The Korean Journal of Applied Statistics (2008)
21(6), 997–1006

Performance Evaluation of Software Task Processing Based on Markovian Perfect Debugging Model

Chong Hyung Lee[1] · Kyu Beom Jang[2] · Dong Ho Park[3]

[1]Dept. of Hospital Management, Konyang University;
[2]Div. of Mathematical Information Science, Hallym University;
[3]Div. of Mathematical Information Science, Hallym University

(Received August 2008; accepted October 2008)

Abstract

This paper proposes a new model by combining an infinite-server queueing model for multi-task processing software system with a perfect debugging model based on Markov process with two types of faults suggested by Lee *et al.* (2001). We apply this model for module and integration testing in the testing process. Also, we compute several measure, such as the expected number of tasks whose processes can be completed and the task completion probability are investigated under the proposed model.

Keywords: Markovian perfect debugging model, software tasking process.

1. Introduction

Most of software reliability models derived under non-homogeneous Poisson process(NHPP) are based on the assumption that whenever a failure occurs, the fault which caused the failure is immediately removed. Such models assume, in general, that the debugging time is negligible. However, the debugging of failure always takes certain length of time in practice. Thus, it is more realistic to consider the debugging time as a random variable when developing a stochastic model which is applicable in real field. To derive a stochastic model that can take into account the random debugging time of failure, the Markov process has been widely used.

Shooman and Trivedi (1976) propose a software availability model based on a Markov process. Laprie (1984) proposes several evaluation methods of reliability and availability of the software systems during its operation periods, while most of the researches are focused on the measures during the test period. Lee *et al.* (2001) have proposed a Markovian perfect debugging model for

[1]Professor, Dept. of Hospital Management, Konyang University, 26 Nae-dong, Nonsan, Chungnam 320-711, Korea. E-mail: chlee@konyang.ac.kr
[2]Doctoral student, Div. of Mathematical Information Science, Hallym University, 39 Hallymdaehak-gil, Chuncheon, Gangwon-do 200-702, Korea. E-mail: tzarm@hallym.ac.kr
[3]Corresponding author: Professor, Div. of Mathematical Information Science, Hallym University, 39 Hallymdaehak-gil, Chuncheon, Gangwon-do 200-702, Korea. E-mail: dhpark@hallym.ac.kr

which is difficult to detect. Tokuno and Yamada (2006) have proposed the software performance evaluation method based on the multiple tasks.

We propose a testing method for a software consisting of modules in the testing phase of software development. Developed modules are assigned to process tasks and tested to be satisfied at the specified level. In the following phase, a integrated software consisting of modules is tested. That is, testing tasks are assigned simultaneously at all modules of software.

This paper is structured as follows. Section 2 introduces the perfect debugging model based on Markov process with two types of faults proposed by Lee *et al.* (2001) and suggest performance evaluation methods for software module tasking process and software tasking process. Section 3 illustrates our models in the context of simulated data. Section 4 summarizes our results.

2. Model Formulation

2.1. Assumptions

The following assumptions are considered for perfect debugging modelling:

1. The software system becomes unavailable and starts to be restored as soon as the software failure occurs and the system cannot operate until the debugging action is complete.
2. Nonworking state can be classified into two types. One type is caused by a fault that is easily detected and the other is caused by a fault that is difficult to detect.
3. The debugging process starts immediately when a failure occurs and the perfect debugging removes exactly one fault for either fault type. The probability that two or more software failures occur simultaneously is negligible.

For the system's task processing, the followings are also assumed.

1. The number of tasks that the system can process simultaneously is sufficiently large.
2. The process $\{N_k(t),\, t \geq 0\}$ which represents the number of tasks for the k^{th} module arriving at the system up to time t follows a homogeneous Poisson process with the arrival rate θ_k. The process $\{N_u(t),\, t \geq 0\}$ representing the number of task sets arriving at the integrated system up to time t follows also a homogeneous Poisson process with θ_u.
3. When the failure occurs before the process of a certain task is completed, the task is cancelled. The processing times of tasks are assumed to be independent.
4. In the integration testing, task sets assigned for each modules are independent.

We denote the perfect debugging model of Lee *et al.* (2001) by LNP model.

2.2. LNP model

Consider a stochastic process $\{X(t),\, t \geq 0\}$ which represents both the total number of removed faults up to time t and the state of software system at a time t, which is classified as working or nonworking during its testing and operation period. Non working state can be further classified into two types. One type is caused by a fault that is easily detected and the other is caused by a fault that is difficult to detect. The former type is referred to as fault type 1 and the latter is referred to as fault type 2. Then, the state of software system is defined by $X(t) = (x_1(t),\, x_2(t))$,

where $x_1(t)$ is a total number of faults removed during a time interval $(0, t]$ and at time t,

$$x_2(t) = \begin{cases} 0, & \text{working,} \\ 1, & \text{nonworking by fault type 1,} \\ 2, & \text{nonworking by fault type 2.} \end{cases}$$

Let i denote the number of faults removed during a time interval $(0, t]$ and let $T_{i,e}$ and $T_{i,d}$ be the random variables representing the waiting times elapsed for the occurrence of fault types 1 and 2, respectively, for the software system which is just returned to the working state after removing the i^{th} fault. We assume that $T_{i,e}$ and $T_{i,d}$ follow the exponential distributions with means, $1/\mu_{i,1}$ and $1/\mu_{i,2}$, respectively and they are mutually independent. Let $T_{i,j}$, $j = 1, 2$, denote the random variables representing the lengths of time needed to remove a fault type j from the software system, which had i faults removed previously. We assume that they follow the exponential distributions with means of $1/\theta_{i,j}$. It is quite reasonable to assume that there repair time becomes shorter and the length of time that the software is in working state is getting longer as the number of removed faults gets larger. Thus, for each i and j, we assume that $\mu_{i,j}$ and $\theta_{i,j}$ are the decreasing and increasing functions of the number of removed faults, respectively. Let $F_{\boldsymbol{\alpha},\boldsymbol{\beta}}(t)$ be one step transition probability that the process $\{X(t),\ t \geq 0\}$ in state $\boldsymbol{\alpha}$ will be in state $\boldsymbol{\beta}$ after time t. The expression for $F_{\boldsymbol{\alpha},\boldsymbol{\beta}}(t)$s are obtained as follows.

$$\begin{aligned} F_{(i,0),(i,j)}(t) &= \frac{\mu_{i,j}}{\mu_{i,1} + \mu_{i,2}} \left[1 - \exp\{-(\mu_{i,1} + \mu_{i,2})t\}\right], \\ F_{(i,j),(i+1.0)}(t) &= 1 - \exp(-\theta_{i,j}t). \end{aligned} \tag{2.1}$$

Let $T_{(i,0),(n,0)}$ be a random variable representing the first passage time of the software system with i faults already removed until the number of faults removed reaches n, where $i < n$. Let $G_{(i,0),(n,0)}(t)$ be the distribution function of $T_{(i,0),(n,0)}$. Then, the distribution function can be expressed as

$$G_{(i,0),(n,0)}(t) = \Pr(T_{(i,0),(n,0)} \leq t) = \sum_{j=1}^{2} F_{(i,0),(i,j)} * F_{(i,j),(i+1,0)} * G_{(i+1,0),(n,0)}(t), \tag{2.2}$$

where $i = 0, 1, \ldots, n-1$, $n = 1, 2, \ldots, N$ and * symbolizes the Stieltjes convolution and $G_{(n,0),(n,0)}(t) = 1$. Here, N denotes the total number of faults latent in the system at $t = 0$. To derive the distribution function of $T_{(i,0),(n,0)}$. We may apply the Laplace-Stieljes transform and consider the special case when $i = 0$. Then, the distribution of $T_{(i,0),(n,0)}$ can be obtained as

$$G_{(0,0),(n,0)}(t) = 1 - \sum_{i=0}^{n-1} \left\{N_{n,i,1}\exp(-x_i t) + N_{n,i,2}\exp(-\theta_{i,1}t) + N_{n.i,3}\exp(-\theta_{i,2}t)\right\}, \tag{2.3}$$

for $t \geq 0$. Denoting $x_i = \mu_{i,1} + \mu_{i,2}$, the values of Ns are defined as

$$N_{n,i,1} = \frac{\prod_{m=0}^{n-1} \{\theta_{m,1}\theta_{m,2}x_m - (\mu_{m,1}\theta_{m,1} + \mu_{m,2}\theta_{m,2})x_i\}}{x_i \prod_{\substack{m=0 \\ m \neq i}}^{n-1} (x_m - x_i) \prod_{m=0}^{n-1} \{(\theta_{m,1} - x_i)(\theta_{m,2} - x_i)\}},$$

$$N_{n,i,2} = \frac{\prod_{m=0}^{n-1} \{\theta_{m,1}\theta_{m,2}x_m - (\mu_{m,1}\theta_{m,1} + \mu_{m,2}\theta_{m,2})\theta_{i,1}\}}{\theta_{i,1} \prod_{\substack{m=0 \\ m\neq i}}^{n-1} (\theta_{m,1} - \theta_{i,1}) \prod_{m=0}^{n-1} \{(x_m - \theta_{i,1})(\theta_{m,2} - \theta_{i,1})\}},$$

$$N_{n,i,3} = \frac{\prod_{m=0}^{n-1} \{\theta_{m,1}\theta_{m,2}x_m - (\mu_{m,1}\theta_{m,1} + \mu_{m,2}\theta_{m,2})\theta_{i,2}\}}{\theta_{i,2} \prod_{\substack{m=0 \\ m\neq i}}^{n-1} (\theta_{m,2} - \theta_{i,2}) \prod_{m=0}^{n-1} \{(x_m - \theta_{i,2})(\theta_{m,1} - \theta_{i,2})\}},$$

where $\prod_{\substack{m=0 \\ m\neq i}}^{0} = 1$.

Next, we consider the working probability that the software system is in state $(n, 0)$ at time t on condition that the system was in state $(i, 0)$ at time 0. Let $p_{(i,0),(n,0)}(t)$ be the probability that a software is in working state at time t after n faults had been removed, given that the software is in working state at time 0 with i faults already removed. Accordingly, the probability can be written as

$$p_{(i,0),(n,0)}(t) = G_{(i,0),(n,0)} * p_{(n,0),(n,0)}(t), \tag{2.4}$$

where $i, n = 0, 1, \ldots, N$, $i \leq n$ and

$$p_{(n,0),(n,0)}(t) = 1 - \sum_{j=1}^{2} F_{(n,0),(n,j)}(t), \tag{2.5}$$

for $n = 0, 1, \ldots, N-1$. As a special case, when $i = 0$, the working probability can be calculated as

$$\begin{aligned} p_{(0,0),(n,0)}(t) &= G_{(0,0),(n,0)}(t) - \sum_{i=0}^{n} M_{n,i,1}\{1 - \exp(-x_i t)\} \\ &\quad - \sum_{i=0}^{n-1} \left[M_{n,i,2}\{1 - \exp(-\theta_{i,1}t)\} + M_{n,i,3}\{1 - \exp(\theta_{i,2}t)\}\right], \end{aligned} \tag{2.6}$$

for $t \geq 0$ by applying the Laplace-Stieljes transform and inverse transformation. The values of Ms represent the followings.

$$M_{n,i,1} = N_{n,i,1} \cdot \frac{x_n}{x_n - x_i},$$

$$M_{n,i,2} = N_{n,i,2} \cdot \frac{x_n}{x_n - \theta_{i,1}},$$

$$M_{n,i,3} = N_{n,i,3} \cdot \frac{x_n}{x_n - \theta_{i,2}} \quad \text{and}$$

$$M_{n,n,1} = \frac{\prod_{m=0}^{n-1} \{\theta_{m,1}\theta_{m,2}x_m - (\mu_{m,1}\theta_{m,1} + \mu_{m,2}\theta_{m,2})x_n\}}{\prod_{m=0}^{n-1} \{(x_m - x_n)(\theta_{m,1} - x_n)(\theta_{m,2} - x_n)\}}.$$

2.3. Task processing model

In general, most of the commercial software products are complex and are composed of a number of modules. In the testing phase of software development, developed modules are assigned to process tasks and tested to be satisfied at the specified level. In the following phase, a integrated software consisting of modules is tested. Also, testing tasks inspecting performance of an integrated software are assigned simultaneously at all modules of software. But, most of existing research have not considered the module testing. Thus, It is more realistic to consider the module testing and the integration testing all together.

Tokuno and Yamada (2006) considered a multi-task software system which can process the plural tasks simultaneously. The stochastic behavior of the multiple tasks whose processes can be completed is modelled with an infinite server queueing model. In this paper, we consider both software modules testing and software testing in evaluating software performance. Also, a new model for the task processing which evaluate software module tasking process and software tasking process based on LNP model is proposed.

2.3.1. Task evaluation of modules Firstly, we discuss the task processing model for each modules. Let a counting process $\{N_k(t),\ t \geq 0\}$ be the random variable representing the number of tasks arriving at the $k(=1,\ldots,r)^{th}$ module of system up to time t and a counting process $\{Z_k(t),\ t \geq 0\}$ be the one denoting the cumulative number of tasks for the k^{th} module whose processes can be completed out of the tasks arriving up to time t. The distribution function of $Z_k(t)$ is given by

$$\Pr\{Z_k(t) = m_k\} = \sum_{l_k=0}^{\infty} \Pr\{Z_k(t) = m_k \mid N_k(t) = l_k\} \cdot \Pr\{N_k(t) = l_k\}. \tag{2.7}$$

The probability that m_k tasks for the k^{th} module among l_k tasks arrived up to t is completed can be calculated as

$$\Pr\{Z_k(t) = m_k \mid N_k(t) = l_k\} = \binom{l_k}{m_k} \{p_k(t)\}^{m_k} \{1 - p_k(t)\}^{l_k - m_k}, \tag{2.8}$$

where $p_k(t)$ is the probability that the process of a task for the k^{th} module arrived up to the time t is completed.

Let Y_k and $F_k(t)$ be the random variables representing the processing time of a task for the k^{th} module and the distribution function of Y_k, respectively. Also, let X_n be the random variable representing the waiting time elapsed for the occurrence of the $(n+1)^{th}$ fault and let $X_k(t)$ represent the $X(t)$ of the k^{th} module. Then, the probability that the process of an arbitrary task for the k^{th} module is completed on condition that $\{X_k(t) = (n,\ 0)\}$ can be obtained as

$$\begin{aligned} \beta_n^k &= \Pr\{Y_k < X_n \mid X_k(t) = (n,\ 0)\} \\ &= \Pr\{Y_k < \min(T_{n,e},\ T_{n,d}) \mid X_k(t) = (n,\ 0)\} \\ &= \int_0^{\infty} f_k(x) \exp\{-(\mu_{n,1} + \mu_{n,2})x\} dx, \end{aligned} \tag{2.9}$$

where $f_k(x) = dF_k(x)/dx$.

Note also follows that the arrival time of an arbitrary task out of all tests for the k^{th} module arriving up to time t is distributed uniformly over the time interval (Ross, 1972). Therefore $p_k(t)$ can be

written as

$$p_k(t) = \int_0^t \sum_{n=0}^{N} \Pr\{X_k(x) = (n, 0)\} \cdot \Pr\{Y_k < X_n \mid X_k(t) = (n, 0)\} \frac{dx}{t}$$
$$= \frac{1}{t} \sum_{n=0}^{N} \left\{ \beta_n^k \int_0^t p_{(0,0),(n,0)}(x) dx \right\}. \tag{2.10}$$

Accordingly, from (2.7) and Assumption 2, the distribution function of $Z_k(t)$ can be obtained as

$$\Pr\{Z_k(t) = m_k\} = \sum_{l_k=0}^{\infty} \binom{l_k}{m_k} \{p_k(t)\}^{m_k} \{1 - p_k(t)\}^{l_k - m_k} \cdot \frac{e^{-\theta_k t}(\theta_k t)^{l_k}}{l_k!}$$
$$= e^{-\theta_k \cdot t \cdot p_k(t)} \frac{\{\theta_k \cdot t \cdot p_k(t)\}^{m_k}}{m_k!}. \tag{2.11}$$

This equation is equivalent to the NHPP with mean value function $\theta_k \cdot t \cdot p_k(t)$. The expected number of tasks completable out of the tasks arriving up to the time t is given by

$$H_k(t) \equiv E\{Z_k(t)\} = \theta_k \cdot \sum_{n=0}^{N} \left\{ \beta_n^k \int_0^t p_{(0,0),(n,0)}(x) dx \right\}. \tag{2.12}$$

2.3.2. Task evaluation of software We consider the model for integration testing which is performed to overall system consisting of several modules. In this model, the tasks to be performed in the integration testing consist of a task set including the tasks for each module testing. A task set is considered as a test case to be able to severally test all modules in software. If we let Y_u be the random variable representing the processing time of a task set, then $Y_u = \max(Y_1, \ldots, Y_r)$.

Let a counting process $\{N_u(t),\ t \geq 0\}$ be the random variable representing the number of task sets arriving at the integrated software system up to time t and a counting process $\{Z_u(t),\ t \geq 0\}$ be the cumulative number of task sets whose processes can be completed out of the test sets arriving up to time t. Also, let $X_u(t)$ represent the $X(t)$ of the integrated software system. Then, the probability that the process of an arbitrary task set is completed on condition that $\{X_u(t) = (n, 0)\}$ is obtained as

$$\beta_n = \Pr\{Y_u < X_n \mid X_u(t) = (n, 0)\}$$
$$= \Pr\{\max(Y_1, \ldots, Y_r) < X_n \mid X_u(t) = (n, 0)\} = \prod_{k=1}^{r} \beta_n^k. \tag{2.13}$$

Thus, by replacing β_n^k of (2.10) by β_n, the probability that the process of a task set arrived up to the time t is completed can be written as

$$p_u(t) = \frac{1}{t} \sum_{n=0}^{N} \left\{ \left(\prod_{k=1}^{r} \beta_n^k \right) \cdot \int_0^t p_{(0,0),(n,0)}(x) dx \right\}. \tag{2.14}$$

Let l be the total number of task sets being arrived for processing by the software system. Then, similar to Equation (2.11) the distribution function of $Z_u(t)$ can be obtained as

$$\Pr\{Z_u(t) = m\} = \sum_{l=0}^{\infty} \binom{l}{m} \{p_u(t)\}^m \{1 - p_u(t)\}^{l-m} \cdot \frac{e^{-\theta_u t}(\theta_u t)^l}{l!}$$
$$= e^{-\theta_u \cdot t \cdot p_u(t)} \frac{\{\theta_u \cdot t \cdot p_u(t)\}^m}{m!}. \tag{2.15}$$

Proceeding just as we did in the previous case when we consider the cumulative number of task sets whose processes can be completed out of the task sets arriving up to time t, it can be shown that $Z_u(t)$ follows a NHPP with the mean value function which is given by

$$\theta_u \cdot \sum_{n=0}^{N} \left\{ \left(\prod_{k=1}^{r} \beta_n^k \right) \cdot \int_0^t p_{(0,0),(n,0)}(x)dx \right\}. \tag{2.16}$$

The expected number of tasks completable out of the tasks arriving up to the time t can be calculated similarly to Equation (2.12). Thus, we obtain

$$H(t) \equiv E\{Z_u(t)\} = \theta_u \cdot \sum_{n=0}^{N} \left\{ \left(\prod_{k=1}^{r} \beta_n^k \right) \cdot \int_0^t p_{(0,0),(n,0)}(x)dx \right\}, \tag{2.17}$$

$$h(t) \equiv \frac{H(t)}{dt} \Big/ \theta_u, \tag{2.18}$$

where $h(t)$ is the instantaneous task completion ratio.

3. Numerical Examples

In this section, we investigate the patterns of $p_{(0,0),(n,0)}(t)$ and software availability for various choice of n. When the i^{th} perfect debugging is completed, we assume that $\mu_{i,j}$ and $\theta_{i,j}$ have the following forms. The form of $\mu_{i,j}$ is suggested by Moranda (1979) and it is expressed as $\mu_{i,j} = \mu_{0,j} \cdot k_j^i$, where $\mu_{0,j} > 0$ and $0 < k_j < 1$. For $j = 1, 2$, the k_j^i is a decreasing function of i, which is the number of faults removed previously. This implies that the failure rate is geometrically decreasing as the number of previous fault removals increase. Similarly, we consider the rate $\theta_{i,j}$ with the form of $\theta_{i,j} = \theta_{0,j}[1 + \{1 - \exp(-\sqrt{i})\}l_j]$, where $\theta_{0,j}$, $l_j > 0$. Here, l_j is a learning factor which affects the probability of perfect debugging. Thus, $\theta_{i,j}$ is geometrically increasing and the necessary time to remove the fault of type j decreases as the number of previous fault removals gets larger and so the experience is accumulated.

Although the model parameters associated with $\mu_{i,j}$ and $\theta_{i,j}$ can be estimated using the actual data sets on the software failures and the debugging times, in this paper we use the simulated data set generated from the actual data cited by Goel and Okumoto (1979) instead and we employ the maximum likelihood estimation method, which has been applied by Tokuno and Yamada (2006). We can derive the likelihood function as follows.

$$L = \prod_{l=1}^{m} \left[\left(\mu_{0,1} k_1^l + \mu_{0,2} k_2^l \right) \cdot \exp\left\{ - \left(\mu_{0,1} k_1^l + \mu_{0,2} k_2^l \right) x_l \right\} \right]. \tag{3.1}$$

By taking the partial derivative of the above likelihood function with respect to the parameters k_1 and k_2 and letting them equal to 0, we have $\partial L/\partial k_1 = \partial L/\partial k_2 = 0$.

As to the estimation of the parameters associated with $\theta_{i,j}$, we can apply the procedure similar to that in the preceding discussions. The results are: $\hat{k}_1 = 0.8521$, $\hat{k}_2 = 0.8131$, $\hat{l}_1 = 0.1512$ and $\hat{l}_2 = 0.0323$.

For this calculation, we take $N = 10$, $\mu_{0,1} = 0.15$, $\mu_{0,2} = 0.10$, $\theta_{0,1} = 0.9$, $\theta_{0,2} = 0.5$. As for the distribution of the processing time of a test, we consider a gamma distribution, Gamma(2, α).

The working probability of the system for this case is drawn in Figure 4.1. The figure indicates that the working probability of the software system during the initial testing period which needs to

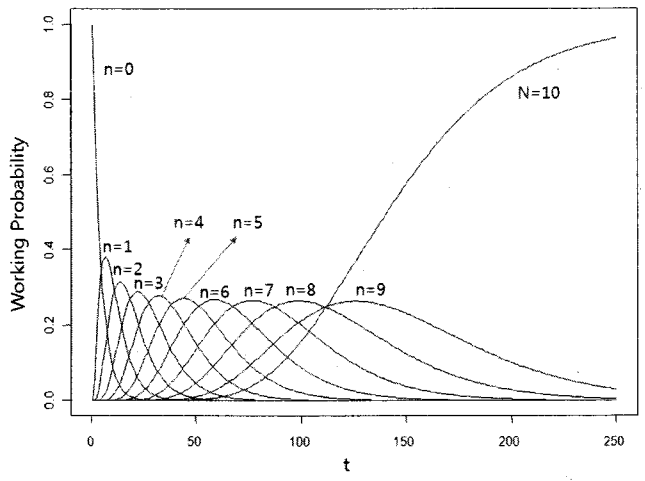

Figure 4.1. Working probability of software $p_{(0,0),(n,0)}(t)$, for various n's with $\mu_{0,1} = 0.15$, $\mu_{0,2} = 0.10$, $\theta_{0,1} = 0.9$, $\theta_{0,2} = 0.5$.

remove the greater number of faults is smaller. On the other hand, the probability that the software system is in working state in the later testing period is smaller probably because there remain the smaller number of faults to be removed.

Figure 4.2 shows the behaviors of availability, $A(t) = \sum_{n=0}^{N} p_{(0,0),(n,0)}(t)$ and the ratio of the number of tasks completed to the one arriving at a module of the software system per unit time at time t. Because the latent faults are detected more frequently during the initial testing period, many software failures may occur early and thus, the availability, $A(t)$ and the ratio, $h(t)$ tends to decrease fast in the initial testing period. However, once a number of initial perfect debugging are performed, the availability of software system and the ratio of the number of tasks completed to the one arriving at a module of the software system per unit time becomes stable and shows slower increase later.

For the purpose of explanation, we consider the case $N = 10$, $\alpha_1 = 10$, $\alpha_2 = 5$, $\alpha_3 = 1$ and α_i is the scale parameter of gamma distribution. In this case, the software system is consisted of three modules and the distribution of the processing time of a task for the k^{th} module($k = 1, 2, 3$) follows a gamma distribution with different parameters. Also we set $l_1 = l_2 = l_3 = l$ and $k_1 = k_2 = k_3 = k$ in the perfect debugging model.

Figure 4.3 shows the task completion probabilities that the process of a task set for the integrated software system consisting of 3 modules and a task for module arrived up to the time t is completed. This figures indicate that the probability that a task processing can be complete tends to decrease fast in the initial testing period and show increase later because the latent faults are detected more frequently during the initial testing period, many software failures may occur early.

4. Concluding Remarks

In this paper, we combine an infinite server queueing model for multi-task processing software system with Lee *et al.* (2001)'s perfect debugging model for modelling the software task processing. Also, we apply this model for module and integration testing in the testing process. We compute

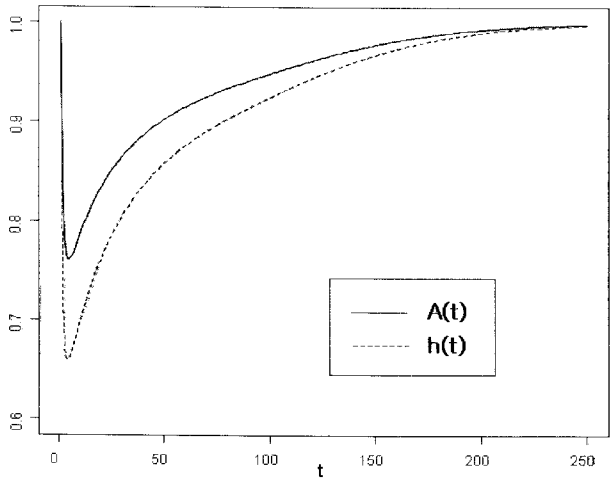

Figure 4.2. $A(t)$, availability and $h(t)$, the ratio of the number of tests completed to the one arriving at a module of the software system per unit time at time t.

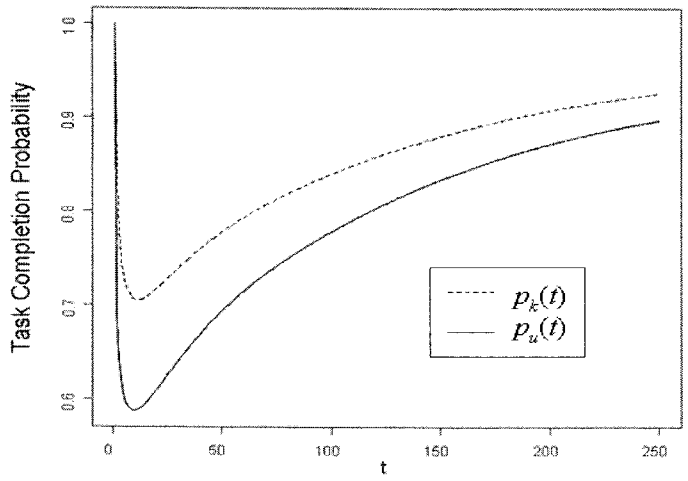

Figure 4.3. Task completion probability for module, $p_k(t)$ and for software, $p_u(t)$.

several measure, such as the expected number of tasks whose processes can be completed and the task completion probability are investigated under the proposed model. In software project management, this model can provide more useful and comprehensive information for software development managers.

References

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

Laprie, J. C. (1984). Dependability evaluation of software systems in operation, *IEEE Transactions on software engineering*, **10**, 701–714.

Lee, C. H., Nam, K. H. and Park, D. H. (2001). Optimal software release policy based on Markovian perfect debugging model, *Communications in Statistics: Theory and Methods*, **30**, 2329–2342.

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

Ross, S. M. (1972). *Introduction to Probability Models*, Academic press, Harcourt.

Shooman, M. L. and Trivedi, A. K. (1976). A many-state Markov model for computer software performance parameters, *IEEE Transactions on Reliability*, **25**, 66–68.

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