Stochastic Performance Evaluation for Software System Considering NHPP Task Arrival

KOICHI TOKUNO*
Department of Social Systems Engineering, Tottori University
Minami 4-101, Koyama, Tottori-shi, 680-8552, Japan

MASAMITSU FUKUDA
Department of Social Systems Engineering, Tottori University
Minami 4-101, Koyama, Tottori-shi, 680-8552, Japan

SHIGERU YAMADA
Department of Social Systems Engineering, Tottori University
Minami 4-101, Koyama, Tottori-shi, 680-8552, Japan

(Received on March 10, 2006)

Abstract: In this paper, we discuss the software performance evaluation method considering the real-time property. The time-dependent behavior of the software system itself alternating between up and down states is described by the Markovian software availability model. Assuming that the software system can process the multiple tasks simultaneously and that the arrival process of the tasks follows a nonhomogeneous Poisson process, we analyze the distribution of the number of tasks whose processes can be completed within a prespecified processing time limit with the infinite server queueing model. We derive several stochastic quantities for software performance measurement considering the real-time property; these are given as the functions of time and the number of debugging activities. Finally, we illustrate several numerical examples of the quantities to investigate the relationship between the software reliability/restoration characteristics and the system performance.

Keywords: Performance evaluation, Software availability model, Markov process, Real-time property, Infinite server queueing model.

1. Introduction

Recently, the service reliability theory and engineering have a growing attention; these consider the situations and behaviors of the users receiving services by the operation of machines or systems as well and are more comprehensive frameworks than the conventional reliability engineering. Tortorella [1, 2] has described the basic concepts and the methods of the service reliability engineering; this aims to establish quantitative evaluation methods for the quality of service created by the use of the artificial industrial products as well as the inherent quality of the products. 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 discuss the performance evaluation methods for

*Corresponding author’s email: toku@sse.tottori-u.ac.jp
software systems oriented to the service reliability engineering. Recently, the consortium of the Service Availability Forum [3] has been created to develop the computing framework and the interface between hardware and software systems with high service availability.

The studies on performance evaluation considering reliability for computing systems have much been discussed from the viewpoint of the hardware configuration [4, 5]. On the other hand, from the viewpoint of software system, the discussions on inherent quality/reliability evaluation such as the estimation of the residual fault content and the prediction of software failure time have much been conducted [6, 7], while there exist few studies on the reliability-related performance evaluation. Kimura et al. [8, 9] have discussed the evaluation methods of the real-time property for the N-version programming and the recovery block software systems; these are well-known as the methodologies of the fault-tolerant software systems, where the real-time property is defined as the attribute that the system can complete the task within the stipulated response time limit [10, 11]. However, Kimura's studies have just applied the framework for the analyzing from the aspect of the hardware configuration to the fault-tolerant software systems and have not included the characteristics particular to software systems such as the reliability growth process and the upward tendency of difficulty in debugging.

In this paper, we discuss the performance evaluation method of the software systems considering the real-time property; this is the different approach from Kimura's studies. We assume that the software system can process the multiple tasks simultaneously and that the cumulative number of the arriving tasks follows a nonhomogeneous Poisson process (NHPP). Then the time-dependent behavior of the software system alternating between up and down states in the dynamic environment are described by the Markovian software availability model [12]. The stochastic behavior of the number of tasks whose processes can be complete within the prespecified processing time limit is modeled with the infinite sever queueing model [13].

The organization of the rest of the paper is shown as follows. Section 2 states the software availability model used in the paper. Section 3 describes the stochastic processes of the numbers of tasks whose processes are complete within the prespecified processing time limit and canceled out of the tasks arriving up to a given time point. Section 4 derives several software performance measures considering the real-time property. The measures are given as the functions of time and the number of debuggings. Section 5 presents several numerical examples of software performance analysis. Finally, Section 6 summarizes the results obtained in this paper.

2. Markovian Software Availability Model

2.1 Model Description

The following assumptions are made for software availability modeling [12].

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 \(b = 1 - a\). When the debugging activity succeeds, one fault is corrected and the software reliability growth occurs.
A3. The next software failure time, $X_n$, and the restoration time, $U_n$, when $n$ faults have already been corrected from the system, follow the exponential distributions with means $1/\lambda_n$ and $1/\mu_n$, respectively. $\lambda_n$ and $\mu_n$ are non-increasing functions of $n$.

The state space of the stochastic process {$Z(t), t \geq 0$} representing the state of the software system at the time point $t$ is defined as follows:

- $W_n$: the system is operating,
- $R_n$: the system is inoperable and in the process of restoration,

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

**Fig. 1: A Sample State Transition Diagram of $Z(t)$**

### 2.2 Software Availability Measures

Let $S_{i,n}$ ($i \leq n$) be the random variable representing the transition time of $Z(t)$ 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 $G_{i,n}(t)$:

$$G_{i,n}(t) = Q_{W_i,R_i} * Q_{R_i,W_{i+1}} * G_{i+1,n}(t) + Q_{W_i,R_i} * Q_{R_i,W_i} * G_{i,n}(t),$$

(1)

where $Q_{A,B}(\tau)$ ($A, B \in \{W_n, R_n; n = 0, 1, 2, \ldots\}$) denotes the one-step transition probability that after making a transition into state $A$, the process {$Z(t), t \geq 0$} makes a transition into state $B$ by time $\tau$, and $*$ denotes a Stieltjes convolution.

We use Laplace-Stieltjes (L-S) transforms to solve Eq. (1), where the L-S transform of $G_{i,n}(t)$ is defined as

$$\tilde{G}_{i,n}(s) = \int_0^\infty e^{-st} \, dG_{i,n}(t).$$

(2)

Accordingly, we obtain $\tilde{G}_{i,n}(s)$ as

$$\tilde{G}_{i,n}(s) = \prod_{m=1}^{n-1} \frac{a_m \lambda_m \mu_m}{(s + x_m)(s + y_m)} = \sum_{m=1}^{n-1} \left( \frac{A_{1,m}(m)x_m}{s + x_m} + \frac{A_{2,m}(m)y_m}{s + y_m} \right),$$

(3)

where
respectively. By inverting Eq. (3), we obtain the distribution function of \( S_{i,n} \) as

\[
Pr\{ S_{i,n} \leq t \} = 1 - \sum_{m=1}^{n-1} \left\{ A_{i,m}^1(m) e^{-\gamma_m t} + A_{i,n-m}^2(m) e^{-\gamma_m t} \right\}.
\]  

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

First, we derive \( P_{W_i,W_n}(t) \). We obtain the following renewal equations with respect to \( P_{W_i,W_n}(t) \):

\[
R_{W_i,W_n}(t) = G_{i,n} * P_{W_i,W_n}(t),
\]

\[
R_{W_i,W_n}(t) = e^{-\lambda_t} + Q_{W_i,R_n} * Q_{W_n,W_n} * P_{W_i,W_n}(t).
\]

Substituting the L-S transforms of Eq. (9) into that of Eq. (8) yields

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

By inverting Eq. (10), \( P_{W_i,W_n}(t) \) is obtained as

\[
P_{W_i,W_n}(t) = Pr\{ Z(t) = W_n \mid Z(0) = W_i \} = \frac{g_{i,n+1}(t)}{a\lambda_n} + \frac{g'_{i,n+1}(t)}{a\lambda_n^2\mu_n},
\]

where \( g_{i,n}(t) \) is the probability density function associated with \( G_{i,n}(t) \) and \( g'_{i,n}(t) = dg_{i,n}(t)/dr \).

Using the similar procedure for the derivation of \( P_{W_i,R_n}(t) \), we obtain the following renewal equations with respect to \( P_{W_i,R_n}(t) \):

\[
P_{W_i,R_n}(t) = G_{i,n} * Q_{W_i,R_n} * P_{R_n}(t),
\]
Stochastic Performance Evaluation for Software System Considering NHPP Task Arrival

\[
P_{R_n,R_n}(t) = e^{-\mu t} + Q_{R_n,W_n} * Q_{W_n,R_n} * P_{R_n,R_n}(t). \tag{13}
\]

Substituting the L-S transform of Eq. (13) into that of Eq. (12) yields

\[
\tilde{P}_{R_n,R_n}(s) = \frac{sG_{l,n+1}(s)}{a\mu_n}. \tag{14}
\]

By inverting Eq. (14), \( P_{W_i,R_n}(t) \) is obtained as

\[
P_{W_i,R_n}(t) \equiv Pr\{Z(t) = R_n \mid Z(0) = W_i\}
\]

\[
= \frac{g_{l,n+1}(t)}{a\mu_n}. \tag{15}
\]

The instantaneous and the average software availabilities are given by

\[
A(t;l) \equiv \sum_{i=0}^{l} \binom{l}{i} a^i b^{l-i} \sum_{n=0}^{\infty} P_{W_i,W_n}(t)
\]

\[
= 1 - \sum_{i=0}^{l} \binom{l}{i} a^i b^{l-i} \sum_{n=0}^{\infty} \frac{g_{l,n+1}(t)}{a\mu_n}, \tag{16}
\]

\[
A_w(t;l) \equiv \int_0^t A(x;l) \, dx
\]

\[
= 1 - \frac{1}{t} \sum_{i=0}^{l} \binom{l}{i} a^i b^{l-i} \sum_{n=0}^{\infty} \frac{g_{l,n+1}(t)}{a\mu_n}, \tag{17}
\]

respectively, where \( \binom{l}{i} \equiv l!/[i!(l-i)!] \) denote the binomial coefficient and we use the identical equation \[ \sum_{n=0}^{\infty} \{ P_{W_i,W_n}(t) + P_{W_n,R_n}(t) \} \equiv 1 \] for arbitrary time \( t \). 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 \( (0, t] \), given that the \( l \)-th debugging activity was complete at time point \( t=0 \), respectively.

3. Model Analysis

We make the following assumptions for system's task processing.

B1. The process \{\( N(t), \geq 0 \)\} representing the number of tasks arriving at the system up to the time \( t \) follows the NHPP with the mean value function \( \Omega(t) \) and the intensity function \( \omega(t) \equiv \Omega'(t)/dt \).

B2. The processing time of a task, \( Y \), follows a general distribution whose distribution function is denoted as \( H(t) \). Each of the processing times is independent.

B3. When the system causes a software failure in task processing or the processing times of tasks exceed the prespecified processing time limit, \( T_r \), the corresponding tasks are canceled.

B4. The number of tasks the system can process simultaneously is sufficiently large.

Figure 2 illustrates the configuration of system's task processing. Hereafter, we discuss
based on the situation where \( i \) faults have already been corrected at time point \( t=0 \).

**Fig. 2: Configuration of System’s Task Processing**

Let \( \{ X_i(tT_r) \}, i \geq 0 \) be the random variable representing the cumulative number of tasks whose processes can be complete within the processing time limit \( T_r \) out of the tasks arriving up to the time \( t \). By conditioning with \( \{ N(t) = k \} \), we can obtain the probability mass function of \( X_i(tT_r) \) as

\[
Pr\{ X_i(tT_r) = j \} = \sum_{k=0}^{\infty} Pr\{ X_i(tT_r) = j \mid N(t) = k \} e^{-\Omega(t)} \frac{[\Omega(t)]^k}{k!} \quad (j = 0, 1, 2, \ldots).
\]

From Fig. 2, given that \( \{ Z(t) = W_n \} \) and that an arbitrary task arrives at the system at the time point \( t \), the probability that the process of an arbitrary task is complete within the processing time limit \( T_r \) is given by

\[
\beta_n(T_r) \equiv Pr\{ T_r > Y, X_n > Y \mid Z(t) = W_n \} = \int_0^{T_r} e^{-\lambda y} \, dH(y);
\]

this equation is independent of time \( t \) since \( X_n \) has a memoryless property. Then unconditioning Eq. (19) with respect to the cumulative number of corrected faults, \( n \), yields

\[
\pi_i(x \mid T_r) = \sum_{n=i}^{\infty} \beta_n(T_r) \cdot Pr\{ Z(x) = W_n \mid Z(0) = W_i \}
= \sum_{n=i}^{\infty} \beta_n(T_r) \left[ g_{i,n+1}(x) + g'_{i,n+1}(x) \right] \frac{1}{a\lambda_n + a\mu_n}. \tag{20}
\]

Equation (20) means that the probability that the process of a task is complete within \( T_r \) on the condition that the task has arrived at the system at the time point \( x \) (\( 0 \leq x \leq t \)). Furthermore, the arrival time of an arbitrary task out of ones arriving up to the time \( t \) is distributed with the density function \( f(x) = \alpha(x)/\Omega(t) \) (\( 0 \leq x \leq t \)) from the infinite server queueing theory [13]. Therefore, the probability that the process of an arbitrary task having arrived up to the time \( t \) is complete within the processing time limit \( T_r \) is obtained as
Stochastic Performance Evaluation for Software System
Considering NHPP Task Arrival

\[ p_i(t | T_r) = \int_0^\infty \pi_i(x | T_r) f(x) \, dx \]
\[ = \frac{1}{\Omega(t)} \sum_{n=1}^{\infty} \beta_n(T_r) \left\{ \frac{g_{i,n+1}(x)}{a\lambda_n} + \frac{g'_{i,n+1}(x)}{a\lambda_n\mu_n} \right\} \sigma(x) \, dx. \]  

Then from assumption B2,

\[ \Pr\{X_i(t | T_r) = j | N(t) = k\} = \begin{cases} \binom{k}{j} \left[ p_i(t | T_r) \right]^j \left[ 1 - p_i(t | T_r) \right]^{k-j} & (j = 0, 1, 2, \ldots, k) \\ 0 & (j > k) \end{cases} \]

That is, given that \( N(t)=k \), the number of tasks whose processes can be complete within the processing time limit \( T_r \) follows the binomial process with mean \( kp_i(t | T_r) \).

Accordingly, from Eq. (18) the distribution of \( X_i(t | T_r) \) is given by

\[ \Pr\{X_i(t | T_r) = j | N(t) = k\} = \frac{e^{-\Omega(t)} \left[ \Omega(t) \right]^j}{j!} \]

Equation (23) means that \( \{X_i(t | T_r) \geq 0\} \) follows the NHPP with mean value function \( \Omega(t)p_i(t | T_r) \).

Let \( \{W_i(t | T_r) \geq 0\} \) be the random variable representing the cumulative number of tasks whose processes are interrupted out of the tasks arriving up to the time \( t \). Then we can apply the same discussion as above to the derivation of the distribution of \( W_i(t | T_r) \), i.e., we can obtain \( \Pr\{W_i(t | T_r) = j\} \) as

\[ \Pr\{W_i(t | T_r) = j\} = e^{-\Omega(t)q_i(t | T_r)} \left[ \Omega(t)q_i(t | T_r) \right]^j \]  

Equation (24) means that \( \{W_i(t | T_r) \geq 0\} \) follows the NHPP with mean value function \( \Omega(t)q_i(t | T_r) \).

4. Derivation of Software Performance Measures

Based on the above analysis, we can obtain several measures for software performance evaluation.

The expected numbers of tasks completable and incompletable within the processing time limit \( T_r \) out of the tasks arriving up to the time \( t \) are given by

\[ \Lambda_i(t | T_r) = E[X_i(t | T_r)] \]
\[ = \sum_{n=1}^{\infty} \beta_n(T_r) \int_0^\infty \left\{ \frac{g_{i,n+1}(x)}{a\lambda_n} + \frac{g'_{i,n+1}(x)}{a\lambda_n\mu_n} \right\} \sigma(x) \, dx. \]
respectively. The instantaneous numbers of tasks completable and incompletable at the
time point \( t \) are given by

\[
M_i(t|T_r) \equiv E[W_i(t|T_r)] = \Omega(t) - \sum_{n=1}^{\infty} \beta_n(T_r) \int_0^t \left( \frac{g_{_{i,n+1}}(x)}{a\lambda_n} + \frac{g'_{_{i,n+1}}(x)}{a\lambda_n\mu_n} \right) \sigma(x) \, dx,
\]

(26)

respectively. These represent the ratios of the numbers of tasks completed and canceled to
that of tasks arriving at the system per unit time at the time point \( t \), respectively. We note
that Eqs. (29) and (30) have no bearing on the arrival rate of the tasks, \( \omega(t) \).

Furthermore, the instantaneous task completion and incompleation ratios are given by

\[
\xi_i(t|T_r) \equiv \frac{d\Lambda_i(t|T_r)}{dt} = \sigma(t) \sum_{n=1}^{\infty} \beta_n(T_r) \left( \frac{g_{_{i,n+1}}(t)}{a\lambda_n} + \frac{g'_{_{i,n+1}}(t)}{a\lambda_n\mu_n} \right),
\]

(27)

\[
m_i(t|T_r) \equiv \frac{dM_i(t|T_r)}{dt} = \sigma(t) \left[ 1 - \sum_{n=1}^{\infty} \beta_n(T_r) \left( \frac{g_{_{i,n+1}}(t)}{a\lambda_n} + \frac{g'_{_{i,n+1}}(t)}{a\lambda_n\mu_n} \right) \right],
\]

(28)

respectively. These represent the ratios of the numbers of tasks completed and canceled to
that of tasks arriving at the system per unit time at the time point \( t \), respectively. We note
that Eqs. (29) and (30) have no bearing on the arrival rate of the tasks, \( \omega(t) \).

As to \( p_i(t|T_r) \) in Eq. (21) and \( q_i(t|T_r) \) in Eq. (24), we can give the following
interpretations:

\[
p_i(t|T_r) = \frac{E[X_i(t|T_r)]}{E[N(t)]},
\]

(31)

\[
q_i(t|T_r) = \frac{E[W_i(t|T_r)]}{E[N(t)]}.
\]

(32)

That is, \( p_i(t|T_r) \) and \( q_i(t|T_r) \) are the cumulative task completion and incompleation ratios in
the time interval \( (0, t] \), respectively.

We should note that it is too difficult to use Eqs. (25)-(32) practically since this model
assumes the imperfect debugging environment and the initial condition \( i \) appearing in the
above equations, which represents the cumulative number of corrected faults, cannot be observed immediately. However, applying the similar idea to $A(t; l)$ and $A_{av}(t; l)$ in Section 2, we can convert Eqs. (25)-(32) into the functions of the number of debuggings, $l$, i.e., we obtain

$$\Lambda(t, l \mid T_r) = \sum_{i=0}^{l} \binom{l}{i} a^i b^{l-i} \Lambda_i (t \mid T_r),$$  \tag{33}$$

$$M(t, l \mid T_r) = \sum_{i=0}^{l} \binom{l}{i} a^i b^{l-i} M_i (t \mid T_r),$$  \tag{34}$$

$$\xi(t, l \mid T_r) = \sum_{i=0}^{l} \binom{l}{i} a^i b^{l-i} \xi_i (t \mid T_r),$$  \tag{35}$$

$$m(t, l \mid T_r) = \sum_{i=0}^{l} \binom{l}{i} a^i b^{l-i} m_i (t \mid T_r),$$  \tag{36}$$

$$h(t, l \mid T_r) = \sum_{i=0}^{l} \binom{l}{i} a^i b^{l-i} h_i (t \mid T_r),$$  \tag{37}$$

$$\chi(t, l \mid T_r) = \sum_{i=0}^{l} \binom{l}{i} a^i b^{l-i} \chi_i (t \mid T_r),$$  \tag{38}$$

$$p(t, l \mid T_r) = \sum_{i=0}^{l} \binom{l}{i} a^i b^{l-i} p_i (t \mid T_r),$$  \tag{39}$$

$$q(t, l \mid T_r) = \sum_{i=0}^{l} \binom{l}{i} a^i b^{l-i} q_i (t \mid T_r),$$  \tag{40}$$

respectively. Equations (33)-(40) represent the expected numbers of tasks completable and incompletable, the instantaneous numbers of tasks completable and incompletable, the instantaneous task completion and incompletion ratios, the cumulative task completion and incompletion ratios at the time point $t$, given that the $l$-th debugging was complete at the time point $t=0$, respectively.

5. Numerical Examples

We show several numerical examples of software performance analysis based on the above measures. Here, we use Moranda’s model [14] to the hazard rate $\lambda_n \equiv Dc^n$ ($D>0$, $0<c<1$) and the restoration rate $\mu_n \equiv Er^n$ ($E>0$, $0<r\leq1$), respectively. We cite the estimates of the parameters associated with $\lambda_n$ and $\mu_n$ from Tokuno [15], i.e., we use the following values:

$$D = 0.246, \quad c = 0.940, \quad E = 1.114, \quad r = 0.960,$$

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 [16]; this consists of 26 software failure-occurrence time-interval data and the unit of time is day.

For the distribution of the processing time of a task, $Y$, we apply the gamma distribution whose density is given by
where $\nu$ and $\alpha$ are the shape and the scale parameters, respectively. Then the mean and the variance of the processing time are given by $E[Y]=\nu/\alpha$ and $\text{Var}[Y]=\nu/\alpha^2$, respectively. Furthermore, we apply the Weibull process to $\{N(t), t \geq 0\}$ representing the number of tasks arriving at the system, i.e., the mean value function is given by $\Omega(t)\equiv \eta t^{\psi} (t \geq 0; \eta > 0, \psi > 0)$. In the special case of $\psi=1$, $N(t)$ follows the homogeneous Poisson process.

Figure 3 shows the instantaneous task completion ratio, $h(t;|T_r)$, in Eq. (37) for the various numbers of debuggings, $l$, respectively, where $T_r=0.005$, $\nu=2.0$, $\alpha=1000.0$. We can see that software performance also improves as the debugging processes.

![Fig. 3: $h(t;|T_r)$ for Various Number of Debuggings, $l$](image)

$(T_r=0.005; \nu=2.0, \alpha=1000.0)$

Figure 4 shows the time-dependent behaviors of $h(t;|T_r)$ and the instantaneous task incompletion ratio, $\chi(t;|T_r)$, in Eq. (38) along with the instantaneous software availability, $A(t;l)$, in Eq. (16). This figure tells us that the new measure considering the real-time property ($h(t;|T_r)$) gives more pessimistic evaluation than the traditional one ($A(t;l)$).

![Fig. 4: $h(t;|T_r)$, $\chi(t;|T_r)$, and $A(t;l)$](image)

$(T_r=0.005; l=26, \nu=2.0, \alpha=1000.0)$
Fig. 5: Dependence of $\xi(t,l|T_r)$ on Distribution of $Y, H(t)$

Figure 5 shows the dependence of the instantaneous number of task completable, $\xi(t,l|T_r)$, in Eq. (35) on the distribution of the processing time of a task, $H(t)$, in the case of $\eta=1.0$ and $\varphi=0.5$; this case shows that $\xi(t,l|T_r)$ decreases with time since the intensity function of $N(t)$ is also the decreasing one of time $t$. In Fig. 5, we set the parameters $\nu$ and $\alpha$ as equalize the means of the processing time for $H_1(t)$ and $H_2(t)$, i.e.,

$$H(t) = H_1(t) = 1 - e^{-\alpha t}$$

(exponential distribution: $\nu=\nu_1=1.0$, $\alpha=\alpha_1=500.0$),

$$H(t) = H_2(t) = 1 - (1 + \alpha t)e^{-\alpha t}$$

(gamma distribution of order two: $\nu=\nu_2=2\nu_1$, $\alpha=\alpha_2=2\alpha_1$),

respectively. This figure indicates that the performance evaluation in the case of the gamma distribution is higher than that of the exponential distribution. As to the variances of the processing time, the cases of $H_1(t)$ and $H_2(t)$ are $1/\alpha_1^2$ and $2/\alpha_2^2=1/(2\alpha_1^2) < 1/\alpha_1^2$, respectively. We can see that the smaller dispersion-degree of the processing time rises the software performance evaluation.

Fig. 6: Dependence of $h(t,l|T_r)$ on $r$

Figure 6 shows the dependence of $h(t,l|T_r)$ on $r$ representing the decreasing ratio of the restoration rate, $\mu_r$. According to Tokuno and Yamada [12], the behavior of the maintenance factor, $\rho_n=\lambda_n/\mu_n$, decides whether the instantaneous and the average software availabilities improve or degrade with time, i.e., the traditional software availability improves (degrades) if $\rho_n$ is a decreasing (increasing) function of $n$. From Fig. 6, we can see that $\rho_n$ has a similar impact on software performance evaluation considering real-time
Figures 7 and 8 show the dependence of $\xi(t, l | T_r)$ on $a$, in the cases of $r \geq c$ and $r < c$, respectively. These figures tell us that the software performance becomes higher (lower) as the perfect debugging rate becomes larger when $r \geq c$ ($r < c$). The case of $r < c$ may be a paradoxical result that the software performance decreases more slowly with decreasing $a$. This reasoning is that the proposed measure is related to the ratio of the software failure time and the restoration time, i.e., $\rho_n$ increases more slowly with decreasing $a$ since smaller $a$ means that it is more difficult to increase the cumulative number of corrected faults.

6. Conclusions

In this paper, we have discussed the performance evaluation method for software systems considering real-time property. The stochastic behavior to the software system alternating between up and down states have been described by the Markovian availability model. Assuming that the cumulative number of the tasks arriving at the system up to a given time point follows the NHPP, we have analyzed the distribution of the number of tasks whose processes can be complete within the processing time limit with the infinite server queueing model. From the model, we have derived the quantitative measures for software performance assessment, which have been given as the functions of time and the number of
debugging activities. We have also illustrated the several numerical examples of these measures to show that these measures are useful for grasping the relationship between software performance evaluation and the number of debugging. In particular, it has been meaningful to correlate the real-time property evaluation with the software reliability and restoration characteristics.

The practical method for identifying the stochastic characteristics of the arrival process and the processing time of the tasks remains a future study.

Acknowledgment

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

References


**Koichi Tokuno** received the B.S.E. and M.S. degrees from Hiroshima University, Japan in 1990 and 1992, respectively, and the D.Eng. degree from Tottori University, Japan in 1999. From 1994 to 2001, he worked as an Assistant Professor of Tottori University, Tottori-shi, Japan. Since 2001 he has been working as an Associate Professor at the Faculty of Engineering of Tottori University. His research area includes software reliability and availability modeling and software performance evaluation. He is a regular member of the Institute of Electronics, Information and Communication Engineers, the Information Processing Society of Japan, the Operations Research Society of Japan, the Japan Society for Software Science and Technology, and the Reliability Engineering Association of Japan.

**Masamitsu Fukuda** was born in Shimane Prefecture, Japan, in 1983. He received B.S.E. degree from Tottori University in 2006. Currently, he is a master course student at Course in Social Systems Engineering, Graduate School of Engineering, Tottori University, Japan. His research interests include software reliability engineering and quality engineering. Mr. Fukuda is a student member of IEEE Computer Society.

**Shigeru Yamada** received the B.S.E., M.S., and Ph.D. degrees from Hiroshima University in 1975, 1977, and 1985, respectively. From 1977 to 1980, he worked at the Quality Assurance Department of Nippondenso Co., Aichi Prefecture, Japan. From 1983 to 1988, he was an assistant professor of Okayama University of Science, Okayama, Japan. From 1988 to 1993, he was an Associate Professor at the Faculty of Engineering of Hiroshima University, Japan. Since 1993, he has been working as a professor at the Faculty of Engineering of Tottori University, Tottori-shi, Japan. He has published numerous technical papers in the area of software reliability models, project management, reliability engineering, and quality control. He has authored several books entitled Software Reliability: Theory and Practical Application (Soft Research Center, Tokyo, 1990), Introduction to Software Management Model (Kyoritsu Shuppan, 1993), Software Reliability Models: Fundamentals and Applications (JUSE, Tokyo, 1994), and Statistical Quality Control for TQM (Corona Publishing, Tokyo, 1998). Dr. Yamada received the Best Author Award from the Information Processing Society of Japan in 1992, the TELECOM System Technology Award from the Telecommunications Advancement Foundation in 1993, and the Paper Award Reliability Engineering Association of Japan in 1999. He is a regular member of the Institute of Electronics, Information and Communication Engineers, the Information Processing Society of Japan, the Operations Research Society of Japan, the Japan SIAM, the Reliability Engineering Association of Japan, Japan Industrial Management Association, the Japanese Society for Quality Control, the Society of Project Management of Japan, and the IEEE.