Journal of the Operations Research
Society of Japan
2007, Vol. 50, No. 4, 444-462

USER-ORIENTED AND -PERCEIVED SOFTWARE AVAILABILITY MEASUREMENT AND ASSESSMENT WITH ENVIRONMENTAL FACTORS

Koichi Tokuno Shigeru Yamada
Tottori University

(Received October 30, 2006; Revised May 15, 2007)

Abstract This paper proposes the methods of the operation-oriented software availability measurement and assessment. Considering the difference of the software failure-occurrence phenomena and the restoration characteristics between the testing and the operation phases, we first introduce the environmental factors into the existing Markovian software availability model. Next we discuss the stochastic modeling for measuring software service availability; this is one of the customer-oriented attribute and defined as the attribute that the software system can successfully satisfy the end users' requests. We derive several service-oriented software availability assessment measures which are given as the functions of time and the number of debuggings. Finally, we present several numerical examples of these measures for software service availability analysis.

Keywords: Reliability, software service availability, operational phase, Markov process, environmental factor, software reliability growth

1. Introduction

Today it has been increasingly important to evaluate not only the inherent quality characteristics of the artificial industrial products but also the quality of service created by the use of the products. Recently, the engineering system of "service engineering" has been suggested [1, 8]. In the traditional engineering, only the inherent functions, performance, and quality of the industrial products have been discussed and designed from the developer's logic. On the other hand, the service engineering aims to establish the comprehensive methodologies for evaluating the functions or quality in consideration of the behaviors and the satisfaction of the end users as well. In other words, the service engineering pays attention to the evaluation of the services the end users receive through the operation of the industrial products. As to the **service reliability engineering** based on the above philosophy of the service engineering, Tortorella [17, 18] has discussed the meaning and the possibility of practical use of the service reliability engineering. Considering the software systems are just the industrial products to provide the services for the users, especially in computer network systems, it is meaningful to organize the user-oriented and -perceived software service reliability/availability modeling. To develop the computing framework and the interface between hardware and software systems with high service availability, the Service Availability Forum (SAF) has been created [20]. The members of SAF include the leading communication and computing companies.

In this paper, we discuss the stochastic modeling for the user-oriented and -perceived software availability measurement and assessment. Software availability is defined as the attribute that the software system is available whenever the end users want to use; this is one of the customer- and operation-oriented attributes. Studies on stochastic software availability measurement and assessment have been conducted [15]. In particular, we aim at the following:

1. the difference of the operational environments between the testing and the user operation phases,
2. consideration of the behavior of the end user.

When we conduct the software quality evaluation with stochastic software reliability or availability models, we generally consider that the software failure-occurrence phenomenon during the testing phase are the same property as the user operation phase. In other words, we implicitly assume that the software reliability growth curves (e.g. the mean value functions in nonhomogeneous Poisson process (NHPP) models) fitted to the testing data also describe the quality characteristics in the operation phase. However, there also exist the negative opinions against the above mention. That is, the impact of the faults latent in the system on software quality depends on usage environment since the testing and the user operation phases differ in terms of workloads, interaction between software and hardware platforms, and the operational profile [6]. Accordingly, it is important to consider the difference between the testing and the user operation environments in evaluating software availability.

Here we characterize the difference between the testing and the operational environments, assuming that the time scale of the testing phase is proportional to that of the operation phase in terms of the software failure-occurrence and the restoration action. We call the time-scale transformation ratio the environmental factor. This idea is based on the accelerated life testing model [2, 10] in hardware products, and we apply the idea to the software availability model. We use the Markovian software availability model [13] to describe the time-dependent behavior of the system alternating between up and down states.

Furthermore, existing software availability models often pay attention to the stochastic behaviors of only software systems themselves. However, from the viewpoint of end users, the traditional software availability measures such as the instantaneous software availability and the interval software reliability are not always appropriate. It will be enough for end users if the system is available only when usage demands occur. In other words, the users do not care about the state of the system, even if the system is down, when the users do not want to use it. Gaver [3] has defined the disappointment time as the time to a failure during a usage period, or to occurrence of a usage demand during a system inoperable period, whichever occurs first, and derived the Laplace-Stieltjes transform of the distribution of this time. Osaki [11] has discussed the disappointment time of a two-unit standby redundant system when it is used intermittently. In this paper, we also discuss the software availability model incorporating the usage behavior of the end user.

Recently, studies on service availability have been taken a growing interest, however, the definition of service availability is still not authorized. We mention several existing definitions and studies on service or user-perceived availability as follows. For the transaction processing systems, Mainkar [7] has considered the probability that the response time of a transaction is less than a given deadline (i.e., the distribution of the response time) and defined the user-perceived availability as the probability that the value of the distribution exceeds a prespecified value. Kaâniche et al. [5] have presented a hierarchical availability modeling framework for a web-based system and discussed the user-perceived availability measures in the steady state. The measures they consider have involved the impact of the performance-related and the inherent failures. Wang and Trivedi [19] have interpreted the user-perceived service availability as the probability that all of user's requests are successfully satisfied during the user session and shown the service availability measures in the steady state in the case where a single user has one or multiple requests in user session. Furthermore, they have computed the service availability in a voice over IP system, using stochastic reward nets to describe the user and the system behaviors.

Here we discuss the service availability modeling for software systems, based on the definition of Wang and Trivedi. That is, we define **software service availability** as the attribute that the software system can successfully complete the services the users request. We assume the situation where an end user intermittently uses the system operating and available anytime. For example, in the software system controlling the mobile communication system, all of the system is working at all time but each of end users uses the system intermittently. Existing studies [5, 7, 19] have often derived the service availability measures in the steady state, whereas we derive the transient solutions of the software service availability measures since we consider the dynamic reliability growth and restoration characteristics of the software systems. In particular, we define the following new measures: (i) the software service availability in use defined as the probability that the user's requests are successfully complete before a software failure occurs, (ii) the software service unavailability due to request cancellation defined as the probability that the system is restored and the user's request is canceled due to the restoration action, and (iii) the software service unavailability under restoration defined as the probability that the user's requests occur before a restoration action is complete when the system is restored.

The organization of the rest of this paper is as follows: Section 2 gives a brief explanation of the Markovian software availability model underlying the discussion in this paper. Section 3 proposes the operational software availability assessment method introducing the environmental factors and derives several quantitative software availability assessment measures for the operation phase. Section 4 discusses the model with the end user's behavior based on the model of Section 3. The measures derived in the respective sections are given as the functions of the time and the number of debugging activities. Section 5 illustrates several numerical examples of software availability analysis based on the model. Finally, Section 6 summarizes the results obtained in this paper.

2. Basic Markovian Software Availability Model [13]

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 includes the debugging activity; this is performed perfectly with the perfect debugging rate a $(0 < a \leq 1)$ and imperfectly with probability $b(= 1 - a)$. One fault is corrected and removed from the software system when the debugging activity is perfect.

A3. The next software failure time, Z_n, and the restoration time, T_n, when n faults have already been corrected from the system, follow the exponential distributions with the following distribution functions:

$$F_{Z_n}(t) \equiv \Pr\{Z_n \leq t\} = 1 - e^{-\lambda_n t}, \tag{1}$$

$$F_{T_n}(t) \equiv \Pr\{T_n \leq t\} = 1 - e^{-\mu_n t}, \tag{2}$$

respectively. λ_n and μ_n are non-increasing functions of n.

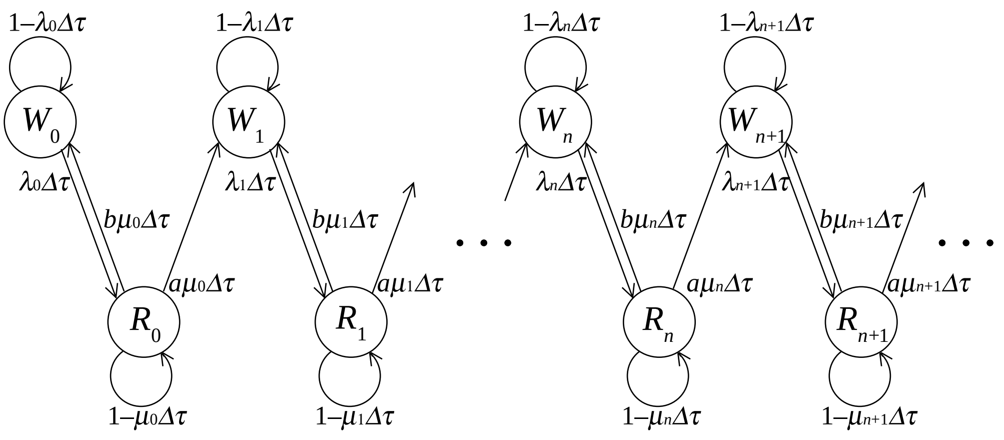

Figure 1: A sample state transition diagram of $X(t)$ for basic model

Let $\{X(t),\ t \geq 0\}$ be the stochastic process representing the state of the software system at the time point t. The state space vector $(\boldsymbol{W},\ \boldsymbol{R})$ of $\{X(t),\ t \geq 0\}$ is denoted as

$\boldsymbol{W} = \{W_n;\ n = 0,\ 1,\ 2,\ \ldots\}$: the system is operating,

$\boldsymbol{R} = \{R_n;\ n = 0,\ 1,\ 2,\ \ldots\}$: the system is inoperable and under restoration,

where n denotes the cumulative number of corrected faults. Figure 1 illustrates the sample state transition diagram of $X(t)$.

For obtaining the software availability measures, we first consider the random variable $S_{i,n}$ representing the transition time of $X(t)$ from state W_i to state W_n $(i \leq n)$. Then the distribution function of $S_{i,n}$, $G_{i,n}(t)$, is given by

$$\left.\begin{array}{l} G_{i,n}(t) \equiv \Pr\{S_{i,n} \leq t\} = 1 - \displaystyle\sum_{m=i}^{n-1} \left[A^1_{i,n}(m) e^{-d^1_m t} + A^2_{i,n}(m) e^{-d^2_m t}\right] \\ \qquad (i, n = 0, 1, 2, \ldots;\ i \leq n) \\ \left(\begin{array}{l} \left.\begin{array}{r} d^1_i \\ d^2_i \end{array}\right\} = \dfrac{1}{2}\left[(\lambda_i + \mu_i) \pm \sqrt{(\lambda_i + \mu_i)^2 - 4a\lambda_i\mu_i}\right] \text{ (double signs in same order)} \\ A^1_{i,n}(m) = \dfrac{\displaystyle\prod_{j=i}^{n-1} d^1_j d^2_j}{d^1_m \displaystyle\prod_{\substack{j=i \\ j \neq m}}^{n-1} (d^1_j - d^1_m) \prod_{j=i}^{n-1} (d^2_j - d^1_m)} \quad (m = i,\ i+1,\ \ldots,\ n-1) \\ A^2_{i,n}(m) = \dfrac{\displaystyle\prod_{j=i}^{n-1} d^1_j d^2_j}{d^2_m \displaystyle\prod_{\substack{j=i \\ j \neq m}}^{n-1} (d^2_j - d^2_m) \prod_{j=i}^{n-1} (d^1_j - d^2_m)} \quad (m = i,\ i+1,\ \ldots,\ n-1) \end{array}\right) \end{array}\right\}. \tag{3}$$

Equation (3) is obtained by solving the following renewal equation:

$$G_{i,n}(t) = Q_{W_i,R_i} * Q_{R_i,W_{i+1}} * G_{i+1,n}(t) + Q_{W_i,R_i} * Q_{R_i,W_i} * G_{i,n}(t) \\ (i = 0,\ 1,\ 2,\ \ldots,\ n-1), \tag{4}$$

where $*$ denotes the Stieltjes convolution and $Q_{A,B}(t)$'s $(A,\ B \in \{\boldsymbol{W},\ \boldsymbol{R}\})$ denote the one-step transition probability between state A and state B. We apply the Laplace-Stieltjes (L-S) transforms to solve Equation (4) [12].

Next we derive the state occupancy probabilities, $P_{A,B}(t) \equiv \Pr\{X(t) = B|X(0) = A\}$ $(A,\ B \in \{\boldsymbol{W},\ \boldsymbol{R}\})$. The renewal equation of $P_{W_i,W_n}(t)$ is obtained as follows:

$$\left.\begin{aligned} P_{W_i,W_n}(t) &= G_{i,n} * P_{W_n,W_n}(t) \\ P_{W_n,W_n}(t) &= e^{-\lambda_n t} + Q_{W_n,R_n} * Q_{R_n,W_n} * P_{W_n,W_n}(t) \end{aligned}\right\}. \tag{5}$$

Solving Equation (5), we have $P_{W_i,W_n}(t)$ as

$$P_{W_i,W_n}(t) \equiv \Pr\{X(t) = W_n|X(0) = W_i\} = \frac{g_{i,n+1}(t)}{a\lambda_n} + \frac{g'_{i,n+1}(t)}{a\lambda_n\mu_n}, \tag{6}$$

where $g_{i,n}(t) \equiv \mathrm{d}G_{i,n}(t)/\mathrm{d}t$ denotes the density function of $S_{i,n}$ and $g'_{i,n}(t) \equiv \mathrm{d}g_{i,n}(t)/\mathrm{d}t$.

Similarly, we have $P_{W_i,R_n}(t)$ as

$$P_{W_i,R_n}(t) \equiv \Pr\{X(t) = R_n|X(0) = W_i\} = \frac{g_{i,n+1}(t)}{a\mu_n}, \tag{7}$$

where Equation (7) is the solution of the following renewal equation:

$$\left.\begin{aligned} P_{W_i,R_n}(t) &= G_{i,n} * Q_{W_n,R_n} * P_{R_n,R_n}(t) \\ P_{R_n,R_n}(t) &= e^{-\mu_n t} + Q_{R_n,W_n} * Q_{W_n,R_n} * P_{R_n,R_n}(t) \end{aligned}\right\}. \tag{8}$$

Based on the above analyses, we can obtain the instantaneous software availability and the average software availability as

$$\begin{aligned} A(t;l) &\equiv \sum_{i=0}^{l} \binom{l}{i} a^i b^{l-i} \sum_{n=i}^{\infty} \Pr\{X(t) = W_n|X(0) = W_i\} \\ &= 1 - \sum_{i=0}^{l} \binom{l}{i} a^i b^{l-i} \sum_{n=i}^{\infty} \frac{g_{i,n+1}(t)}{a\mu_n}, \end{aligned} \tag{9}$$

$$\begin{aligned} A_{av}(t;l) &\equiv \frac{1}{t}\int_0^t A(x;l)\mathrm{d}x \\ &= 1 - \frac{1}{t}\sum_{i=0}^{l} \binom{l}{i} a^i b^{l-i} \sum_{n=i}^{\infty} \frac{G_{i,n+1}(t)}{a\mu_n}, \end{aligned} \tag{10}$$

respectively, where $\binom{l}{i}a^i b^{l-i}$ denotes the probability that i faults are corrected at the completion of the l-th debugging $(l = 0,\ 1,\ 2,\ \ldots;\ i = 0,\ 1,\ 2,\ \ldots,\ l)$ and we use the equation $\sum_{n=i}^{\infty}[P_{W_i,W_n}(t) + P_{W_i,R_n}(t)] = 1$. Equations (9) and (10) represent the probability that the system is operating at the time point t and the expected proportion of the system's operating time to the time interval $(0,\ t]$, given that the l-th debugging was complete at time point $t = 0$, respectively.

Furthermore, the interval software reliability and the conditional mean available time [14] are given by

$$\begin{aligned} R_I(t,x;l) &\equiv \sum_{i=0}^{l} \binom{l}{i} a^i b^{l-i} \sum_{n=i}^{\infty} \Pr\{X(t) = W_n,\ Z_n > x|X(0) = W_i\} \\ &= \sum_{i=0}^{l} \binom{l}{i} a^i b^{l-i} \sum_{n=i}^{\infty} P_{W_i,W_n}(t) e^{-\lambda_n x}, \end{aligned} \tag{11}$$

$$MAT(t;l) \equiv \sum_{i=0}^{l} \binom{l}{i} a^i b^{l-i} \frac{\sum_{n=i}^{\infty} \mathrm{E}[Z_n] \cdot \Pr\{X(t)=W_n | X(0)=W_i\}}{\Pr\{\text{system is up at time point } t | X(0)=W_i\}}$$
$$= \sum_{i=0}^{l} \binom{l}{i} a^i b^{l-i} \frac{\sum_{n=i}^{\infty} P_{W_i,W_n}(t)/\lambda_n}{1-\sum_{n=i}^{\infty} g_{i,n+1}(t)/(a\mu_n)}, \tag{12}$$

respectively, Equations (11) and (12) represent the probability that the system is operable at the time point t and will continue to be available for the time interval $(t,\ t+x]$ and the expected available time interval on the condition that the system is operating at the time point t, given that the l-th debugging was complete at time point $t=0$, respectively.

3. Operational Software Availability Assessment

Hereafter, let the notations with and without superscript ^{O} denote ones associated with the operation phase and the testing phase, respectively. For example, Z_i^O and Z_i denote the random variables representing the next software failure time in the operation phase and the testing phase when i faults have already been corrected, respectively.

We assume the following relationships between Z_i and Z_i^O, and T_i and T_i^O:

$$Z_i^O = \alpha Z_i, \quad F_{Z_i^O}(t) = F_{Z_i}(t/\alpha) \qquad (\alpha > 0), \tag{13}$$
$$T_i^O = \beta T_i, \quad F_{T_i^O}(t) = F_{T_i}(t/\beta) \qquad (\beta > 0), \tag{14}$$

where we call α and β the environmental factors. From the viewpoint of the software reliability assessment, $\alpha > 1$ $(0 < \alpha < 1)$ reflects the situation where the operation phase is milder (severer) in the usage condition than the testing phase, and $\beta > 1$ $(0 < \beta < 1)$ reflects the situation where the restoration time in the operation phase is apt to be longer (shorter) than that in the testing phase. The case of $\alpha = \beta = 1$ means that the operational environment is equivalent to the testing one.

Using Equations (13) and (14), we can obtain the distribution of $S_{i,n}^O$ as

$$\left.\begin{array}{l}
G_{i,n}^O(t) \equiv \Pr\{S_{i,n}^O \le t\} = 1 - \sum_{m=i}^{n-1} \left[A_{i,n}^{1O}(m) e^{-d_m^{1O} t} + A_{i,n}^{2O}(m) e^{-d_m^{2O} t}\right] \\
\qquad (i,n = 0,1,2,\ldots;\ i \le n) \\
\left(\begin{array}{l}
\lambda_i^O = \lambda_i/\alpha,\ \mu_i^O = \mu_i/\beta \\
\left.\begin{array}{l} d_i^{1O} \\ d_i^{2O} \end{array}\right\} = \frac{1}{2}\left[(\lambda_i^O + \mu_i^O) \pm \sqrt{(\lambda_i^O + \mu_i^O)^2 - 4a\lambda_i^O \mu_i^O}\right] \\
\qquad \text{(double signs in same order)} \\
A_{i,n}^{1O}(m) = \dfrac{\prod_{j=i}^{n-1} d_j^{1O} d_j^{2O}}{d_m^{1O} \prod_{\substack{j=i \\ j\ne m}}^{n-1} (d_j^{1O} - d_m^{1O}) \prod_{j=i}^{n-1} (d_j^{2O} - d_m^{1O})} \quad (m = i,\ i+1,\ \ldots,\ n-1) \\
A_{i,n}^{2O}(m) = \dfrac{\prod_{j=i}^{n-1} d_j^{1O} d_j^{2O}}{d_m^{2O} \prod_{\substack{j=i \\ j\ne m}}^{n-1} (d_j^{2O} - d_m^{2O}) \prod_{j=i}^{n-1} (d_j^{1O} - d_m^{2O})} \quad (m = i,\ i+1,\ \ldots,\ n-1)
\end{array}\right)
\end{array}\right\}. \tag{15}$$

Let $l = l_r$ be the number of debuggings performed before release. Then the instantaneous software availability and the average software availability in the operation phase are given by

$$A^O(t;l) = 1 - \sum_{i=0}^{l} \binom{l}{i} a^i b^{l-i} \sum_{n=i}^{\infty} \frac{\beta g^O_{i,n+1}(t)}{a\mu_n} \quad (l = l_r,\ l_r+1,\ l_r+2,\ \ldots), \tag{16}$$

$$A^O_{av}(t;l) = 1 - \frac{1}{t}\sum_{i=0}^{l} \binom{l}{i} a^i b^{l-i} \sum_{n=i}^{\infty} \frac{\beta G^O_{i,n+1}(t)}{a\mu_n} \quad (l = l_r,\ l_r+1,\ l_r+2,\ \ldots), \tag{17}$$

respectively. Equations (16) and (17) represent the probability that the system is operating at the time point t and the expected proportion of the system's operating time to the time interval $(0,\ t]$ in the operation phase, given that the system is release after the l_r-th debugging was complete in the testing phase, respectively.

Furthermore, the interval software reliability and the conditional mean available time in the operation phase are given by

$$R^O_I(t,x;l) = \sum_{i=0}^{l} \binom{l}{i} a^i b^{l-i} \sum_{n=i}^{\infty} P^O_{W_i,W_n}(t) e^{-\lambda_n x/\alpha} \quad (l = l_r,\ l_r+1,\ l_r+2,\ \ldots), \tag{18}$$

$$MAT^O(t;l) = \sum_{i=0}^{l} \binom{l}{i} a^i b^{l-i} \frac{\sum_{n=i}^{\infty} \alpha P^O_{W_i,W_n}(t)/\lambda_n}{1 - \sum_{n=i}^{\infty} \beta g^O_{i,n+1}(t)/(a\mu_n)} \quad (l = l_r,\ l_r+1,\ l_r+2,\ \ldots), \tag{19}$$

respectively. Equations (18) and (19) represent the probability that the system is operable at the time point t and will continue to be available for the time interval $(t,\ t+x]$ and the expected available time interval on the condition that the system is operating at the time point t in the operation phase, given that the system is release after the l_r-th debugging was complete in the testing phase, respectively.

4. Model with User Behavior

4.1. Model description

In the preceding section, we have discussed the software availability model considering only the time-dependent behavior of the system itself, i.e., only up and down states. In this section, we consider the user behavior as well and discuss the model for the software availability assessment from the viewpoint of the user. Here we assume that a user intermittently uses the system which is accessible anytime. We make the following additional assumptions for the software availability modeling:

A4. The system is released and transferred to the operation phase after $i(\geq 0)$ faults are corrected, and then the release point in time is set to the time origin $t = 0$. The user does not use the system at time point $t = 0$. The time to occurrence of a usage request, V_{ur}, and the usage time of the user, V_{ut}, follow the exponential distributions with means $1/\theta$ and $1/\eta$, respectively.

A5. In the cases where the software failure occurs when the user is using the system, or the usage request occurs under the system restoration, the corresponding usage request is canceled.

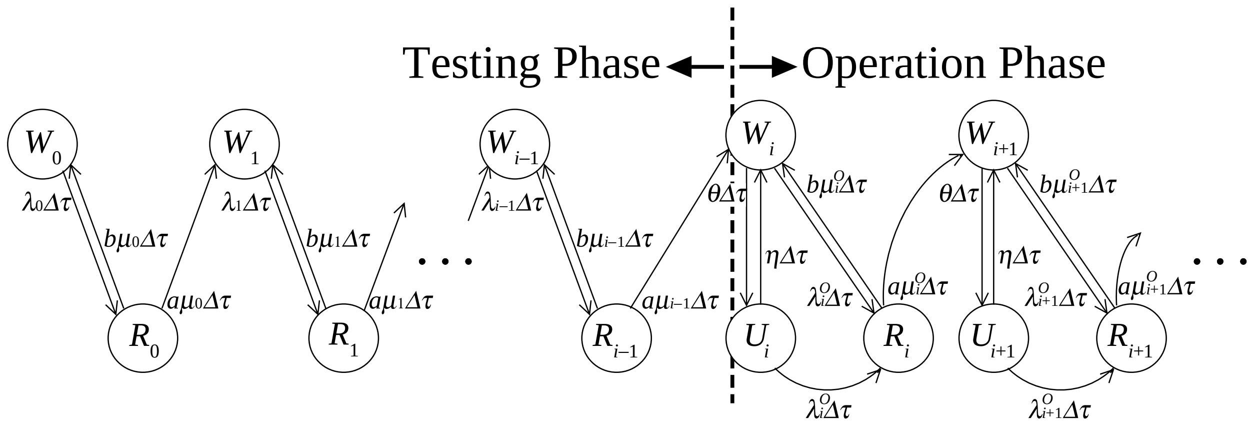

Figure 2: A sample state transition diagram of $X(t)$ for model with user behavior

Recall that $\{X(t),\ t \geq 0\}$ denotes the stochastic process representing the state of the system at the time point t during the operation phase. Then we redefine its state space vector $(\boldsymbol{W},\ \boldsymbol{U},\ \boldsymbol{R})$ as follows:

$\boldsymbol{W} = \{W_n;\ n = 0,\ 1,\ 2,\ \ldots\}$: the system is available but the user does not use the system,
$\boldsymbol{U} = \{U_n;\ n = 0,\ 1,\ 2,\ \ldots\}$: the system is available and the user is using the system,
$\boldsymbol{R} = \{R_n;\ n = 0,\ 1,\ 2,\ \ldots\}$: the system is restored due to a software failure-occurrence.

Figure 2 illustrates a sample state transition diagram of $X(t)$. From Figure 2, we have the following $Q^O_{A,B}(\tau)$'s:

$$Q^O_{W_n,U_n}(\tau) = \frac{\theta}{\lambda^O_n + \theta}\left[1 - e^{-(\lambda^O_n+\theta)\tau}\right], \tag{20}$$

$$Q^O_{W_n,R_n}(\tau) = \frac{\lambda^O_n}{\lambda^O_n + \theta}\left[1 - e^{-(\lambda^O_n+\theta)\tau}\right], \tag{21}$$

$$Q^O_{U_n,W_n}(\tau) = \frac{\eta}{\lambda^O_n + \eta}\left[1 - e^{-(\lambda^O_n+\eta)\tau}\right], \tag{22}$$

$$Q^O_{U_n,R_n}(\tau) = \frac{\lambda^O_n}{\lambda^O_n + \eta}\left[1 - e^{-(\lambda^O_n+\eta)\tau}\right], \tag{23}$$

$$Q^O_{R_n,W_{n+1}}(\tau) = a\left(1 - e^{-\mu^O_n\tau}\right), \tag{24}$$

$$Q^O_{R_n,W_n}(\tau) = b\left(1 - e^{-\mu^O_n\tau}\right). \tag{25}$$

4.2. Derivation of software service availability measures

4.2.1. Distribution of transition time between state $\boldsymbol{W}$

We have the following renewal equation of $G^O_{i,n}(t)$:

$$\left.\begin{aligned} G^O_{i,n}(t) &= L^O_{W_i,R_i} * Q^O_{R_i,W_{i+1}} * G^O_{i+1,n}(t) + L^O_{W_i,R_i} * Q^O_{R_i,W_i} * G^O_{i,n}(t) \\ &\quad (i = 0,\ 1,\ 2,\ \ldots,\ n-1) \\ L^O_{W_i,R_i}(t) &= Q^O_{W_i,R_i}(t) + Q^O_{W_i,U_i} * Q^O_{U_i,R_i}(t) + Q^O_{W_i,U_i} * Q^O_{U_i,W_i} * L^O_{W_i,R_i}(t) \end{aligned}\right\}. \tag{26}$$

The L-S transform of Equation (26) is given by

$$\widetilde{G}^O_{i,n}(s) = \prod_{m=i}^{n-1} \frac{d^{1O}_m d^{2O}_m}{(s + d^{1O}_m)(s + d^{2O}_m)}$$

$$= \sum_{m=i}^{n-1} \left(\frac{A_{i,n}^{1O}(m) d_m^{1O}}{s + d_m^{1O}} + \frac{A_{i,n}^{2O}(m) d_m^{2O}}{s + d_m^{2O}} \right). \tag{27}$$

By inverting Equation (27), we have the identical solution with Equation (15) and should note that $G_{i,n}^O(t)$ in Equation (26) has no bearing on the parameters θ and η.

4.2.2. State occupancy probability

We have the following renewal equation of $P_{W_i,W_n}^O(t)$:

$$\left.\begin{array}{l} P_{W_i,W_n}^O(t) = G_{i,n}^O * P_{W_n,W_n}^O(t) \\ P_{W_n,W_n}^O(t) = e^{-(\theta+\lambda_n^O)t} + Q_{W_n,R_n}^O * Q_{R_n,W_n}^O * P_{W_n,W_n}^O(t) + Q_{W_n,U_n}^O * Q_{U_n,W_n}^O * P_{W_n,W_n}^O(t) \\ \qquad\qquad + Q_{W_n,U_n}^O * Q_{U_n,R_n}^O * Q_{R_n,W_n}^O * P_{W_n,W_n}^O(t) \end{array}\right\}. \tag{28}$$

The L-S transform of $P_{W_i,W_n}^O(t)$ is obtained as

$$\tilde{P}_{W_i,W_n}^O(s) = \frac{s(s+\lambda_n^O+\eta)(s+\mu_n^O)}{(s+\lambda_n^O+\theta+\eta)(s+d_n^{1O})(s+d_n^{2O})} \cdot \prod_{m=i}^{n-1} \frac{d_m^{1O} d_m^{2O}}{(s+d_m^{1O})(s+d_m^{2O})}. \tag{29}$$

By inverting Equation (29), we obtain $P_{W_i,W_n}^O(t)$ as

$$\left.\begin{array}{l} P_{W_i,W_n}^O(t) \equiv \Pr\{X(t)=W_n | X(0)=W_i\} \\ \qquad = B_{i,n}^{0O} e^{-(\lambda_n^O+\theta+\eta)t} + \sum_{m=i}^{n} \left[B_{i,n}^{1O}(m) e^{-d_m^{1O}t} + B_{i,n}^{2O}(m) e^{-d_m^{2O}t} \right] \\ \qquad (i,n = 0,\ 1,\ 2,\ \ldots;\ i \le n), \\ \left(\begin{array}{ll} B_{i,n}^{0O} = \dfrac{-\theta(\mu_n^O - \lambda_n^O - \theta - \eta) \prod_{j=i}^{n-1} d_j^{1O} d_j^{2O}}{\prod_{j=i}^{n} (d_j^{1O} - \lambda_n^O - \theta - \eta)(d_j^{2O} - \lambda_n^O - \theta - \eta)} & (i,n = 0,1,2,\ldots;\ i \le n) \\ B_{i,n}^{1O}(m) = \dfrac{(\lambda_n^O + \eta - d_m^{1O})(\mu_n^O - d_m^{1O}) \prod_{j=i}^{n-1} d_j^{1O} d_j^{2O}}{(\lambda_n^O + \theta + \eta - d_m^{1O}) \prod_{\substack{j=i \\ j \ne m}}^{n} (d_j^{1O} - d_m^{1O}) \prod_{j=i}^{n} (d_j^{2O} - d_m^{1O})} & (m = i, i+1, \ldots, n) \\ B_{i,n}^{2O}(m) = \dfrac{(\lambda_n^O + \eta - d_m^{2O})(\mu_n^O - d_m^{2O}) \prod_{j=i}^{n-1} d_j^{1O} d_j^{2O}}{(\lambda_n^O + \theta + \eta - d_m^{2O}) \prod_{\substack{j=i \\ j \ne m}}^{n} (d_j^{2O} - d_m^{2O}) \prod_{j=i}^{n} (d_j^{1O} - d_m^{2O})} & (m = i, i+1, \ldots, n) \end{array} \right) \end{array}\right\}. \tag{30}$$

It should be noted that

$$\left.\begin{array}{ll} B_{i,i}^{0O} + B_{i,i}^{1O}(i) + B_{i,i}^{2O}(i) = 1 & (i = n) \\ B_{i,n}^{0O} + \sum_{m=i}^{n} \left[B_{i,n}^{1O}(m) + B_{i,n}^{2O}(m) \right] = 0 & (i < n) \end{array}\right\}. \tag{31}$$

Similarly, we have the following renewal equation of $P_{W_i,R_n}^O(t)$:

$$\left.\begin{array}{l} P_{W_i,R_n}^O(t) = G_{i,n}^O * L_{W_n,R_n}^O * P_{R_n,R_n}^O(t) \\ P_{R_n,R_n}^O(t) = e^{-\mu_n^O t} + Q_{R_n,W_n}^O * L_{W_n,R_n}^O * P_{R_n,R_n}^O(t) \end{array}\right\}, \tag{32}$$

where $L^O_{W_n,R_n}(t)$ has appeared in Equation (26). The L-S transform of $P^O_{W_i,R_n}(t)$ is obtained as

$$\begin{aligned}\widetilde{P}^O_{W_i,R_n}(s) &= \frac{s}{a\mu^O_n} \cdot \frac{a\lambda^O_n \mu^O_n}{(s+d^{1O}_n)(s+d^{2O}_n)} \cdot \widetilde{G}^O_{i,n}(s)\\ &= \frac{s}{a\mu^O_n}\widetilde{G}^O_{i,n+1}(s), \end{aligned} \tag{33}$$

where $d^{1O}_n d^{2O}_n = a\lambda^O_n\mu^O_n$ from Equation (15). By inverting Equation (33), we obtain $P^O_{W_i,R_n}(t)$ as

$$\begin{aligned}P^O_{W_i,R_n}(t) &\equiv \Pr\{X(t)=R_n|X(0)=W_i\}\\ &= \frac{g^O_{i,n+1}(t)}{a\mu^O_n} \quad (i,n=0,\ 1,\ 2,\ \ldots;\ i\le n),\end{aligned} \tag{34}$$

where $g^O_{i,n}(t)$ is the density function of $S^O_{i,n}$. We note that $P^O_{W_i,R_n}(t)$ has no bearing on the parameters θ and η.

Let $\{Y(t), t\ge 0\}$ be the counting process representing the cumulative number of faults corrected at the time point t. Then we have the following relationship:

$$\begin{aligned}&\{Y(t)=n|X(0)=W_i\} \Longleftrightarrow\\ &\{X(t)=W_n|X(0)=W_i\}\cup\{X(t)=R_n|X(0)=W_i\}\cup\{X(t)=U_n|X(0)=W_i\}\\ &\hfill (i\le n).\end{aligned} \tag{35}$$

Furthermore, the probability mass function of $\{Y(t), t\ge 0\}$ is given by

$$\Pr\{Y(t)=n|X(0)=W_i\} = G^O_{i,n}(t) - G^O_{i,n+1}(t). \tag{36}$$

Accordingly, $P^O_{W_i,U_n}(t)$ is given by

$$\begin{aligned}P^O_{W_i,U_n}(t) &\equiv \Pr\{X(t)=U_n|X(0)=W_i\}\\ &= G^O_{i,n}(t) - G^O_{i,n+1}(t) - P^O_{W_i,W_n}(t) - P^O_{W_i,R_n}(t)\ (i,n=0,\ 1,\ 2,\ \ldots;\ i\le n),\end{aligned} \tag{37}$$

since the events $\{X(t)=W_n|X(0)=W_i\}$, $\{X(t)=R_n|X(0)=W_i\}$, and $\{X(t)=U_n|X(0)=W_i\}$ are mutually exclusive.

4.2.3. Software service availability

In the discussion below, consider that i faults have already been corrected at time point $t=0$.

The probabilities that the system is in state $\boldsymbol{W}$, $\boldsymbol{U}$, and $\boldsymbol{R}$ at the time point t are given by

$$\Pr\{X(t)\in \boldsymbol{W}|X(0)=W_i\} \equiv \sum_{n=i}^{\infty} P^O_{W_i,W_n}(t), \tag{38}$$

$$\Pr\{X(t)\in \boldsymbol{U}|X(0)=W_i\} \equiv \sum_{n=i}^{\infty} P^O_{W_i,U_n}(t), \tag{39}$$

$$\Pr\{X(t)\in \boldsymbol{R}|X(0)=W_i\} \equiv \sum_{n=i}^{\infty} P^O_{W_i,R_n}(t), \tag{40}$$

respectively. Furthermore, the probabilities that the usage of the user can be complete without cancellation, i.e., the user's request is satisfied before a software failure occurs, and

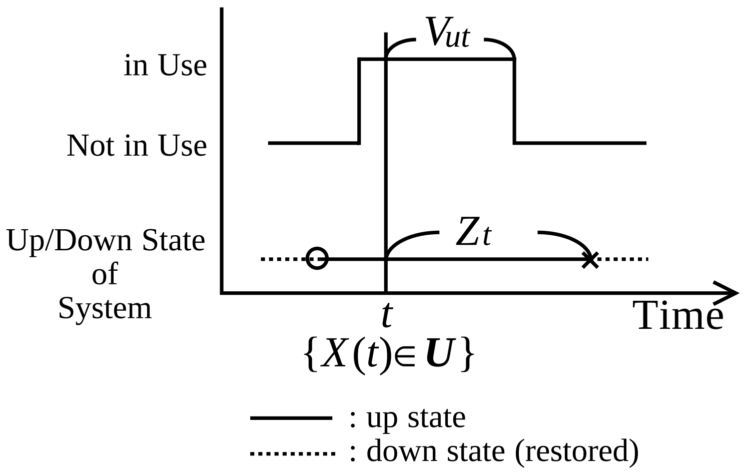

Figure 3: Example of completion of user request in use

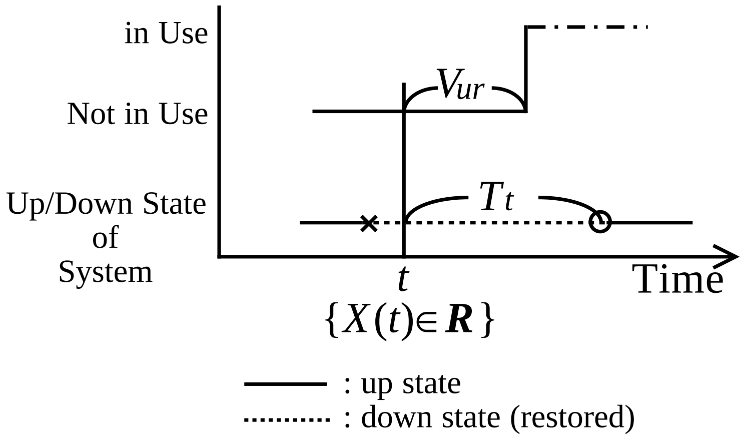

Figure 4: Example of user-perceived system failure under restoration

that the user's request occurs when the system is being restored, given that n faults have already been corrected, are given by

$$\Pr\{Z_n^O > V_{ut}\} = \frac{\eta}{\eta + \lambda_n^O}, \tag{41}$$

$$\Pr\{V_{ur} < T_n^O\} = \frac{\theta}{\theta + \mu_n^O}, \tag{42}$$

respectively.

Let Z_t be the random variable representing the software failure-occurrence time measured from the arbitrary time point t. **The software service availability in use** can be defined as the conditional probability that the user's requests are satisfied before a software failure occurs, provided the system is being used at the time point t (see Figure 3), and given by

$$\begin{aligned} SA_{u(i)}^O(t) &\equiv \Pr\{Z_t > V_{ut} | X(t) \in \boldsymbol{U}\} \\ &= \frac{\Pr\left\{\bigcup_{n=i}^{\infty} (Z_n^O > V_{ut}, X(t) = U_n)\right\}}{\Pr\{X(t) \in \boldsymbol{U} | X(0) = W_i\}} \end{aligned}$$

$$= \sum_{n=i}^{\infty} \frac{\eta P^O_{W_i,U_n}(t)}{\eta + \lambda^O_n} \Bigg/ \sum_{n=i}^{\infty} P^O_{W_i,U_n}(t). \tag{43}$$

On the other hand, let T_t be the random variable representing the restoration time measured from the arbitrary time point t. **The software service unavailability due to request cancellation** can be defined as the probability that the system is restored at the time point t and the user's request is canceled due to the corresponding restoration action (see Figure 4), and given by

$$\begin{aligned} SUA^O_{rc(i)}(t) &\equiv \Pr\{V_{ur} < T_t, X(t) \in \boldsymbol{R}\} \\ &= \Pr\left\{\bigcup_{n=i}^{\infty}(V_{ur} < T_n, X(t) = R_n)\right\} \\ &= \sum_{n=i}^{\infty} \frac{\theta P^O_{W_i,R_n}(t)}{\theta + \mu^O_n}. \end{aligned} \tag{44}$$

Furthermore, **The software service unavailability under restoration** can be defined as the conditional probability that the user's request is canceled, provided the system is being restored at the time point t, and given by

$$\begin{aligned} SUA^O_{r(i)}(t) &\equiv \Pr\{V_{ur} < T_t | X(t) \in \boldsymbol{R}\} \\ &= \sum_{n=i}^{\infty} \frac{\theta P^O_{W_i,R_n}(t)}{\theta + \mu^O_n} \Bigg/ \sum_{n=i}^{\infty} P^O_{W_i,R_n}(t). \end{aligned} \tag{45}$$

We should note that it is too difficult to use Equations (43), (44), and (45) directly as the software service availability measures. The reason is that the cumulative number of faults corrected at the time origin, i.e., integer i cannot be observed immediately since this model assumes the imperfect debugging environment. However, we can easily observe the number of debugging activities and the cumulative number of faults corrected after the completion of the l-th debugging, C_l, is distributed with the probability mass function $\Pr\{C_l = i\} = \binom{l}{i} a^i b^{l-i}$. Similar to Section 3, we can convert Equations (43), (44), and (45) into the functions of the number of debuggings, l, i.e., we can obtain

$$SA^O_u(t;l) = \sum_{i=0}^{l} \binom{l}{i} a^i b^{l-i} \sum_{n=i}^{\infty} \frac{\eta P^O_{W_i,U_n}(t)}{\eta + \lambda^O_n} \Bigg/ \sum_{n=i}^{\infty} P^O_{W_i,U_n}(t) \;\; (l = l_r,\ l_r+1,\ l_r+2,\ \ldots), \tag{46}$$

$$SUA^O_{rc}(t;l) = \sum_{i=0}^{l} \binom{l}{i} a^i b^{l-i} \sum_{n=i}^{\infty} \frac{\theta P^O_{W_i,R_n}(t)}{\theta + \mu^O_n} \;\; (l = l_r,\ l_r+1,\ l_r+2,\ \ldots), \tag{47}$$

$$SUA^O_r(t;l) = \sum_{i=0}^{l} \binom{l}{i} a^i b^{l-i} \sum_{n=i}^{\infty} \frac{\theta P^O_{W_i,R_n}(t)}{\theta + \mu^O_n} \Bigg/ \sum_{n=i}^{\infty} P^O_{W_i,R_n}(t) \;\; (l = l_r,\ l_r+1,\ l_r+2,\ \ldots), \tag{48}$$

respectively. Equations (46), (47), and (48) represent the software service availability in use and the software service unavailabilities due to request cancellation and under restoration, given that the system is release after the l_r-th debugging was complete in the testing phase, respectively. We note that Equations (47) and (48) have no bearing on the parameter η.

5. Numerical Examples

Using the model discussed above, we present several numerical illustrations of operational software availability assessment, where we apply $\lambda_n \equiv Dc^n$ $(D > 0,\ 0 < c < 1)$ and $\mu_n \equiv Er^n$ $(E > 0,\ 0 < r \leq 1)$ to the hazard and the restoration rates, respectively [9].

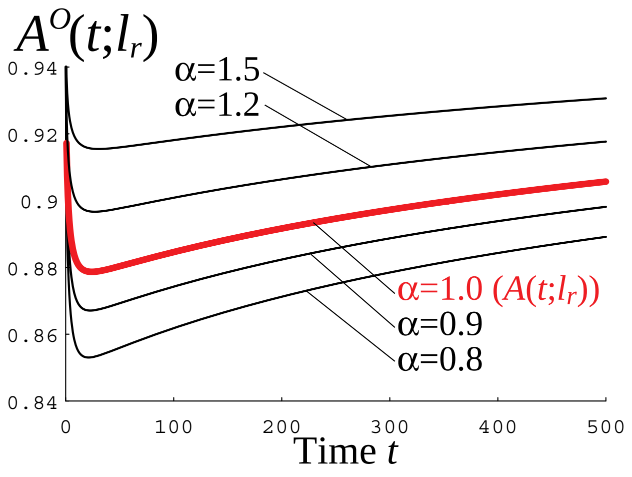

Figure 5: $A^O(t;l_r)$ for various values of α $(\beta = 1.0,\ l_r = 26)$

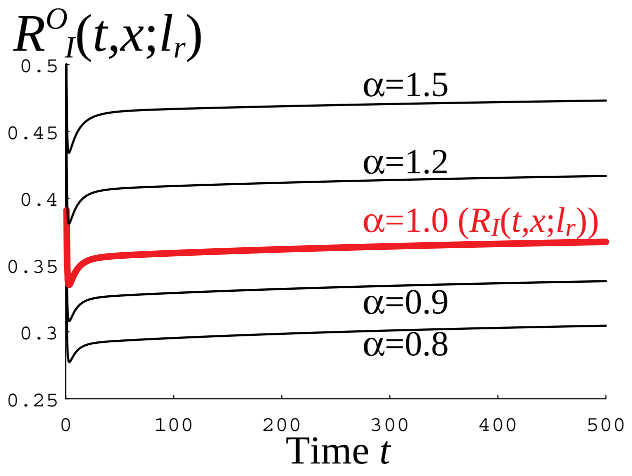

Figure 6: $R_I^O(t,x;l_r)$ for various values of α $(\beta = 1.0,\ l_r = 26)$

We cite the estimates of the parameters associated with λ_n and μ_n from Ref. [16], i.e., we use the following values:

$$\widehat{D} = 0.246,\ \hat{c} = 0.940,\ \widehat{E} = 1.114,\ \hat{r} = 0.960,$$

where we set $a = 0.8$. These values have been estimated based on the simulated data set generated from data cited by Goel and Okumoto [4]; this consists of 26 software failure-occurrence time-interval data $(l_r = 26)$ and the unit of time is day.

The inherent availabilities for one up-down cycle when n faults have been corrected in the testing and the operation phases can be defined as

$$A_I(n) \equiv \frac{\mathrm{E}[Z_n]}{\mathrm{E}[Z_n] + \mathrm{E}[T_n]} = \frac{1}{1+\rho_n} \quad (\rho_n \equiv \lambda_n/\mu_n), \tag{49}$$

$$A_I^O(n) \equiv \frac{\mathrm{E}[Z_n^O]}{\mathrm{E}[Z_n^O] + \mathrm{E}[T_n^O]} = \frac{1}{1+\rho_n^O} \quad \left(\rho_n^O \equiv (\beta/\alpha)\cdot(\lambda_n/\mu_n)\right), \tag{50}$$

respectively, where ρ_n and ρ_n^O are called the maintenance factor. $A_I(n)$ and $A_I^O(n)$ are the simplest availability measures. From the forms of Equations (49) and (50), the difference

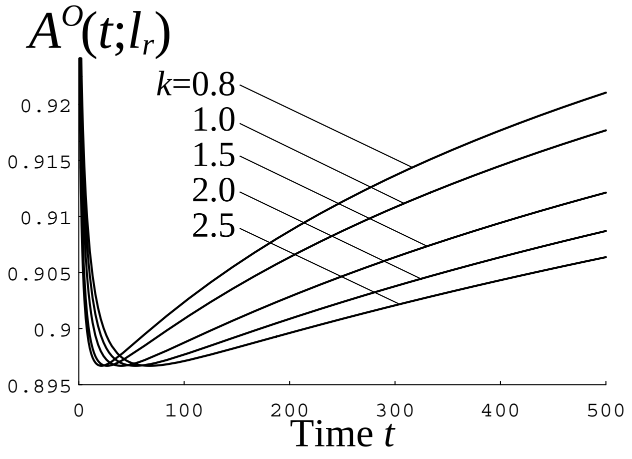

Figure 7: $A^O(t; l_r)$ for various values of α and β, given $\beta/\alpha = 1/1.2$ ($l_r = 26$)

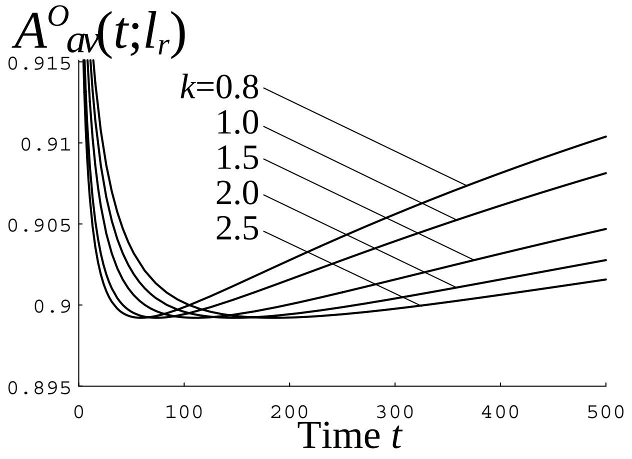

Figure 8: $A^O_{av}(t; l_r)$ for various values of α and β, given $\beta/\alpha = 1/1.2$ ($l_r = 26$)

of software availability assessment between the testing and the operation phases with the inherent availability depends on the value of β/α. In other words, the same evaluation in software availability is given when the value of β/α is constant even though α and β take different values. Especially in the case of $\alpha = \beta$, we judge that the testing and the operation phases are the same availability evaluation since $A_I(n) = A^O_I(n)$.

The software availability measures shown in this paper include infinite series, however, in practical calculation of these measures, we need to specify the supremum of n, denoted as N_0, instead of infinity. For example, we calculate $\sum_{n=i}^{N_0} \frac{\beta g^O_{i,n+1}(t)}{a\mu_n}$ instead of $\sum_{n=i}^{\infty} \frac{\beta g^O_{i,n+1}(t)}{a\mu_n}$ in Equation (16). If we can estimate the initial fault content in the system, n_0, with some method, it is appropriate that $N_0 = n_0$. Otherwise we set an adequate integer for practical calculation of software availability measures to N_0. In the case of Figure 5, the time axis designated is $[0,\ 500]$ and $l_r = 26$, then, $G^O_{26,55}(500) = 1.011 \times 10^{-9}$. That is, the probability that the system makes a transition from state W_{26} to W_{55} in the time interval $[0,\ 500]$ is sufficiently small. Accordingly, we set $N_0 = 55$.

Figures 5 and 6 show the dependence of the instantaneous software availability, $A^O(t; l)$, in Equation (16) and the interval software reliability, $R^O_I(t, x; l)$, in Equation (18) on the value of α, where the cases of $\alpha = 1.0$ designated by thick lines are identical with Equa-

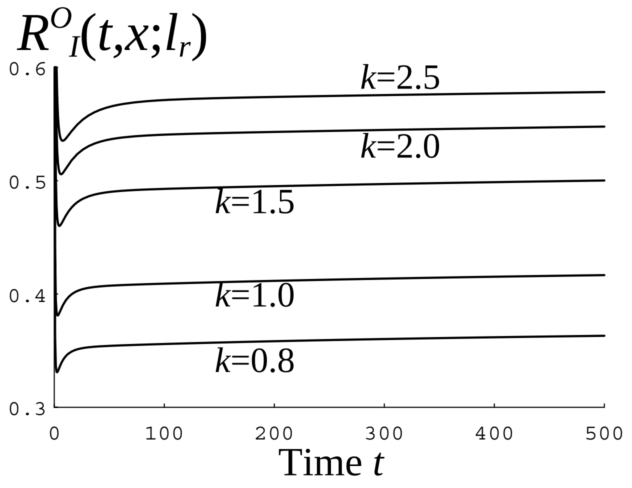

Figure 9: $R^O_I(t,x;l_r)$ for various values of α and β, given $\beta/\alpha = 1/1.2$ ($x = 10.0,\ l_r = 26$)

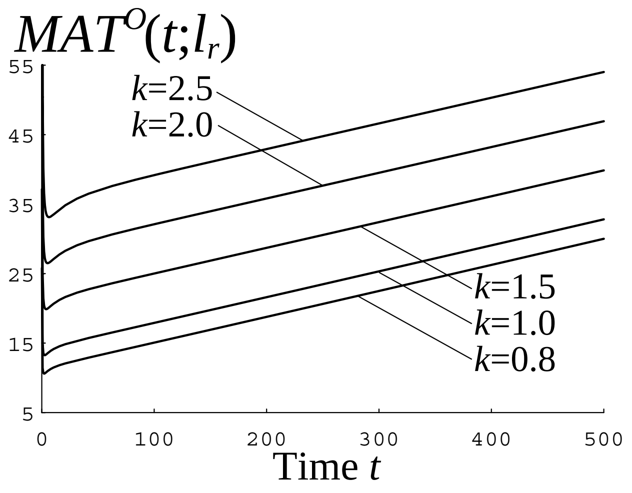

Figure 10: $MAT^O(t;l_r)$ for various values of α and β, given $\beta/\alpha = 1/1.2$ ($l_r = 26$)

tions (9) and (11), respectively. We can see that software availability becomes higher as the value of α is estimated larger. This is the same tendency as the case of the inherent availability.

Hereafter, we set $\alpha_0 = 1.2$, $\beta_0 = 1.0$, $\alpha = k\alpha_0$, and $\beta = k\beta_0$, and show the numerical examples on the dependence of the value of k, i.e., the both of the values of α and β are varied, given β/α is constant.

Figures 7 and 8 show the dependence of $A^O(t;l)$ and the average software availability, $A^O_{av}(t;l)$, in Equation (17) on the values of α and β, given β/α is constant, respectively. These figures tell us that software availability becomes lower as both of the values of α and β are estimated larger; this result is different from the case of the inherent availability. The larger α and β mean that an up-down cycle period becomes longer. This fact also leads slow software reliability growth. In other words, the shorter up-down cycle period means that software reliability growth speeds up.

Figures 9 and 10 show the dependence of $R^O_I(t,x;l)$ and the conditional mean available time, $MAT^O(t;l)$, in Equation (19), on the values of α and β, given β/α is constant, respectively. These figures display the opposite tendency to the Figures 7 and 8, i.e., software availability becomes larger as both of the values of α and β becomes larger. The instantaneous and the average software availabilities are the measures focusing on time instant, on the other hand, the interval software reliability and the conditional mean available time focus on whether or not the system is available continuously for a time interval. The larger α implies the following effects: (A) up time lengthens, on the other hand, (B) software reliability growth slows down. As to the software availability evaluation based on the interval software reliability or the conditional mean available time, effect (A) has a greater impact than effect (B).

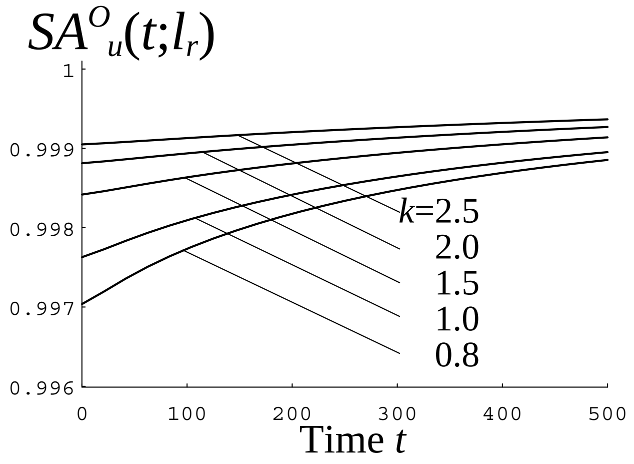

Figure 11: $SA^O_u(t;l_r)$ for various values of α and β, given $\beta/\alpha = 1/1.2$ ($l_r = 26$; $\theta = 5.0$, $\eta = 24.0$)

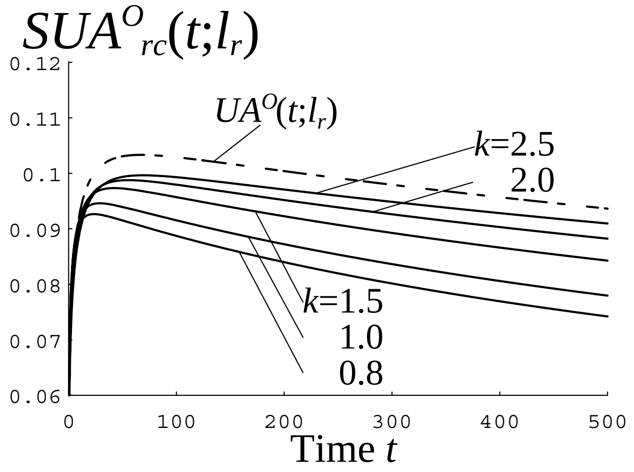

Figure 12: $SUA^O_{rc}(t;l_r)$ for various values of α and β, given $\beta/\alpha = 1/1.2$ ($l_r = 26$; $\theta = 5.0$)

Next we show the numerical examples of the software service availability measures. Figure 11 shows the dependence of the software service availability in use, $SA^O_u(t;l)$, in Equation (46) on the values of α and β, given β/α is constant, in the case where the user uses the system five times a day on average ($\theta = 5.0$) and the mean usage time is one hour ($1/\eta = 1/24.0$). This figure tells us that software service availability becomes larger as time elapses and both of the values of α and β becomes larger; this is a similar tendency to

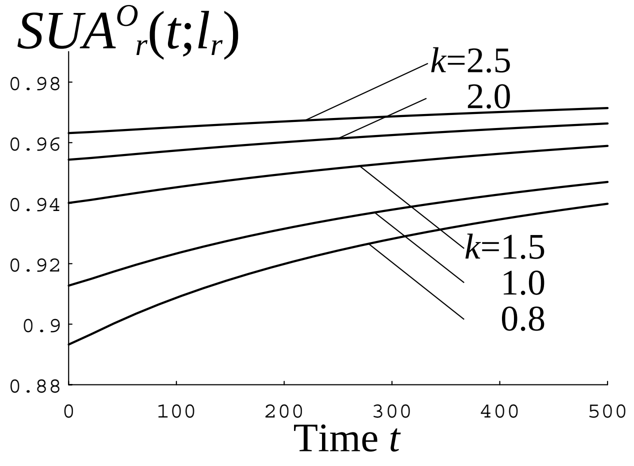

Figure 13: $SUA_r^O(t;l_r)$ for various values of α and β, given $\beta/\alpha = 1/1.2$ ($l_r = 26;\ \theta = 5.0$)

$R_I^O(t,x;l)$ and $MAT^O(t;l)$.

Figure 12 shows the dependence of the software service unavailability due to request cancellation, $SUA_{rc}^O(t;l)$, in Equation (47) on the values of α and β, given β/α is constant, where the broken line designates the instantaneous software unavailability denoted as $UA^O(t;l) \equiv 1 - A^O(t;l)$ and defined as the probability that the system is down and restored at the time point t. As shown in this figure, the probability that the user's request is canceled becomes lower with the lapse of time. We can also see that the proposed measure shows more optimistic evaluation than the traditional measure. Furthermore, this figure indicates that $SUA_{rc}^O(t;l)$ shows the opposite tendency to $SA_u^O(t;l)$, i.e., the software service unavailability is estimated higher when α and β are estimated larger. This reasoning is that $SUA_{rc}^O(t;l)$ is the measure noting the relationship between the usage frequency of the user and the restoration time, not the usage time and the operating time of the system. The larger β leads that the restoration time is estimated longer.

On the other hand, if we observe that the system is down and restored, then we have the different evaluation from the above mention. Figure 13 shows the dependence of the software service unavailability under restoration, $SUA_r^O(t;l)$, in Equation (48) on the values of α and β, given β/α is constant. As this figure indicates, $SUA_r^O(t;l)$ increases, i.e., the software availability evaluation becomes more unfavorable with the lapse of time. The reason for this behavior is that the mean restoration time is assumed the non-increasing function of n from assumption A3.

6. Concluding Remarks

In this paper, we have discussed the user-oriented software availability evaluation methods. We have used the Markovian software availability model to describe the software failure and restoration characteristics in the testing phase of the software development process and the user operation phase. Assuming that the ratio of the time-scale transformation between the testing and the operation phases is constant, we have introduced the environmental factors to express the difference between the testing and the operation environments. Furthermore, defining the user-perceived software failure, we have proposed the modeling for the software service availability assessment. From this discussion, we have derived several quantitative measures for software availability measurement and assessment oriented to operational use;

these have been given as the functions of the operation time and the number of debuggings performed in the testing phase. We have presented several numerical examples of operational software service availability analysis and investigated the impacts of the environmental factors and the consideration of the user's behavior on the software availability evaluation. It is meaningful that this study has revealed a clue to quantitate "the quality of service" of software systems.

We have illustrated the numerical examples based on the simulation data, especially, the values of the environmental factors, α and β, have been given experimentally. In the practical use of this model, the estimation of α and β is important. However, it seems to be difficult to estimate the values of α and β by using the data obtained from the corresponding software project. In the present state of affairs, we cannot help deciding the values of α and β experimentally based on the analysis of the past field data which are obtained from the software systems developed before by similar projects. The practical estimation of α and β remains a future study.

Acknowledgment
This work was supported in part by Grants-in-Aid for Scientific Research (C) of the Ministry of Education, Culture, Sports, Science and Technology of Japan under Grant No. 18510124.

References

[1] H. Asama: Service engineering and system integration. *Journal of the Society of Instrument and Control Engineering*, **44** (2005), 278–283 (in Japanese).

[2] A. Birolini: *Reliability Engineering — Theory and Practice — Third Edition* (Springer-Verlag, Berlin, 1999).

[3] D.P. Gaver, Jr.: A probability problem arising in reliability and traffic studies. *Operations Research*, **12** (1964), 534–542.

[4] A.L. Goel and K. Okumoto: Time-dependent error-detection rate model for software reliability and other performance measures. *IEEE Transactions on Reliability*, **R-28** (1979), 206–211.

[5] M. Kaâniche, K. Kanoun and M. Martinello: A user-perceived availability evaluation of a web based travel agency. In *Proceedings of the 2003 International Conference on Dependable Systems and Networks* (2003), 709–718.

[6] M.R. Lyu, ed.: *Handbook of Software Reliability Engineering* (McGraw-Hill, New York, 1996).

[7] V. Mainkar: Availability analysis of transaction processing systems based on user-perceived performance. In *Proceedings of the 16th Symposium on Reliable Distributed Systems* (1997), 10–17.

[8] H. Mizuta: Emergence of service science: Services sciences, management and engineering (SSME). *IPSJ Magazine*, **47** (2006), 457–472 (in Japanese).

[9] P.B. Moranda: Event-altered rate models for general reliability analysis. *IEEE Transactions on Reliability*, **R-28** (1979), 376–381.

[10] H. Okamura, T. Dohi and S. Osaki: A reliability assessment method for software products in operational phase —Proposal of an accelerated life testing model—. *Transactions of IEICE*, **J83-A-3** (2000), 294–301 (in Japanese).

[11] S. Osaki: Reliability analysis of a system when it is used intermittently. *Transactions of IECE*, **54-C** (1971), 83–89 (in Japanese).

[12] S. Osaki: *Applied Stochastic System Modeling* (Springer-Verlag, Heidelberg, 1992).

[13] K. Tokuno and S. Yamada: Markovian software availability measurement based on the number of restoration actions. *IEICE Transactions on Fundamentals*, **E83-A-5** (2000), 835–841.

[14] K. Tokuno and S. Yamada: Markovian software availability measurement for continuous use. In H. Pham and M.-W. Lu (eds.): *Proceedings of the Sixth ISSAT International Conference on Reliability and Quality in Design* (2000), 280–284.

[15] K. Tokuno and S. Yamada: Software availability theory and its applications. In H. Pham (ed.): *Handbook of Reliability Engineering* (Springer-Verlag, London, 2003), 235–244.

[16] K. Tokuno and S. Yamada: Stochastic performance evaluation for multi-task processing system with software availability model. *Journal of Quality in Maintenance Engineering*, **12** (2006), 412–424.

[17] M. Tortorella: Service reliability theory and engineering, I: Foundations. *Quality Technology and Quantitative Management*, **2** (2005), 1–16.

[18] M. Tortorella: Service reliability theory and engineering, II: Models and examples. *Quality Technology and Quantitative Management*, **2** (2005), 17–37.

[19] D. Wang and K.S. Trivedi: Modeling user-perceived service availability. In M. Malek, E. Nett and N. Suri (eds.): *Service Availability — 2nd International Service Availability Symposium, ISAS 2005 —* (Springer-Verlag, Berlin, 2005), 107–122.

[20] `http://www.saforum.org`.

Koichi Tokuno
Department of Social Systems Engineering
Faculty of Engineering
Tottori University
4-101, Koyama, Tottori-shi, 680-8552, Japan
E-mail: `toku@sse.tottori-u.ac.jp`