

Availability-Based Software Performability Model with User-Perceived Performance Degradation

Koichi Tokuno and Shigeru Yamada

*Department of Social Management Engineering,
Graduate School of Engineering, Tottori University, Japan
toku@sse.tottori-u.ac.jp, yamada@sse.tottori-u.ac.jp*

Abstract

This paper discusses software performability evaluation considering the real-time property. We assume that the software system has two operational states from the viewpoint of the end users: one is operating with the desirable performance level according to specification and the other is with degraded performance level. The time-dependent behavior of the system is described by the Markovian software availability model with performance degradation. Assuming that the system can process the multiple tasks simultaneously, we analyze the distribution of the number of tasks whose processes can be completed within the processing time limit with the infinite server queueing model. We derive several software performability measures; these are given as the functions of time and the number of debugging activities. Finally, we illustrate several numerical examples of the measures to investigate the impact of consideration of the performance degradation on the system performability evaluation.

Keywords: *performability, real-time property, performance degradation, software availability model, infinite-server queueing model*

1. Introduction

The studies on performability evaluation methods for hardware-oriented computing systems have much been discussed [1,2,3,4]. However, on the other hand, most of studies on software-oriented reliability evaluation have treated only the inherent reliability characteristics such as the residual fault content, the mean time between software failures (MTBSF), and the software reliability function [5,6]. However, recently, software-conscious approaches extended to performability evaluation have also increased [7,8,9].

Most of the existing software-conscious approaches are discussed on the basis of performability measures in steady states and assume that the probabilistic or stochastic characteristics in system failure and restoration do not change even though the system is debugged or refreshed, i.e., the system returns to the initial condition in terms of the failure and restoration characteristics, neither better nor worse states. As to this point, the analytical framework in the above studies is basically similar to the hardware-conscious approach even though the authors of previous works often say that their works are software-oriented. Traditional stochastic software reliability modeling often considers the dynamic reliability/performance growth process. Musa [10] says that the above mention is one of main differences from the modeling for the hardware system.

In this paper, we discuss the user-oriented performability evaluation method for the software system considering the performance degradation in operation; this is the different approach from [7,8,9]. Most of traditional software availability models [11,12] often assume only up and down states and provide the probabilistic measures such as the instantaneous availability defined as the probability that the system is operating at a given time point; this value does not reflect the operational performance levels. Recently, it is often that the traditional measures, however, are not appropriate from the viewpoint of end users. As mentioned in [13,14], software intensive systems could not always display their peak performance or service, or some internal parts of systems might be unfavorable states even though they are available or do not seem to fall into operation stoppage outwardly in actual operational environment. For instance, the system is capable of decreasing throughput due to not only software aging but also the concentration of loads into some specified system resources. In the web-based system, end users may often perceive the performance degradation due to congestion of the network. As another case, some parts of system functions are unavailable due to maintenance of the corresponding software subcomponents [8]. We assume that there are two user-perceived operational states: one is providing with performance according to specification, i.e., a desirable operational state, and the other is with degraded service performance. In particular, we consider the real-time property; this is defined as the attribute that the system can complete the task within the stipulated response time limit [15]. Then the phenomenon of performance degradation, the dynamic software reliability growth, and the upward tendency of difficulty in debugging are described by the user-oriented Markovian software availability model [16]. Assuming that the process of the tasks arriving at the system follows the nonhomogeneous Poisson process (NHPP), we model the stochastic behavior of the number of tasks whose processes can be complete within the processing time limit with the infinite-server queueing theory [17].

The organization of the rest of the paper is shown as follows: Section 2 states the Markovian software availability model with performance degradation. Section 3 defines the operating regulation of the system and analyzes the distribution of the number of tasks whose processes are complete within the processing time limit up to a given time point. Section 4 derives several software performability measures from the model. The measures are given as the functions of time and the number of debuggings. Section 5 illustrates the numerical example of the measures and examines the software performability analysis. Finally, Section 6 summarizes the conclusion of the paper.

2. Software availability model with performance degradation

We make the following assumptions for software availability modeling with performance degradation based on the model of [16]:

- AI-1. When the software system is operating, the time-interval of operation with performance according to specification, T_s , and the holding time of performance degradation, T_d , follow the exponential distributions with means $1/\theta$ and $1/\eta$, respectively.
- AI-2. The software system breaks down and starts to be restored as soon as a software failure occurs, and the system cannot operate until the restoration action completes.

AI-3. The restoration action includes the debugging activity and software reliability growth occurs if a debugging activity is perfect.

AI-4. Consider the imperfect debugging environment where the debugging activity may fail, i.e., it is probabilistic whether the debugging activity succeeds or fails. The debugging activity is perfect with perfect debugging probability a ($0 < a < 1$), while imperfect with probability $b(=1-a)$. A perfect debugging activity corrects and removes one fault from the system.

AI-5. When n faults have been corrected, the next software failure-occurrence time-interval, U_n , and the restoration time, V_n , follow the exponential distributions with means $1/\lambda_n$ and $1/\mu_n$, respectively.

AI-6. T_s , T_d , U_n , and V_n are mutually independent.

We refer to the treatment of the probabilistic characteristics of T_s and T_d in this paper. As mentioned in the previous section, There exist various causes of performance degradation, for example, not only the internal factors such as software aging or temporal suspension of some functions due to restoration but also the external factors such as the congestion of the network or the concentration of the access to some system resources. From the viewpoint of end users, the phenomenon of the performance degradation is one of interesting issues, but users hardly care about the causes of performance degradation. Since we pay attention to the modeling from the user perspective, it is assumed that both of the occurrence of performance degradation and the retrieval from the performance degradation arise randomly.

We introduce a stochastic process $\{X(t), t \geq 0\}$ representing the user-perceived state of the software system at the time point t . The state space of the process $\{X(t), t \geq 0\}$ is defined as follows:

$W = \{W_n: n=0, 1, 2, \dots\}$: the system is operating with performance according to specification (desirable operational state),

$L = \{L_n: n=0, 1, 2, \dots\}$: the system is operating with degraded performance,

$R = \{R_n: n=0, 1, 2, \dots\}$: the system is utterly inoperable and in process of restoration,

where $n=0, 1, 2, \dots$ denotes the cumulative number of corrected faults. Figure 1 illustrates the sample state transition diagram of $X(t)$.

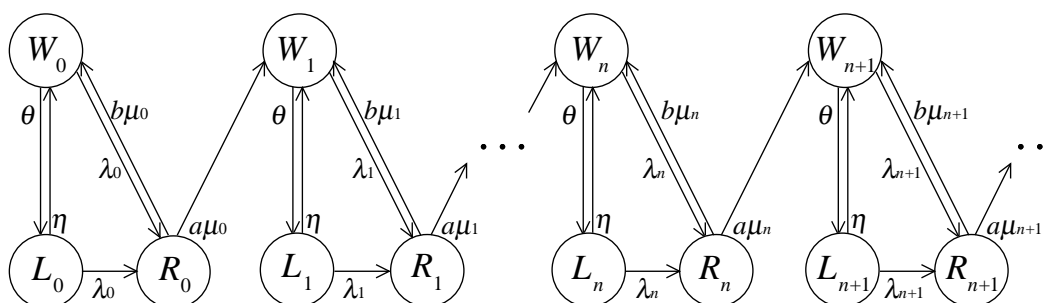


Figure 1. Sample state transition diagram of $X(t)$

Let $P_{W_i,A}(t) \equiv \Pr\{X(t) = A | X(0) = W_i\}$ ($A \in \{W_n, L_n, R_n\}$; $i, n=0, 1, 2, \dots; i \leq n$) be the state occupancy probability representing the conditional probability that the system is in state A at the time point t on the condition that the system was in state W_i at time point $t=0$ [18]. Then we can obtain $P_{W_i,A}(t)$'s as

$$P_{W_i,W_n}(t) \equiv \Pr\{X(t) = W_n | X(0) = W_i\}$$

$$= B_{i,n}^0 e^{-(\lambda_n + \theta + \eta)t} + \sum_{m=i}^n \left[B_{i,n}^1(m) e^{-d_m^1 t} + B_{i,n}^2(m) e^{-d_m^2 t} \right]$$

$$(i, n = 0, 1, 2, \dots; i \leq n), \quad (1)$$

$$\left. \begin{matrix} d_m^1 \\ d_m^2 \end{matrix} \right\} = \frac{1}{2} \left[(\lambda_m + \mu_m) \pm \sqrt{(\lambda_m + \mu_m)^2 - 4a\lambda_m\mu_m} \right] \text{ (double signs in same order),} \quad (2)$$

$$P_{W_i,R_n}(t) \equiv \Pr\{X(t) = R_n | X(0) = W_i\}$$

$$= \frac{g_{i,n+1}(t)}{a\mu_n} \quad (i, n = 0, 1, 2, \dots; i \leq n), \quad (3)$$

$$P_{W_i,L_n}(t) \equiv \Pr\{X(t) = L_n | X(0) = W_i\}$$

$$= G_{i,n}(t) - G_{i,n+1}(t) - P_{W_i,W_n}(t) - P_{W_i,R_n}(t) \quad (i, n=0, 1, 2, \dots; i \leq n), \quad (4)$$

respectively, where $B_{i,n}^0$, $B_{i,n}^1(m)$, and $B_{i,n}^2(m)$ in Eq. (1) are given by

$$B_{i,n}^0 = \frac{-\theta(\mu_n - \lambda_n - \theta - \eta) \prod_{j=i}^{n-1} d_j^1 d_j^2}{\prod_{j=i}^n (d_j^1 - \lambda_n - \theta - \eta)(d_j^2 - \lambda_n - \theta - \eta)} \quad (i, n=0, 1, 2, \dots; i \leq n), \quad (5)$$

$$B_{i,n}^1(m) = \frac{(\lambda_n + \eta - d_m^1)(\mu_n - d_m^1) \prod_{j=i}^{n-1} d_j^1 d_j^2}{(\lambda_n + \theta + \eta - d_m^1) \prod_{\substack{j=i \\ j \neq m}}^n (d_j^1 - d_m^1) \prod_{j=i}^n (d_j^2 - d_m^1)} \quad (m=i, i+1, \dots, n), \quad (6)$$

$$B_{i,n}^2(m) = \frac{(\lambda_n + \eta - d_m^2)(\mu_n - d_m^2) \prod_{j=i}^{n-1} d_j^1 d_j^2}{(\lambda_n + \theta + \eta - d_m^2) \prod_{\substack{j=i \\ j \neq m}}^n (d_j^2 - d_m^2) \prod_{j=i}^n (d_j^1 - d_m^2)} \quad (m=i, i+1, \dots, n), \quad (7)$$

respectively, $G_{i,n}(t)$ appearing in Eq. (4) is the distribution function of the first passage time of $X(t)$ from state W_i to state W_n ($i \leq n$), and $g_{i,n}(t) \equiv dG_{i,n}(t)/dt$ appearing in Eq. (3) is

the density function of $G_{i,n}(t)$ (see [16] for the details of the derivation processes of $P_{W_i,A}(t)$'s).

3. Model description and analysis for task processing

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

- AII-1. The number of tasks the system can process simultaneously is sufficiently large.
- AII-2. The process $\{N(t), t \geq 0\}$ representing the number of tasks arriving at the system up to the time t follows the NHPP with the arrival rate $\omega(t)$ and the mean value function $\mathcal{Q}(t) \equiv E[N(t)] = \int_0^t \omega(x) dx$.
- AII-3. Each task has a processing time limit, T_r , which follows a general distribution whose distribution function is denoted as $F_{T_r}(t) \equiv \Pr\{T_r \leq t\}$.
- AII-4. The processing times of a task in state **W**, Y_w , and in state **L**, Y_L , are distributed generally; their distribution functions are denoted as $F_{Y_w}(t) \equiv \Pr\{Y_w \leq t\}$ and $F_{Y_L}(t) \equiv \Pr\{Y_L \leq t\}$ ($E[Y_w] < E[Y_L]$), respectively. Each of the processing times is independent. The distribution of the processing time is determined by the state of the system at the time point when the corresponding task has just arrived at the system. In other words, once a task process starts, its distribution does not vary even though the state of the system in operation may change afterward. Y_w , Y_L , and T_r are mutually independent.
- AII-5. When the system causes a software failure in task processing or the processing times of tasks exceed the processing time limit, the corresponding tasks are canceled.

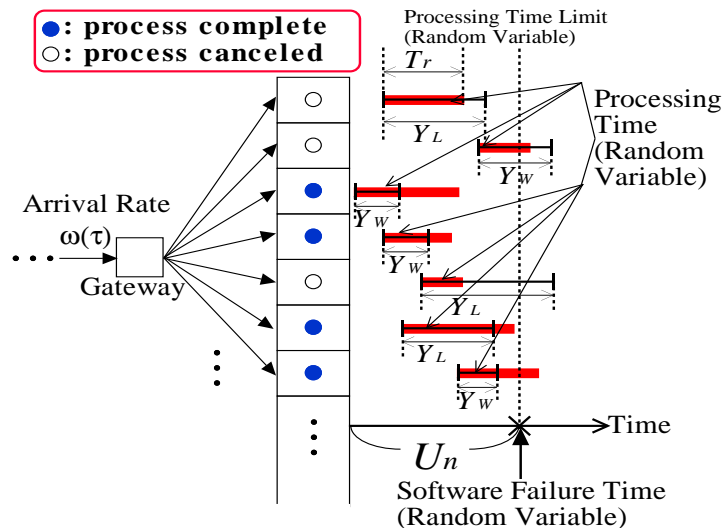


Figure 2. Configuration of task processing and relationship between process completion/cancellation and U_n , Y_L , Y_W , T_r

Here we derive the distribution of the number of tasks whose processes are complete within the processing time limit. Figure 2 illustrates the configuration of the system's task processing we consider. Hereafter, we set the time origin $t=0$ at the time point when the debugging activity is complete and i faults are corrected.

Let $\{Z_i^1(t), t \geq 0\}$ be the stochastic process representing the cumulative number of tasks whose processes can be complete within the processing time limit out of the tasks arriving up to the time t . By conditioning with $\{N(t)=k\}$, we obtain the probability mass function of $Z_i^1(t)$ as

$$\Pr\{Z_i^1(t) = j\} = \sum_{k=0}^{\infty} \Pr\{Z_i^1(t) = j \mid N(t) = k\} \cdot e^{-\Omega(t)} \frac{[\Omega(t)]^k}{k!}. \quad (8)$$

From Fig. 2, the probabilities that the process of an arbitrary task is complete within the processing time limit when the system is in state W_n and state L_n are given by

$$\beta_{W_n} \equiv \Pr\{Y_W < U_n, Y_W < T_r \mid X(t) = W_n\} = \int_0^{\infty} e^{-\lambda_n y} \overline{F_{T_r}}(y) dF_{Y_W}(y), \quad (9)$$

$$\beta_{L_n} \equiv \Pr\{Y_L < U_n, Y_L < T_r \mid X(t) = L_n\} = \int_0^{\infty} e^{-\lambda_n y} \overline{F_{T_r}}(y) dF_{Y_L}(y), \quad (10)$$

respectively, where we denote $\overline{F}(\cdot) \equiv 1 - F(\cdot)$. Furthermore, from the property of the NHPP, given $\{N(t)=k\}$, k arrival times of tasks are independent and identically distributed random variables having the following probability density function [17]:

$$f(x) = \frac{\omega(x)}{\Omega(t)} \quad (0 \leq x \leq t). \quad (11)$$

Therefore, the probability that the process of an arbitrary task having arrived up to the time t is complete within the processing time limit is obtained as

$$\begin{aligned} p_i^1(t) &= \int_0^t \sum_{n=i}^{\infty} [\Pr\{X(x) = W_n \mid X(0) = W_i\} \cdot \beta_{W_n} + \Pr\{X(x) = L_n \mid X(0) = W_i\} \cdot \beta_{L_n}] f(x) dx \\ &= \frac{1}{\Omega(t)} \sum_{n=i}^{\infty} \int_0^t [\beta_{W_n} P_{W_i, W_n}(x) + \beta_{L_n} P_{W_i, L_n}(x)] \omega(x) dx, \end{aligned} \quad (12)$$

from the infinite-server queueing theory [17]. Then from assumption AII-4,

$$\Pr\{Z_i^1(t) = j \mid N(t) = k\} = \begin{cases} \binom{k}{j} [p_i^1(t)]^j [1 - p_i^1(t)]^{k-j} & (j = 0, 1, 2, \dots, k) \\ 0 & (j > k) \end{cases}, \quad (13)$$

where $\binom{k}{j} \equiv k! / [(k-j)! j!]$ denotes the binomial coefficient. Equation (13) means that, given that $\{N(t)=k\}$, the number of tasks whose processes can be complete within the processing time limit follows the binomial process with mean $kp_i^1(t)$. Accordingly, from Eq. (8) the distribution of $Z_i^1(t)$ is given by

$$\begin{aligned} \Pr\{Z_i^1(t) = j\} &= \sum_{k=j}^{\infty} \binom{k}{j} [p_i^1(t)]^j [1 - p_i^1(t)]^{k-j} \times e^{-\Omega(t)} \frac{[\Omega(t)]^k}{k!} \\ &= e^{-\Omega(t)p_i^1(t)} \frac{[\Omega(t)p_i^1(t)]^j}{j!}. \end{aligned} \quad (14)$$

Equation (14) means that $\{Z_i^1(t), t \geq 0\}$ follows the NHPP with the mean value function $\Omega(t)p_i^1(t)$.

Letting $\{Z_i^2(t), t \geq 0\}$ be the stochastic process representing the cumulative number of tasks canceled out of ones arriving up to the time t , we can have a similar discussion on $\{Z_i^2(t), t \geq 0\}$, i.e., the distribution of $Z_i^2(t)$ is given by

$$\left. \begin{aligned} \Pr\{Z_i^2(t) = j\} &= e^{-\Omega(t)p_i^2(t)} \frac{[\Omega(t)p_i^2(t)]^j}{j!} \\ p_i^2(t) &= 1 - p_i^1(t) \end{aligned} \right\}. \quad (15)$$

Equation (15) means that $\{Z_i^2(t), t \geq 0\}$ follows the NHPP with the mean value function $\Omega(t)p_i^2(t)$.

4. Derivation of software performability measures

Based on the above analysis, we can obtain several measures for software performability evaluation considering the real-time property.

The expected number of tasks completable out of the tasks arriving up to the time t is given by

$$\Lambda_i^1(t) \equiv E[Z_i^1(t)] = \sum_{n=i}^{\infty} \int_0^t [\beta_{W_n} P_{W_i, W_n}(x) + \beta_{L_n} P_{W_i, L_n}(x)] \omega(x) dx. \quad (16)$$

Furthermore, the instantaneous task completion ratio is obtained as

$$v_i^1(t) \equiv \frac{d\Lambda_i^1(t)}{dt} / \omega(t) = \sum_{n=i}^{\infty} [\beta_{W_n} P_{W_i, W_n}(t) + \beta_{L_n} P_{W_i, L_n}(t)], \quad (17)$$

which represents the ratio of the number of tasks completed within the processing time limit to one arriving at the system per unit time at the time point t . As to $p_i^1(t)$ in Eq. (12), we can give the following interpretation:

$$p_i^1(t) = \frac{E[Z_i^1(t)]}{E[N(t)]}. \quad (18)$$

That is, $p_i^1(t)$ is the cumulative task completion ratio up to the time t which represents the expected proportion of the cumulative number of tasks completed within the processing time limit to one arriving at the system in the time-interval $(0, t]$.

For the number of tasks canceled out of ones arriving up to the time t , we can have the similar discussion, i.e., the expected number of tasks canceled up to the time t , the instantaneous and the cumulative task incompleteness ratios are given by

$$\Lambda_i^2(t) \equiv E[Z_i^2(t)] = \Omega(t) - \sum_{n=i}^{\infty} \int_0^t [\beta_{W_n} P_{W_i, W_n}(x) + \beta_{L_n} P_{W_i, L_n}(x)] \omega(x) dx, \quad (19)$$

$$\nu_i^2(t) \equiv \frac{d\Lambda_i^2(t)}{dt} / \omega(t) = 1 - \sum_{n=i}^{\infty} [\beta_{W_n} P_{W_i, W_n}(t) + \beta_{L_n} P_{W_i, L_n}(t)], \quad (20)$$

$$p_i^2(t) = \frac{E[Z_i^2(t)]}{E[N(t)]}, \quad (21)$$

respectively.

We should note that it is too difficult to use Eqs. (16)–(21) 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 faults corrected at time point $t=0$, cannot be observed immediately. However, the numbers of software failures or debugging activities can be easily observed. Furthermore, the cumulative number of faults corrected immediately after the completion of the l -th debugging activity, C_l , follows the binomial distribution whose probability mass function is given by

$$\Pr\{C_l = i\} = \Pr\{X(0) = W_i\} = \binom{l}{i} a^i b^{l-i} \quad (i=0, 1, 2, \dots, l). \quad (22)$$

Accordingly, we can convert Eqs. (16)–(21) into the functions of the number of debuggings, l , i.e., we obtain

$$\Lambda^1(t, l) = \sum_{i=0}^l \binom{l}{i} a^i b^{l-i} \Lambda_i^1(t), \quad (23)$$

$$\nu^1(t, l) = \sum_{i=0}^l \binom{l}{i} a^i b^{l-i} \nu_i^1(t), \quad (24)$$

$$p^1(t, l) = \frac{1}{\Omega(t)} \sum_{i=0}^l \binom{l}{i} a^i b^{l-i} \Lambda_i^1(t), \quad (25)$$

$$\Lambda^2(t, l) = \Omega(t) - \sum_{i=0}^l \binom{l}{i} a^i b^{l-i} \Lambda_i^1(t), \quad (26)$$

$$\nu^2(t, l) = 1 - \sum_{i=0}^l \binom{l}{i} a^i b^{l-i} \nu_i^1(t), \quad (27)$$

$$p^2(t, l) = 1 - \frac{1}{\Omega(t)} \sum_{i=0}^l \binom{l}{i} a^i b^{l-i} \Lambda_i^1(t), \quad (28)$$

respectively. Equations (23)–(28) represent the expected cumulative number of tasks completable, the instantaneous and the cumulative task completion ratios, the expected cumulative number of tasks canceled, the instantaneous and the cumulative task

incompletion ratios at the time point t , given that the l -th debugging is complete at time point $t=0$, respectively. In particular, we should note that Eqs. (24) and (27) are no bearing on $\mathcal{Q}(t)$, i.e., $v^1(t,l)$ and $v^2(t,l)$ are independent of the task arrival process.

5. Numerical examples

Here we apply the model of Moranda [19] to the hazard rate λ_n and the restoration rate μ_n in the numerical example, i.e., $\lambda_n \equiv Dc^n$ ($D>0, 0<c<1$) and $\mu_n \equiv Er^n$ ($E>0, 0<r\leq 1$), respectively. Furthermore, we use the following values of the model parameters shown in Ref. [20]:

$$\hat{D} = 0.245, \quad \hat{c} = 0.940, \quad \hat{E} = 1.114, \quad \hat{r} = 0.960,$$

where we set $a=0.8$ (see [20] for the detail of parameter estimation). For the distributions of the processing times in state \mathbf{W} , $F_{Y_W}(t)$, and in state \mathbf{L} , $F_{Y_L}(t)$, and the processing time limit, $F_T(t)$, we apply the gamma distribution of order two denoted by

$$F_I(t) \equiv H(t | \alpha_I) = 1 - (1 + \alpha_I t)e^{-\alpha_I t} \quad (t \geq 0; \alpha_I \geq 0; I \in \{Y_W, Y_L, T_r, \}), \quad (29)$$

where α_I denotes the scale parameter, and the mean and the variance are given by $2/\alpha_I$ and $2/\alpha_I^2$, respectively.

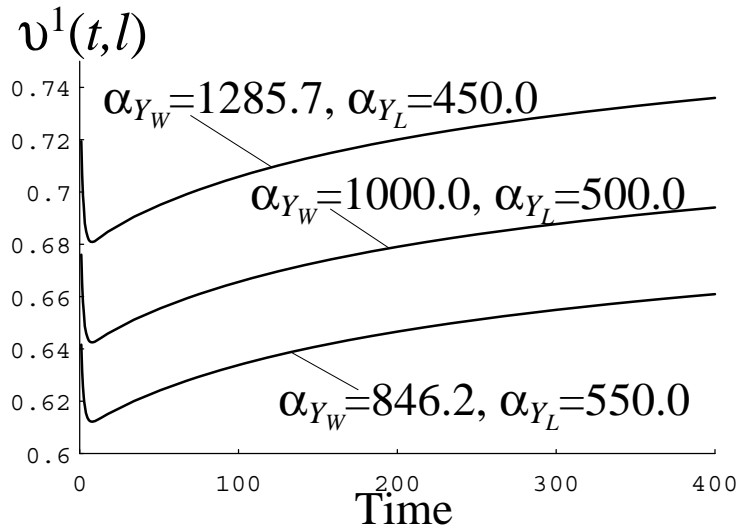


Figure 3. Dependence of $v^1(t,l)$ on the values of α_{Y_W} and α_{Y_L} , given

$$1/\alpha_{Y_W} + 1/\alpha_{Y_L} = 0.003 \quad (l=0, \theta=5.0, \eta=48.0, \alpha_T=400.0)$$

Figure 3 shows the dependence of the instantaneous task completion ratio, $v^1(t,l)$, in Eq. (24) on the values of α_{Y_W} and α_{Y_L} , given the arithmetic average of the means of the

processing times, i.e., the value of $(2/\alpha_{Y_W} + 2/\alpha_{Y_L})/2$ is constant. This figure displays that the case where the difference of performance levels between state **W** and state **L** becomes smaller decreases the performability evaluation of the whole system.

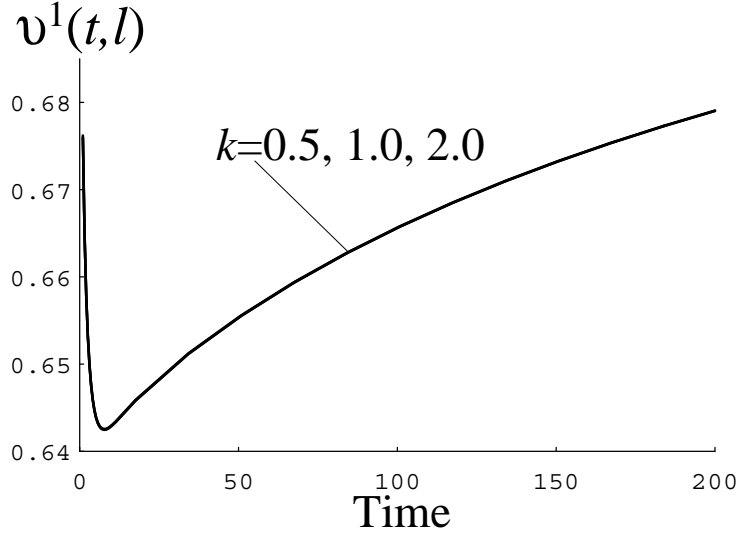


Figure 4. Dependence of $v^1(t, l)$ on the values of θ and η , given $\theta/\eta=5/48$
 ($l=0, \alpha_{Y_W}=1000.0, \alpha_{Y_L}=500.0, \alpha_{T_r}=400.0$)

Without software failure-occurrence, we might consider only two states; operational states with the desirable performance level, denoted as state W_∞ , and with the degraded performance level, denoted as state L_∞ . Then the state occupancy probabilities are given by

$$P_{W_\infty, W_\infty}(t) \equiv \Pr\{X(t) = W_\infty \mid X(0) = W_\infty\} = \frac{1}{1+\nu} + \frac{\nu}{1+\nu} e^{-\eta(1+\nu)t}, \quad (30)$$

$$P_{W_\infty, L_\infty}(t) \equiv \Pr\{X(t) = L_\infty \mid X(0) = W_\infty\} = \frac{\nu}{1+\nu} [1 - e^{-\eta(1+\nu)t}], \quad (31)$$

respectively, where we denote $\nu \equiv \theta/\eta$, and the instantaneous and the limiting task completion ratios can be obtained as

$$v^1_\infty(t) \equiv \beta_{W_\infty} P_{W_\infty, W_\infty}(t) + \beta_{L_\infty} P_{W_\infty, L_\infty}(t) = \frac{\beta_{W_\infty} + \beta_{L_\infty} \nu}{1+\nu} + \frac{(\beta_{W_\infty} - \beta_{L_\infty}) \nu}{1+\nu} e^{-\eta(1+\nu)t}$$

$$\left(\beta_{W_\infty} = \int_0^\infty \overline{F_{T_r}}(y) dF_{Y_W}(y), \quad \beta_{L_\infty} = \int_0^\infty \overline{F_{T_r}}(y) dF_{Y_L}(y), \quad \beta_{W_\infty} > \beta_{L_\infty} \right), \quad (32)$$

$$v^1_\infty \equiv \lim_{t \rightarrow \infty} v^1_\infty(t) = \frac{\beta_{W_\infty} + \beta_{L_\infty} v}{1 + v}, \quad (33)$$

respectively. From the form of Eq. (32), $v^1_\infty(t)$ is a decreasing function of t . Furthermore, Eqs. (32) and (33) imply that the task completion ratio depends on the ratio of θ and η rather than the individual values of θ and η , and that the smaller values of both of θ and η with their ratio constant increase the value of $v^1_\infty(t)$ and converge it to $v^1_\infty(t)$ slower. However, the performability evaluation considering software failure-occurrence in this paper is different from the above mention. Figure 4 shows the dependence of $v^1(t, l)$ on the value of k , given θ/η is constant, where we set $\theta_0=5.0$ and $\eta_0=48.0$, then $\theta=k\theta_0$ and $\eta=k\eta_0$. This figure displays that $v^1(t, l)$ is an increasing function of t except in the short time interval immediately after the beginning of operation; this is different from the above mention. On the other hand, Fig. 4 tells us that the behavior of $v^1(t, l)$ is almost same in any case of k ; i.e., the software performability evaluation almost depends on only the value of θ/η , not individual values of θ and η ; this is same tendency as the case without software failure-occurrence. As another point of view, Fig. 4 seems to show that $v^1(t, l)$ converges to a certain value; this is expected to be v^1_∞ since $t \rightarrow \infty$ leads to $n \rightarrow \infty$, then $\lambda_n \rightarrow 0$, i.e., the possibility of software failure-occurrence becomes vanishingly small as $t \rightarrow \infty$.

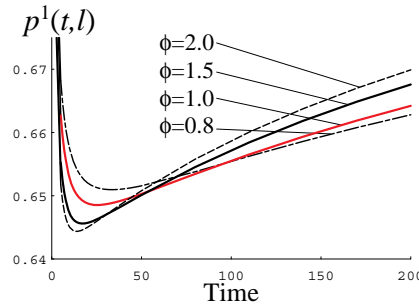


Figure 5. Dependence of $p^1(t, l)$ on the value of ϕ ($l=0$, $\xi=1.0$, $\alpha_{Y_w}=1000.0$, $\alpha_{Y_L}=500.0$, $\alpha_{T_r}=400.0$)

For the mean value function of $\{N(t), t \geq 0\}$, we apply the Weibull process, i.e., $E[N(t)] \equiv \Omega(t) = \xi t^\phi$ ($t \geq 0$; $\xi > 0$, $\phi > 0$). Figure 5 shows the dependence of the cumulative task completion ratio, $p^1(t, l)$, in Eq. (25) on parameter ϕ ; this reflects the intensity of the task arrival process. In this case, it is noted that $p^1(t, l)$ is independent of the value of ξ from the form of Eq. (18). We can see that the software performability evaluation rises with the increasing ϕ after a certain lapse of time, i.e., the higher task arrival rate results in the higher performability evaluation although the opposite tendency appears in the short time interval immediately after the beginning of the system operation in this figure. In other words, the performability evaluation based on $p^1(t, l)$ or $p^2(t, l)$ are susceptible to the task arrival process, whereas $v^1(t, l)$ or $v^2(t, l)$ are independent of the task arrival process from the form of Eqs. (24) or (27).

6. Concluding remarks

In this paper, we have constructed the stochastic performability evaluation model for the software system with processing time limit, considering the performance degradation in system operation. 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 with the concept of the infinite-server queueing model. From the model, we have derived several software performability measures considering the real-time property. They have been given as the functions of time and the number of debuggings. We have also illustrated the numerical example of these measures and investigated the impacts of the reliability growth, the degradation of the performance level, and the task arrival characteristics on the software performability evaluation.

Acknowledgments

This work was supported in part by Grants-in-Aid for Scientific Research (C) of Japan Society for the Promotion of Science under Grant No. 20510136 and Takahashi Industrial and Economic Research Foundation.

References

- [1] M. D. Beaudry, "Performance-related reliability measures for computing systems", IEEE Transactions on Computers, Vol. C-27, No. 6, pp. 540-547, June 1978.
- [2] J. F. Meyer, "On evaluating the performability of degradable computing systems", IEEE Transactions on Computers, Vol. C-29, No. 8, pp. 720-731, Aug. 1980.
- [3] M. Nakamura and S. Osaki, "Performance/reliability evaluation of a multi-processor system with computational demands", International Journal of Systems Sciences, Vol. 15, No. 1, pp. 95-105, Aug. 1984.
- [4] A. Sols, "System degraded availability", Reliability Engineering & System Safety, Vol. 56, No. 1, pp. 91-94, April 1997.
- [5] M. R. Lyu, ed., Handbook of Software Reliability Engineering, IEEE CS Press, McGraw-Hill, Los Alamitos, California, 1996.
- [6] S. Yamada, "Software reliability models", in Stochastic Models in Reliability and Maintenance (S. Osaki, ed.), Chapter 10, pp. 253-280, Springer-Verlag, Berlin Heidelberg, 2002.
- [7] H. Okamura, S. Miyahara and T. Dohi, "Dependability analysis of a transaction-based multi-server system with rejuvenation", IEICE Transactions on Fundamentals, Vol. E86-A., No. 8, pp. 2081-2090, Aug. 2003.
- [8] H. Eto and T. Dohi, "Analysis of a service degradation model with preventive rejuvenation", in Service Availability, 3rd International Service Availability Symposium, ISAS 2006, Helsinki, Finland, (D. Penkler, M. Reitenspiess and F. Tam, eds.), LNCS 4328, pp. 17-29, Springer-Verlag, Berlin, May 2006.
- [9] H.-P. Schwefel and I. Antonios, "Performability models for multi-server systems with high-variance repair durations", in 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, pp. 770-779, 2007.
- [10] J. D. Musa, Software Reliability Engineering, McGraw-Hill, New York, 1999.
- [11] J. H. Kim, Y. H. Kim and C. J. Park, "A modified Markov model for the estimation of computer software performance", Operations Research Letters, Vol. 1, No. 6, pp. 253-257, Dec. 1982.
- [12] K. Tokuno and S. Yamada, "Markovian software availability measurement based on the number of restoration actions", IEICE Transactions on Fundamentals, Vol. E83-A, No. 5, pp. 835-841, May 2000.
- [13] A. Pfening, S. Garg, A. Puliafito, M. Telek and K. S. Trivedi, "Optimal software rejuvenation for tolerating soft failures", Performance Evaluation, Vols. 27-28, pp. 491-506, Oct. 1996.
- [14] S. Garg, A. Puliafito, M. Telek and K. S. Trivedi, "Analysis of preventive maintenance in transactions based software systems", IEEE Transactions on Computers, Vol. 47, No. 1, pp. 96-107, Jan. 1998.
- [15] J. K. Muppala, S. P. Woolet and K. S. Trivedi, "Real-time-systems performance in the presence of failures", Computer, Vol. 24, No. 5, pp. 37-47, 1991.

- [16] K. Tokuno and S. Yamada, "User-perceived software service availability modeling with reliability growth", in Service Availability, 5th International Service Availability Symposium, ISAS 2008, Tokyo, Japan, (T. Nanya, F. Maruyama, A. Pataricza and M. Malek, eds.), LNCS 5017, pp. 75-89, Springer-Verlag, Berlin, May 2008.
- [17] S. M. Ross, Introduction to Probability Models, 9th Edition, Academic Press, San Diego, 2007.
- [18] S. Osaki, Applied Stochastic System Modeling, Springer-Verlag, Heidelberg, 1992.
- [19] P. B. Moranda, "Event-altered rate models for general reliability analysis", IEEE Transactions on Reliability, Vol. R-28, No. 5, pp. 376-381, Dec. 1979.
- [20] K. Tokuno and S. Yamada, "Stochastic performance evaluation for multi-task processing system with software availability model", Journal of Quality in Maintenance Engineering, Vol. 12, No. 4, pp. 412-424, 2006.

Authors



Koichi Tokuno received the B.S.E. and M.S. degrees from Hiroshima University, Japan in 1990 and 1992, respectively, and the Ph.D. degree from Tottori University, Japan in 1999. From 1992 to 1994, he worked at the Japan Steel Works Ltd., Hiroshima Prefecture, Japan. From 1994 to 2001, he was an assistant professor at Tottori University, Japan. Since 2001, he has been working as an associate professor at the Graduate School of Engineering, Tottori University, Tottori-shi, Japan. His research area includes software reliability engineering, stochastic modeling, and system dependability measurement and assessment. 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 Reliability Engineering Association of Japan and the Japan Society for Software Science and Technology. Dr. Tokuno is serving as a Regional Editor of the International Journal of Reliability and Applications.



Shigeru Yamada received the B.S.E., M.S., and Ph.D. degrees from Hiroshima University in 1975, 1977, and 1985, respectively. Since 1993, he has been working as a professor at the Graduate School of Engineering, 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 *Introduction to Software Management Model* (Kyoritsu Shuppan, 1993), *Software Reliability Models: Fundamentals and Applications* (JUSE, Tokyo, 1994), *Statistical Quality Control for TQM* (Corona Publishing, Tokyo, 1998), *Software Reliability: Model, Tools, Management* (The Society of Project Management, 2004), and *Quality-Oriented Software Management* (Morikita Shuppan, 2007). 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, the Paper Award from the Reliability Engineering Association of Japan in 1999, the International Leadership Award in Reliability Engg. Research from the ICQRIT/SREQOM in 2003, the Best Paper Award from the Society of Project Management in 2006, the Leadership Award from the ISSAT in 2007, and the Outstanding Paper Award at the IEEE-IEEM2008. 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.