting the L-S transform of (23) into that of (22) yields

\[ \tilde{P}_{W_i,R_n}(s) = \frac{s \tilde{G}_{i,n+1}(s)}{a_n \mu_n}. \]

(24)

By inverting (24), $P_{W_i,R_n}(t)$ is obtained as

\[ P_{W_i,R_n}(t) = \Pr\{X(t) = R_n \mid X(0) = W_i\} = \frac{g_{i,n+1}(t)}{a_n \mu_n}. \]

(25)

3.3. Software Availability

Once we specify integer $i$, the following equation holds for arbitrary time $t$:

\[ \sum_{n=i}^{\infty} [P_{W_i,w_n}(t) + P_{W_i,R_n}(t)] - 1. \]

(26)

Here we consider the relationship between the number of the restoration actions and software availability measurement. Let $l = 0, 1, 2, \ldots$ denote the number of the restoration actions. Furthermore, we introduce the binary indicator variable $I_A(t)$ taking the value 1 (0) if the system is operating (inoperable) at time point $t$, given that it was in state $A \in (W, R)$ at time point $t = 0$, respectively. Then $A_i(t) \equiv \Pr\{I_{W_i}(t) = 1\}$ (i = 0, 1, 2, \ldots) denotes the instantaneous software availability when the system was in state $W_i$ at time point $t = 0$, i.e.,

\[ A_i(t) = \sum_{n=i}^{\infty} P_{W_i,w_n}(t) \]
\[ = 1 - \sum_{n=i}^{\infty} P_{W_i,R_n}(t) \]

(27)
It is noted that the cumulative number of corrected faults at the completion of the $l$th restoration action, $C_l$, is not explicitly observed since imperfect debugging is assumed throughout this paper. However, the probability mass function $W_l(i) = \Pr\{C_l = i\}$ can be calculated by the following recursive formulae:

$$W_0(0) = 1,$$
$$W_l(i) = W_{l-1}(i-1)a_{l-1} + W_{l-1}(i)b_l,$$
$$l = 1, 2, \ldots; \quad i = 0, 1, 2, \ldots, l,$$

where we postulate $W_{l-1}(l) = W_{l-1}(0) = 0 \ (l = 1, 2, \ldots)$ and $a_{-1} = 0$. In particular, when $w = 1$ in (2), $C_l$ follows a binomial distribution with mean $l(v + \alpha)$. Accordingly, the instantaneous software availability after the completion of the $l$th restoration action is given by

$$A(t; l) = \sum_{i=0}^{l} \Pr\{C_l = i\} A_i(t),$$

which represents the probability that the system is operating at time point $t$ when the $l$th restoration action was complete at time point $t = 0$. Furthermore, the average software availability after the completion of the $l$th restoration action is given by

$$A_{av}(t; l) = \frac{1}{t} \int_{0}^{t} A(x; l) \, dx,$$

which represents the average proportion of the system's operating time to the time interval $(0, t]$ when the $l$th restoration action was complete at time point $t = 0$. Using (25), we can express (29) and (30) as

$$A(t; l) = 1 - \sum_{i=0}^{l} W_l(i) \sum_{n=i}^{\infty} \frac{g_{i,n+1}(t)}{a_n \mu_n},$$
$$A_{av}(t; l) = 1 - \frac{1}{t} \sum_{i=0}^{l} W_l(i) \sum_{n=i}^{\infty} \frac{G_{i,n+1}(t)}{a_n \mu_n},$$

respectively.

![Figure 2. Sample behavior of the system and event $[I_{W_l}(t) = 1]$.](image-url)
Using the software availability model discussed above, we show numerical illustrations for software availability measurement and assessment.

We define the maintenance factor as

\[ \rho_n \equiv \frac{\lambda}{\mu_n} = C v^n, \]  

where we call \( C \) and \( v \) the initial maintenance factor and the availability improvement parameter, respectively.

Figures 3 and 4 show the time-dependent behavior of the average software availability, \( A_{av}(t;l) \) in (32) for various numbers of the restoration actions, \( l \), in the cases of \( v < 1 \) and \( v > 1 \), respectively. Figures 3 and 4 illustrate that software availability increases and decreases with increase in the number of restorations, respectively. These suggest that \( v \) can be an index to determine whether software availability improves.
Figures 5 and 6 show the instantaneous software availability, $A(t; l)$ in (31) and $A_{av}(t; l)$ for various values of the decreasing ratio of the perfect debugging rate, $w$. These figures tell us that the software availability becomes higher as the perfect debugging rate becomes larger when $v < 1$. $A(t; l)$ and $A_{av}(t; l)$ have one minimum value in the case of $v < 1$. Then we can find the minimum number of restoration actions, $l_{min}$, satisfying that the minimum value of $A(t; l)$ or $A_{av}(t; l)$ exceeds the prespecified availability objective, $\theta$. Table 1 summarizes $l_{min}$s for $A(t; l)$ and $A_{av}(t; l)$ for various values of $w$, in the case of $\theta = 0.95$. As shown in Table 1, the higher certainty of debugging attains the objective of software availability earlier.

5. CONCLUDING REMARKS

In this paper, we have developed a stochastic model describing the relationship between the number of restoration actions and software availability measurement. We have used a Markov process for the description of the behavior of the system alternating between operable and inoperable states. We have given the instantaneous and the average software availability as the functions of the number of restoration actions. Numerical illustrations for software availability
measurement have also been presented to show that these measures are very useful for software performance assessment. This model has been more generalized in terms of the imperfect debugging and the upwardness of difficulty in debugging than several previous models.

The unknown parameters must be estimated based on the actual data for assessing software availability with this model. But it is difficult to observe and collect the testing or the field data. In particular, it is necessary to equip the collection procedure of the restoration times. The establishment of practical estimation of the model parameters remains a future study.

### REFERENCES


---

**Table 1**: $t_{\text{min}}$ for $A(t; t)$ and $A_{\text{av}}(t; t)$ ($\theta = 0.95; \psi = 0.6, \alpha = 0.4, D = 0.1, k = 0.8, E = 1.0, r = 0.9$).

<table>
<thead>
<tr>
<th>$w$</th>
<th>$t_{\text{min}}$ for $A(t; t)$</th>
<th>$t_{\text{min}}$ for $A_{\text{av}}(t; t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>0.9</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>0.8</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>0.7</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>0.6</td>
<td>9</td>
<td>8</td>
</tr>
</tbody>
</table>