한국통계학회논문집
2012, 19권, 4호, 571–577

DOI: http://dx.doi.org/10.5351/CKSS.2012.19.4.571

Software Taskset Processing Evaluation Based on a Mixed Debugging Process

U-Jung Kim[a], Chong Hyung Lee[1,b]

[a]College of General Education, Hallym University
[b]Department of Hospital Management, Konyang University

Abstract

Modules that consist of software are respectively coded in the early development phase and the modules are unified as a software. After unification, the software is repeatedly tested with a given taskset (the set of module tasks that are tested simultaneously) until a required performance level is satisfied. In this paper, we expand the one-module software debugging model of Jang and Lee (2011) to a multi-module debugging model and derive the taskset completion probability and the mean of the completed tasksets under the assumption that the processing times of module tasks given in a taskset are mutually dependent.

Keywords: Multi-module software, mixed debugging, taskset completion probability.

1. 서론

소프트웨어의 가동기간과 수리기간을 고려한 성능평가를 위하여, 특정시점에서 가동하는 확률인 가용성(Availability)을 측정하는 모형이 지금까지 꾸준하게 개발되어 오고 있다. Shooman과 Trivedi (1976)는 마코프 과정을 사용하여 소프트웨어 가용성 평가모형을 제안하였으며, 고장수리 기간의 초기시점에서 소프트웨어가 가동하면서 점차 고장율이 감소하는 경우를 고려하여 가용성이 감소 후 증가하는 결과를 제시하였다. Laprie와 Kanoun (1992)은 소프트웨어와 하드웨어가 결합된 시스템의 신뢰성(Reliability) 및 가용성을 평가하였다. Lee 등 (2001)은 고장의 원인이 되는 결함을 수리가 어려운 경우와 쉬운 경우로 세분화하고, 완전수리(perfect debugging)를 적용하여 가용성 모형을 제시하였다. 또한 Tokuno와 Yamada (2004, 2006)는 소프트웨어에 주어지는 업무들을 동시에 다중처리(multi-tasking)하는 소프트웨어 성능 평가모형을 제시하였다. 또한 Jang과 Lee (2011)는 고장의 원인이 되는 결함의 유형이 수리가 쉬운 경우에는 완전수리를 수행하며, 수리가 어려운 경우에는 불완전수리를 반영한 혼합수리(mixed debugging)를 바탕으로 소프트웨어 가용성 모형을 제안하였다.

최근의 연구들은 일반적인 소프트웨어의 개발상황을 반영하기 위하여, 모듈들을 개별적으로 개발하는 단계와 각 모듈들을 하나의 소프트웨어로 통합하고 개발하는 단계로 세분화 하고 있다. Kim과 Lee (2010)는 고장이 불완전 수리되는 상황하에서 다중모듈 소프트웨어의 업무처리 능력을 평가하는 모형을 제시하였으나, 모듈간 업무처리시간은 서로 독립적인 경우를 반영하였다. Lee 등 (2011)은 Lee 등 (2001)의 완전수리 모형을 기반으로, 각 모듈과 소프트웨어에 주어진 업무들을 완전히 처리하는 확률을 유도하였다. 이를 위하여 다중모듈 소프트웨어의 성능평가를 위하여 사용되는 업무는 각 모듈에 주어진 업무들의 집합으로써, 업무집합(taskset)으로 정의하였다. 또한 업무집합을 구성하는 업무들을 처리하는 시간들은 상호종속(mutually dependent)임을 반영하였다.

[1] Corresponding author: Associate Professor, Department of Hospital Management, Konyang University, Daejeon 320-711, Korea. E-mail: chlee@konyang.ac.kr

본 논문에서는 Jang과 Lee (2011)의 혼합수리 모형을 다중모듈로 구성된 소프트웨어에 적용될 수 있도록 확장한다. 또한 Lee 등 (2011)의 연구와 같이 모듈들에 주어진 업무들을 처리하는 시간들은 상호종속적인 경우를 고려하나, 업무집합의 완전처리확률을 유도함에 있어서는 소프트웨어 초기 고장율을 모듈 개발기간 중 얻어진 각 모듈의 최종고장율의 함수로 새롭게 반영하여 적용하고자 한다. 2절에서는 Jang과 Lee (2011)의 소프트웨어 혼합수리모형을 간략히 소개하며, 3절에서는 소프트웨어에 포함된 각 모듈들에 주어진 업무집합 완전처리 확률 및 완전처리된 업무집합의 평균수에 관한 측도를 제안한다. 4절에서는 수치 예제들이 주어진다.

2. 소프트웨어 혼합수리 모형

소프트웨어의 상태는 시간 t까지 제거된 결함의 전체 수와 시점 t에서 가동 또는 고장상태를 통하여 나타낸다고 하자. 여기서 $x_1(t)$는 구간 $(0,t]$동안에 제거된 결함의 전체 수이며, 시점 t에서 소프트웨어의 상태는

$$x_2(t) = \begin{cases} 0, & \text{가동상태,} \\ 1, & \text{수리가 쉬운 결함에 의한 고장상태 (고장유형 1),} \\ 2, & \text{수리가 어려운 결함에 의한 고장상태 (고장유형 2)} \end{cases}$$

로 정의한다. 따라서 소프트웨어의 상태는 $X(t) = (x_1(t), x_2(t))$으로 표현될 수 있다. 여기서 고장유형 1은 완전수리를, 고장유형 2는 완전수리 확률을 p로 가정한다. 또한 t시간동안 i개의 결함이 제거되고 시스템이 작동 상태일 때, 고장유형 1 또는 2의 고장 발생까지의 시간은 각각 평균 $1/\mu_{i,1}$과 $1/\mu_{i,2}$을 갖는 지수분포를 따르며, 서로 독립임을 가정한다. 또한 고장유형 1 또는 2에 대한 수리과정에서 걸리는 시간은 각각 평균 $1/\theta_{i,1}$과 $1/\theta_{i,2}$을 갖는 지수분포를 따르며, 서로 독립임을 가정한다. $F_{\alpha,\beta}(t)$는 시간 t 이후에 $X(t)$가 상태 α에서 상태 β로 전이되는 전이확률(transition probability)이라고 하자. 그러면 다음과 같은 전이확률들이

$$\begin{aligned} F_{(i,0),(i,j)}(t) &= \frac{\mu_{i,j}}{\mu_{i,1}+\mu_{i,2}}(1-\exp(-(\mu_{i,1}+\mu_{i,2})t)), & F_{(i,1),(i+1,0)}(t) &= 1-\exp(-\theta_{i,1}t) \\ F_{(i,2),(i,0)}(t) &= (1-p)(1-\exp(-\theta_{i,2}t)), & F_{(i,2),(i+1,0)}(t) &= p(1-\exp(-\theta_{i,2}t)) \end{aligned} \tag{2.1}$$

과 같이 얻어지며, 여기서 $j = 1, 2$이다.

확률변수 $T_{(i,0)(n,0)}$는 i개의 결함 제거되고 가동중인 소프트웨어가 처음으로 제거된 결함의 수가 n개가 되기까지의 시간을 나타내며, N은 소프트웨어 시스템에 내재하는 초기 결함의 갯수라고 하자. $i = 0$일 때 $T_{(0,0)(n,0)}$의 분포함수는 식 (2.1)로 부터

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

모든 $t \geq 0$에 대해 $G_{(0,0),(0,0)}(t) = 1$로 정의하며, $N_{n,i,1}, N_{n,i,2}, N_{n,i,3}$와 x_i, y_i는 상수이다 (Jang과 Lee, 2011).

또한 $p_{(i,0),(n,0)}(t)$는 i개의 결함을 제거하고 초기시점에서 소프트웨어 시스템이 가동상태에 있을때, t시간동안 n개의 결함을 제거하고 시점 t에서 시스템이 가동상태에 있을 확률로써, 가동확률(working

probability)이라고 하자. $i = 0$이고 $n = 0, 1, \ldots, N$일 때, 시점 t에서의 소프트웨어 가동확률은

$$p_{(0,0),(n,0)}(t) = G_{(0,0),(n,0)}(t) - \sum_{i=0}^{n-1} M_{n,i,1}(1 - \exp(-\theta_{i,1}t)) + \sum_{i=0}^{n} [M_{n,i,2}(1 \exp(-x_i t)) + M_{n,i,3}(1 - \exp(-y_i t))] \quad (2.2)$$

로 유도된다. 여기서 $M_{n,i,1}, M_{n,i,2}, M_{n,i,3}$는 상수이다 (Jang과 Lee, 2011). 따라서 시점 t에서의 소프트웨어 가용성은 다음과 같이 얻을 수 있다 (Barlow와 Proschan, 1981).

$$A(t) = \sum_{n=0}^{N} p_{(0,0),(n,0)}(t).$$

3. 소프트웨어 업무처리 평가모형

다중모듈 소프트웨어에 주어지는 업무집합의 완전처리확률을 유도하기 위하여, 단일 모듈로 구성된 소프트웨어를 기반으로 하는 Jang과 Lee (2011)의 혼합수리 모형을 다중모듈의 개념이 반영될 수 있도록 확장하고자 한다. 또한 다중모듈 소프트웨어의 업무집합 처리 모형에 대한 가정은 다음과 같다.

3.1. 가정

1) 소프트웨어의 업무처리 능력을 평가하기 위하여, 소프트웨어를 구성하는 모든 모듈에 업무를 부여하고, 동시에 업무처리를 실시한다. 모든 모듈들이 요구되는 업무를 정확히 처리한 경우, 해당 업무들의 집합인 업무집합은 완전처리된 것으로 판정한다.

2) 업무집합을 구성하는 업무들의 처리시간은 상호종속적이다.

3) 시간 t까지 소프트웨어에 주어지는 업무집합의 전체 수는 도착률(arriving rate)이 γ인 포아송과정(Poisson process)을 따른다.

3.2. 혼합수리를 반영한 소프트웨어 업무처리 평가모형

소프트웨어를 구성하는 모듈들은 모듈 개발기간동안 여러 개발팀에서 나누어 독립적으로 개발되며, 각 모듈들이 요구되는 성능을 달성하면 하나의 소프트웨어로 통합하게 된다. 모듈개발 기간이 종료되었을 때 k번째 모듈의 고장유형 j에 관한 최종 고장률은 $\mu_{k,j,m}$이며 소프트웨어는 r개의 모듈로 구성되었다고 하자. 모듈개발 기간중 모듈들이 서로 독립적으로 개발되었으므로, 소프트웨어 개발시작 시점에서의 소프트웨어의 초기고장율은 $\sum_{k=1}^{r} \mu_{k,j,m}$인 지수분포를 따른다고 가정한다. 또한 소프트웨어의 첫 번째 고장이 발생했을 때, 소프트웨어 고장률은 제거된 결함수와 결함을 제거할 때마다 배우는 학습량에 따라 감소하는 형태인 Moranda (1979)의 모형을 따른다고 가정하였다. 이로부터 소프트웨어에서 i개의 결함을 제거한 후의 고장유형 j에 대한 고장률은

$$\mu_{i,j} = \left(\sum_{k=1}^{r} \mu_{k,j,m} \right) s_j^i \quad (3.1)$$

이며, $0 < s_j < 1$이다. 따라서 식 (3.1)을 식 (2.1)에 적용하면 식 (2.2)와 같은 형태의 다중모듈 소프트웨어의 가동확률 함수를 유도할 수 있다.

시간 t까지 도착하는 전체 업무집합 중에서 처리가 완료되는 업무집합의 전체 수는 $\{Z(t), t \geq 0\}$라고 하자. 그러면 $Z(t)$의 분포함수는

$$\Pr\{Z(t) = m\} = \sum_{l=0}^{\infty} \Pr\{Z(t) = m | N(t) = l\} \cdot \Pr\{N(t) = l\} \tag{3.2}$$

이며 l은 시간 t까지 소프트웨어에 도착한 업무집합의 전체 수를 나타낸다. 또한 소프트웨어에 시간 t까지 주어진 전체 l개의 업무집합 중에서 m개의 업무집합이 완전처리될 확률은

$$\Pr\{Z(t) = m | N(t) = l\} = \binom{l}{m} [p(t)]^m [1 - p(t)]^{l-m}$$

여기서 $m = 0, 1, 2, \ldots, l$이며 $p(t)$는 시간 t동안 소프트웨어로부터 임의의 업무집합이 완전처리 되는 확률이다.

Y는 소프트웨어에 주어진 업무집합의 처리시간을 나타내는 확률변수이며, 소프트웨어는 r개의 모듈로 구성되었다고 하자. 업무집합은 모든 모듈에 부여되는 업무들로 이루어져 있으므로, 업무집합의 처리시간은 각각의 모듈에 부여된 업무들을 처리하는 데에 필요한 시간들 중에서 가장 오래 소요되는 시간이 된다. 따라서 Y_k를 k번째 모듈에 부여되는 업무의 처리시간이라고 하면 $Y = \max(Y_1, \ldots, Y_r)$라고 할 수 있다. 또한 X_n은 n개의 결함이 제거된 소프트웨어에서 다음 $(n+1)$번째 결함에 의한 고장이 발생할 때까지의 시간간격을 나타내며, $X(t)$는 시간 t에서 다중모듈 소프트웨어 시스템의 상태를 나타낸다. 그러면, n개의 결함이 제거되고 소프트웨어가 작동상태에 있을때 임의의 업무집합에 대한 처리가 완료될 확률, β_n은

$$\begin{aligned} \beta_n &= \Pr\{Y < X_n | X(t) = (n, 0)\} \\ &= \int_0^{\infty} \exp\left(-\left(\sum_{j=1}^{2} \sum_{k=1}^{r} \mu_{k,j,m} s_j^n \right) y \right) f_Y(y) dy \end{aligned} \tag{3.3}$$

이며, $f_Y(t)$는 Y의 확률밀도함수이다. 모듈들의 업무처리 시간의 상호종속성의 반영을 위하여 식 (3.4)를 활용한다 (Mood 등, 1974).

$$f_{Y_1,\ldots,Y_r}(t_1, \ldots, t_r) = \prod_{i=1}^{r} f_{Y_i}(t_i) \left[1 + \alpha \prod_{i=1}^{r} (2F_{Y_i}(t_i) - 1) \right], \quad -1 \leq \alpha \leq 1. \tag{3.4}$$

이로부터 Y에 관한 확률밀도함수 및 분포함수는

$$\begin{aligned} f_Y(t) &= \sum_{i=1}^{r} \left[f_{Y_i}(t) \prod_{\substack{j=1 \\ j \neq i}}^{r} F_{Y_j}(t) \right] + \alpha \sum_{i=1}^{r} \left[f_{Y_i}(t) (2F_{Y_i}(t) - 1) \prod_{\substack{j=1 \\ j \neq i}}^{r} \left[F_{Y_j}(t) \left(F_{Y_j}(t) - 1 \right) \right] \right] \\ F_Y(t) &= \prod_{i=1}^{r} F_{Y_i}(t) + \alpha \prod_{i=1}^{r} [F_{Y_i}(t) \cdot (F_{Y_i}(t) - 1)] \end{aligned} \tag{3.5}$$

단 $F_Y(t) = \int_0^t f_Y(y) dy$이다. 따라서 식 (3.5)로 부터 $\alpha = 0$인 경우는 각각의 모듈에 부여되는 업무의 처리시간은 서로 독립인 경우를 반영한다. 또한 식 (3.3)과 (3.5)로 부터 다중모듈 소프트웨어가 시간 t동

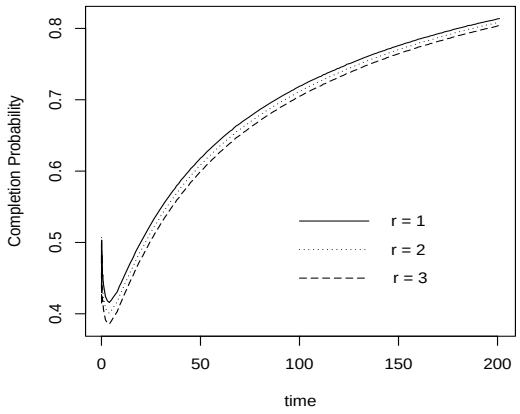

Figure 1: *Software completion probability for various α's($r = 2, p = 0.9$)*

안 업무집합을 완전처리를 수행할 확률, $p(t)$는

$$p(t) = \int_0^t \sum_{n=0}^{N} \Pr\{X(x) = (n,0)\} \cdot \Pr\{Y < X_n | X(x) = (n,0)\} \cdot \frac{1}{t} dx$$
$$= \frac{1}{t} \sum_{n=0}^{N} \left[\beta_n \cdot \int_0^t p_{(0,0),(n,0)}(x) dx \right], \tag{3.6}$$

여기서 $p_{(0,0),(n,0)}(t)$는 다중모듈 소프트웨어에 대한 가동확률을 나타내며, N는 다중모듈 소프트웨어에 내재되어 있는 결함의 전체 수를 나타낸다. 식 (3.2)와 (3.6)으로 부터 $Z(t)$의 분포는

$$\Pr\{Z(t) = m\} = \sum_{l=0}^{\infty} \binom{l}{m} [p(t)]^m [1 - p(t)]^{l-m} \cdot e^{-\gamma t} \frac{(\gamma t)^l}{l!}$$
$$= e^{-\gamma \cdot t \cdot p(t)} \frac{[\gamma \cdot t \cdot p(t)]^m}{m!}$$

으로 얻어지며, 여기서 γ는 업무집합의 도착율이다. 따라서 $Z(t)$는 평균값 함수(mean value function)가 $\gamma \cdot t \cdot p(t)$인 비동질적 포아송과정(Non-homogeneous Poisson Process; NHPP)임을 확인할 수 있다.

4. 수치예제

3절에서 유도된 업무집합 완전처리확률과 완전처리된 업무집합의 평균수에 관한 수치예제들을 제시한다. 이를 위하여 고장유형 j의 i번째 고장이 발생할 때 해당 고장의 원인이 되는 결함을 제거하는 시간은 평균 $1/\theta_{i,j}$인 지수분포를 따르고, $\theta_{i,j}$는 $\theta_{i,j} = \theta_0[1 + (1 - \exp(-\sqrt{i}))l_j]$의 형태를 가정한다. 여기서 l_j는 결함제거에 관한 학습효과를 나타낸다. 따라서 제거된 결함의 수가 커짐에 따라 $\theta_{i,j}$는 증가하며 수리 경험이 축척되어 고장유형 j의 고장을 제거하는 시간이 줄어드는 것을 의미한다. 또한 고장유형 2와 비교할 때, 고장유형 1의 고장이 발생까지의 평균시간은 보다 오래 소요되며 반면에 수리시간은 보다 짧은 상황을 반영할 수 있도록 $N = 10$, $\mu_{1,1,m} = 0.05$, $\mu_{2,1,m} = 0.02$, $\mu_{1,2,m} = 0.1$, $\mu_{2,2,m} = 0.08$, $\theta_{0,1} = 0.9$, $\theta_{0,2} = 0.75$, $l_1 = 0.1$, $l_2 = 0.05$, $s_1 = 0.7$, 그리고 $s_2 = 0.7$라고 하자. 또한 주어진 업무집합에 대하여 k번째 모듈에 주어진 업무를 처리하는데 필요한 시간, Y_k의 분포는 Gamma(2, 0.5)로 가정한다.

Figure 1은 $r = 2, p = 0.9$이고 $\alpha = 0.9, 0, -0.9$일 때 완전처리확률 $p(t)$를 나타내고 있다. $\alpha = 0.9$인 경우의 $p(t)$는 $\alpha = 0$인 경우의 완전처리확률 보다 높게 나타나고 있으므로, 모듈간 종속성 모수가 양수

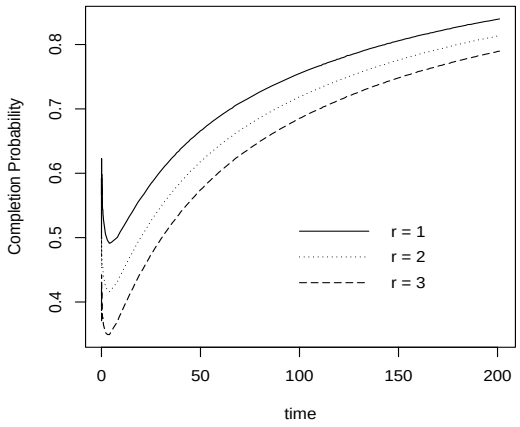

Figure 2: *Software completion probability for various r's($p = 0.9, \alpha = 0.9$)*

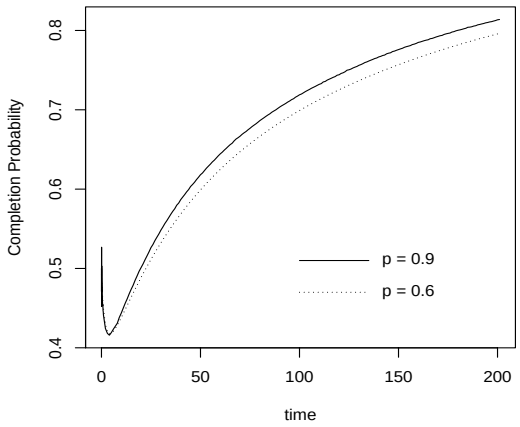

Figure 3: *Software completion probability for various p's($r = 2, \alpha = 0.9$)*

일 때, 업무의 완전처리 능력이 높아진다고 할 수 있다. 또한 소프트웨어 고장은 주로 초기에 발생하므로 초기의 $p(t)$는 감소하다가 이후 증가하는 것으로 나타나고 있다.

Figure 2는 $p = 0.9$이고 $\alpha = 0.9$일 때 소프트웨어를 구성하는 모듈 수의 변화에 따른 $p(t)$의 변화를 보이고 있다. 업무집합에 주어진 업무들의 처리 시간들이 서로 종 속적이므로 단일모듈($r = 1$)로 구성된 소프트웨어의 경우의 $p(t)$가 가장 높게 나타나고 있으며, 3개 모듈($r = 3$)로 구성된 경우의 $p(t)$가 가장 낮게 나타나고 있다. 또한 Figure 3은 $r = 2$이고 $\alpha = 0.9$일 때 고장유형 2의 완전수리확률 p의 값이 증가함에 따라 $p(t)$가 증가함을 보이고 있으며, Table 1은 $p = 0.9$이고 $\alpha = 0.9$일때, 모듈수가 많은 복잡한 소프트웨어의 경우일수록 완전처리된 업무집합의 수는 감소됨을 나타내고 있다.

5. 결론

본 논문에서는 개별적으로 개발된 모듈들이 요구수준의 성능을 달성했을 때 하나의 소프트웨어로 통합되는 개발현장의 상황을 반영하여, 소프트웨어의 성능 평가모형을 제안하였다. 특히 각 모듈들에 해당 업무를 부여하고 이를 동시에 처리하는 능력을 평가하였으며, 모듈들의 업무처리 시간들은 상호

Table 1: Mean of completed tasksets for various r's($p = 0.9, \alpha = 0.9$)

r	time							
	1	5	10	30	50	100	150	200
1	0.271	1.264	2.488	8.882	16.430	37.539	60.275	83.731
2	0.227	1.060	2.135	8.027	15.216	35.697	58.007	81.110
3	0.187	0.881	1.822	7.252	14.099	33.997	55.907	78.678

종속적인 경우와 독립적인 경우로 세분화 하여 차이점을 비교하였다. 이로부터 다중모듈 소프트웨어의 가용성 함수 및 특정 기간동안 업무의 완전처리 확률함수를 제안하였다.

References

Barlow, R. E. and Proschan, F. (1981). *Statistical Theory of Reliability and Life Testing: Reliability Models*, Silver Spring, Maryland.

Jang, K. B. and Lee, C. H. (2011). A software performance evaluation model with mixed Debugging process, *Communications of the Korean Statistical Society*, **18**, 741–750.

Kim, U. J. and Lee, C. H. (2010). Evaluation of software task processing based on Markovian imperfect debugging model and its release policy, *Communications of the Korean Statistical Society*, **17**, 891–898.

Laprie, J. C. and Kanoun, K. (1992). X-ware reliability and availability modeling, *IEEE Transactions on Software Engineering*, **18**, 130–147.

Lee, C. H., Kim, Y.-H. and Park, D. H. (2011). Evaluation of multi-tasking software system performance with consideration of module dependency, *Journal of Software Maintenance and Evolution: Research and Practices*, **23**, 361–374.

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.

Mood, A. M., Graybill, F. A. and Boes, D. C. (1974). *Introduction to the Theory of Statistics*, McGraw-Hill.

Moranda, P. B. (1979). Event-altered rate models for general reliability analysis, *IEEE Transactions on Reliability*, **R-28(5)**, 376–381.

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

Tokuno, K. and Yamada, S. (2004). Performance evaluation for multi-task processing system with software availability model, *Proceedings of the 2004 Asian International Workshop(AIWARM 2004)*, 539–546.

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

2012년 5월 9일 접수; 2012년 6월 15일 수정; 2012년 6월 21일 채택