한국통계학회논문집
2010, 17권, 6호, 891–898

DOI: 10.5351/CKSS.2010.17.6.891

마코프 불완전 수리모형에 따른 소프트웨어 업무처리 능력평가 및 출하정책에 관한 연구

김유정[a], 이종형[1,b]

[a]한림대학교 기초교육대학, [b]건양대학교 병원관리학과

요 약

소프트웨어는 독립적으로 개발된 모듈들을 통합하는 개발과정을 거치며, 통합된 소프트웨어는 가용도, 소프트웨어에 내재된 결함의 제거 수 및 주어진 업무(task)들의 처리능력에 의하여 성능이 평가된다. 본 연구에서는 Lee와 Park (2003)의 마코프 불완전 수리모형을 기반으로 모듈들로 구성된 소프트웨어의 업무처리 능력을 평가할 수 있는 업무의 완전처리확률(completion probability)을 유도하며, 이와 함께 개발된 소프트웨어가 출하되는 최적의 시점을 결정하는 방안을 제시하고자 한다.

주요용어: 소프트웨어, 모듈, 불완전수리, 완전처리확률, 출하정책.

1. 서론

개발단계에 있는 소프트웨어(software)가 개발자 및 클라이언트가 요구하는 성능수준과 가격수준을 만족하는지를 평가할 수 있는 모형들을 개발하기 위해 많은 연구들이 수행되어 오고 있다. 이중 소프트웨어의 성능평가를 위하여, 특정시간동안 소프트웨어가 고장 없이 가동하는 확률인 신뢰도(reliability) 측정 모형뿐만 아니라 특정 시점에서 가동하는 확률인 가용도(availability) 평가모형이 개발되어 오고 있다.

Goel과 Okumoto (1979)는 비동질적 포아송 과정을 활용하여, 특정시간동안 소프트웨어가 고장 없이 가동하는 확률인 소프트웨어 신뢰도(software reliability)를 측정하는 모형을 제안하였다. 그러나 이러한 신뢰성 모형의 경우 소프트웨어 고장 시 고장이 즉각적으로 제거되어, 고장의 원인이 되는 결함을 제거하는 수리시간(debugging time)은 반영되지 않는 단점이 있다. 따라서 수리시간을 반영하여 성능평가 모형을 제안하는 것이 보다 현실적이며 이를 위하여 마코프 확률과정(Markov process)을 통한 모형 개발이 널리 이용되고 있다.

Shooman와 Trivedi (1976)는 마코프 확률과정을 이용하여 고장 시 완전수리가 가능한 소프트웨어 가용도 모형을 제안하였다. 결함수리 시 완전수리를 고려하여 가용성 평가모형을 개발한 연구로는 Tokuno와 Yamada (1998) 및 Lee 등 (2001) 등이 있다. 그러나 이러한 완전수리를 불완전 수리의 경우로 확장한 연구로는 Goel과 Okumoto (1979), Goel과 Soenjoto (1981), Tokuno와 Yamada (1998, 2004, 2006) 및 Lee와 Park (2003) 등이 있다. 이중 Goel과 Soenjoto (1981)는 하드웨어와 소프트웨어 결합시스템의 성능평가를 수행가능한 모형을 제안하였으며, Tokuno와 Yamada (1998)의 경우는 소프트웨어의 결함이 발견되었을 때 p의 확률로 수리작업을 통하여 복구되며, $1-p$의 확률로는 수리작업 없이 복구되는 모형을 제안하였다. 그러나 Lee와 Park (2003)의 경우 소프트웨어에 내재되어 있는 결함의 심각성에 따라 결함의 유형을 쉽게 발견되는 결함과 어렵게 발견되는 결함으로 분류하며, 수리 후 불완전

[1] 교신저자: (320-711) 충청남도 논산시 대학로길 119, 건양대학교 병원관리학과, 부교수.
E-mail: chlee@konyang.ac.kr

수리를 반영한 모형을 개발하였다. 또한 소프트웨어의 고장율은 결함의 디버깅 수가 많아질수록 소프트웨어의 고장율은 점점 감소하고, 고장수리에 소요되는 시간은 점점 짧아지는 학습요인(learning factor)을 반영한 평가모형을 제안하였다. 또한 Tokuno와 Yamada (2004, 2006)는 소프트웨어에 주어지는 업무를 동시에 다중처리(multi-tasking)하는 소프트웨어 평가모형을 제안하였다.

본 논문에서는 소프트웨어 개발현장에서와 동일하게, 소프트웨어는 여러 개의 모듈로 구성되어 있으며, 소프트웨어의 첫 번째 고장율은 모듈의 초기고장율의 함수로 나타나도록 하여, 소프트웨어의 업무처리 능력을 평가하는 모형을 제안한다. 이에 있어서 Lee와 Park (2003)의 마코프 불완전 수리모형을 기반으로 하며, 이와 함께 소프트웨어의 완전수리의 기대수, 주어진 소프트웨어 업무의 완전처리확률 및 소프트웨어의 가용도를 반영한 최적의 출하시점을 결정하는 출하정책도 함께 제안된다.

2장에서는 기호와 가정, 3장에서는 Lee와 Park (2003)의 소프트웨어 불완전 수리모형이 주어지며, 4장에서는 소프트웨어의 성능평가 측도로써 업무의 완전처리확률, 완전처리된 업무의 기대수 및 최적의 출하정책이 제안된다. 또한 5장에서는 수치예제가 주어진다.

2. 기호와 가정

2.1. 기호

N	소프트웨어에 내재된 초기 결함수
$X(t) = (i, j)$	소프트웨어의 상태
i	제거된 결함의 수
$j = 0$	소프트웨어 가동
$j = 1$	쉽게 발견되는 결함에 의한 소프트웨어 고장(고장유형 1)
$j = 2$	어렵게 발견되는 결함에 의한 소프트웨어 고장(고장유형 2)
$\mu_{i,1}(\mu_{i,2})$	i 개의 결함 제거 후 고장유형 1(2)에 의해 발생하는 고장율
$\theta_{i,j}$	i 개의 결함 제거 후 고장유형 j가 발생할 때 수리율
k	소프트웨어를 구성하는 k번째 모듈
$\mu'_{k,j}$	k번째 모듈의 고장유형 j에 관한 최종 고장율
Y_u	소프트웨어에 주어진 한 개 업무의 처리시간
$N(t)$	t 시간동안 소프트웨어에 주어진 업무 수
$Z(t)$	t 시간동안 완전처리된 소프트웨어 업무 수
$p(t)$	t 시간동안 한 개의 업무를 완전처리 하는 확률
a_0	출하시점에서 요구되는 소프트웨어의 가용도
β_0	출하시점에서 요구되는 소프트웨어 업무처리 확률

2.2. 가정

1) 소프트웨어 고장은 소프트웨어에 내재된 결함이 발견될때 발생하며, 즉각적으로 수리가 시행된다.

2) 고장의 원인결함을 1개 제거하는 완전수리는 p의 확률로 발생하며, 원인결함의 제거 없이 소프트웨어를 가동할 수 있도록 하는 불완전수리는 $q(= 1 - p)$의 확률로 발생한다.

3) 소프트웨어 성능 시험단계에서 소프트웨어의 업무는 소프트웨어를 구성하는 모든 모듈들에 주어진 보조업무들로 구성된다.

4) t시간 동안 주어지는 소프트웨어 업무의 수 $\{N(t), t \geq 0\}$는 λ_s를 도착율로 갖는 동질적 포아송과정을

따른다.

5) 소프트웨어에 내재된 각 모듈들의 첫 번째 고장까지의 시간은 서로 독립이다. 또한 소프트웨어 업무의 처리시간은 서로 독립이다.

3. LP 모형

$\{X(t), t \geq 0\}$는 t시간 동안 제거된 결함수와 t시간에서 소프트웨어의 가동 또는 고장상태를 나타낸다. 예를 들어 $X(t) = (5,2)$는 5개의 결함이 제거되고 소프트웨어가 가동하다가 유형 2 결함에 의해 고장난 상태를 나타낸다. 또한 $F_{a,b}(t)$는 상태 a에 있다가 t시간 후 상태 b로 전이되는 확률이라고 할 때, i번째 결함이 제거되고 가동된 시점부터 유형 j의 고장이 발생할 때까지의 가동시간의 분포는 평균 $\mu_{i,j}$를 갖는 지수분포를 가정하며, 고장발생 후 소요되는 수리시간의 분포는 평균 $\theta_{i,j}$를 갖는 지수분포를 가정한다. 이로부터 다음의 전이확률들이 유도된다.

$$F_{(i,0),(i,j)}(t) = \mu_{i,j}\frac{1-\exp\left(-\left(\mu_{i,1}+\mu_{i,2}\right)t\right)}{\mu_{i,1}+\mu_{i,2}},$$
$$F_{(i,j),(i,0)}(t) = q\left(1-\exp\left(-\theta_{i,j}t\right)\right),$$
$$F_{(i,j),(i+1,0)}(t) = p\left(1-\exp\left(-\theta_{i,j}t\right)\right).$$

$T_{(0,0),(n,0)}$는 n개의 결함이 제거되고 소프트웨어가 처음 가동될 때까지의 시간을 나타내는 확률변수이며, $G_{(0,0),(n,0)}(t)$는 $T_{(0,0),(n,0)}$의 분포함수라고 하면

$$\begin{aligned} G_{(i,0),(n,0)}(t) &\equiv \Pr\{T_{(i,0),(n,0)} \leq t\} \\ &= \sum_{j=1}^{2} F_{(i,0),(i,j)} * F_{(i,j),(i,0)} * G_{(i,0),(n,0)}(t) + \sum_{j=1}^{2} F_{(i,0),(i,j)} * F_{(i,j),(i+1,0)} * G_{(i+1,0),(n,0)}(t) \end{aligned}$$

으로 정의되고 여기서 *는 스틸제스 컨볼루션(Stieltjes Convloution)이다. 이로부터

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

이 얻어지며, $m = 1,2,3$일 때 계수 $N_{n,i,m}$은 $G_{(0,0),(n,0)}(t)$을 분포함수로 하는 상수이다 (Lee와 Park, 2003). 또한 $Y(t)$를 t시간 동안 수행된 완전수리 횟수라 할 때, 완전수리된 기대수는

$$\mathrm{EPD}_{(0,0)}(t) = E[Y(t)\,|\,X(0) = (0,0)] = \sum_{n=1}^{N} G_{(0,0),(n,0)}(t)$$

으로 얻어진다. $p_{(0,0),(n,0)}(t)$는 0개의 결함이 제거되고 소프트웨어가 가동할 때, n개의 결함이 제거되고 시점 t에서 소프트웨어가 가동할 확률이라고 하자. 그러면

$$p_{(0,0),(n,0)}(t) = G_{(0,0),(n,0)} * p_{(n,0),(n,0)}(t)$$

으로 얻어지며, 이 중 $p_{(n,0),(n,0)}(t)$은

$$p_{(n,0),(n,0)}(t) = \exp(-(\mu_{n,1}+\mu_{n,2})t) + \sum_{j=1}^{2} F_{(n,0),(n,j)} * F_{(n,j),(n,0)} * p_{(n,0),(n,0)}(t)$$

이다. 따라서

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

이 얻어지며, $m = 1, 2, 3$일 때 계수 $M_{n,i,m}$은 $p_{(0,0),(n,0)}(t)$의 계수이다 (Lee와 Park, 2003, 참조). 이로부터 시점 t에서의 소프트웨어의 가용도(availability)는 다음과 같이 주어진다.

$$A(t) = \sum_{n=1}^{N} p_{(0,0),(n,0)}(t).$$

4. 소프트웨어 평가모형

4.1. 소프트웨어 업무처리 능력평가모형

소프트웨어에 주어지는 업무는 소프트웨어를 구성하는 각 모듈에 주어진 업무의 집합으로 이루어지며, 각 모듈에 주어진 업무는 이후 보조업무(subtask)라고 한다. 본 절에서는 소프트웨어를 구성하는 각 모듈들의 고장율이 주어진 경우, 소프트웨어에 주어진 업무들의 완전처리하는 확률 및 완전처리되는 업무의 기대수를 유도하고자 한다.

개별적으로 개발된 k개의 모듈들은 하나의 소프트웨어로 통합되고, 소프트웨어의 업무처리 능력을 평가하는 소프트웨어 테스트 기간을 갖게 된다. 소프트웨어 성능평가에서 소프트웨어에 주어진 업무들이 완전처리 된다는 것은, 각 모듈들에 주어진 보조업무들이 모두 완전처리는 되는 경우를 의미한다. 이때 소프트웨어에서 유형 j의 고장이 첫 번째 발생할 때까지의 시간은 각 모듈에서 유형 j의 고장이 첫 번째 발생하는 시간들 중 최소시간으로 정의될 수 있다. 이 때, 소프트웨어를 구성하는 모듈의 고장율은 각 모듈 테스트 기간 중 마지막 고장이 발생된 경우의 고장율이다. 따라서, 유형이 j인 고장이 처음으로 발생할 때까지의 시간의 분포는 평균 $1/\sum_{k=1}^{r}\mu'_{k,j}$을 갖는 지수분포를 따르게 된다. 여기서 $\mu'_{k,j}$는 k번째 모듈의 유형 j에 관한 최종 고장율이다. 첫 번째 소프트웨어 결함 이후의 소프트웨어 고장율은 소프트웨어로부터 제거된 결함수와 결함제거 할 때마다 배우는 학습량에 따라 감소하는 형태인 Moranda (1979)의 모형을 따르고자 한다. 따라서 소프트웨어로 부터 i개의 결함을 제거후의 유형 j의 고장율은

$$\mu_{i,j} = \left(\sum_{k=1}^{r}\mu'_{k,j}\right)s_j^i$$

를 가지며, $0 < s_j < 1$이다.

Y_u는 소프트웨어에 주어진 한 개 업무의 처리시간이고 $f(y)$는 Y_u의 확률밀도함수라고 하자. 그러면 $Y_u = \max(Y_1, \ldots, Y_r)$로 정의되며, 소프트웨어에서 t시간동안 z개의 업무를 완전처리하는 확률은

$$\Pr\{Z(t) = z\} = \sum_{l=0}^{\infty}\Pr\{Z(t) = z \mid N(t) = l\}\Pr\{N(t) = l\} \tag{4.1}$$

이며, 소프트웨어에 t 시간동안 l개의 업무가 주어질 때, 각 업무간에 독립적으로 처리되는 경우 z개를 완전처리하는 확률은

$$\Pr\{Z(t) = z \mid N(t) = l\} = \binom{l}{z}[p(t)]^z[1-p(t)]^{l-z} \tag{4.2}$$

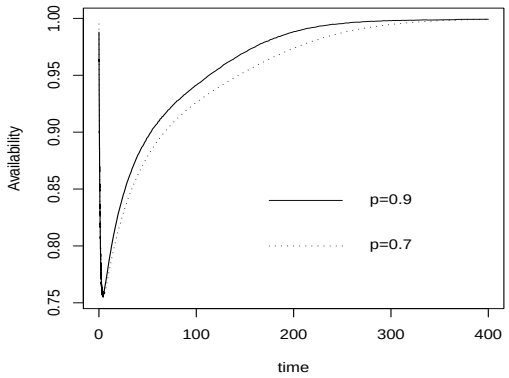

그림 1: *소프트웨어 가용도 $A(t)$ 비교($p = 0.9, 0.7$)*

이다. X_n은 n번째 고장과 $n+1$ 고장간의 경과시간이며, $T_{n,j}$는 n개의 결함이 제거된 후 유형 j의 고장이 발생할때까지의 시간이라고 하자. 이때 n개의 결함이 제거된 후 주어진 한 개의 업무를 완전처리 하는 확률 β_n은

$$\begin{aligned}\beta_n &= \Pr\{Y_u < X_n | X(t) = (n,0)\} \\ &= \Pr\{Y_u < \min(T_{n,1}, T_{n,2}) | X(t) = (n,0)\} \\ &= \int_0^\infty f(y) \exp\left(-\left(\sum_{j=1}^{2}\sum_{k=1}^{r} \mu'_{k,j} s_j^n\right) y\right) dy \end{aligned} \tag{4.3}$$

으로 유도되며, t 시간동안 한 개의 업무를 완전처리하는 확률 $p(t)$는 수식 (4.2)로 부터 다음과 같다.

$$\begin{aligned} p(t) &= \int_0^t \sum_{n=0}^{N} \Pr\{X(y) = (n,0)\} \Pr\{Y_u < X_n \,|\, X(t) = (n,0)\} \frac{dy}{t} \\ &= \frac{1}{t} \sum_{n=0}^{N} \left[\beta_n \int_0^t p_{(0,0),(n,0)}(y) dy\right]. \end{aligned}$$

따라서 $\Pr\{Z(t) = z\}$는 수식 (4.1)과 (4.3)으로 부터

$$\Pr\{Z(t) = z\} = \sum_{l=0}^{\infty} \binom{l}{z} [p(t)]^z [1-p(t)]^{l-z} \frac{(\lambda_s t)^l \exp(-\lambda_s t)}{l!} = \exp(-\lambda_s t p(t)) \frac{[\lambda_s t p(t)]^z}{z!}$$

이며, t 시간동안 완전처리되는 업무의 기대수, $E[Z(t)]$는 $\lambda_s t p(t)$로 얻어진다.

4.2. 소프트웨어 출하모형

소프트웨어의 출하시점을 결정함에 있어서 요구되는 소프트웨어의 완전수리의 기대수, 완전수리된 기대수 만큼 소프트웨어 결함이 제거 후 업무의 완전처리 되는 확률 β_n 및 소프트웨어 가용도의 요구수준을 모두 만족하는 시점으로 결정하고자 한다. 특히 소프트웨어의 완전수리된 수가 증가할수록, $\beta_{\lceil \gamma N \rceil}$도 증가하므로 다음과 같은 2단계를 통하여 최적출하시점 t^*를 결정한다. 여기서 $\lceil \gamma N \rceil$은 γN과 같거나 큰 정수 중 가장 작은 정수를 의미한다.

<단계 1> $\mathrm{EPD}_{(0,0)}(t) \geq \lceil \gamma N \rceil$ 을 만족하는 시점 t_1 결정.

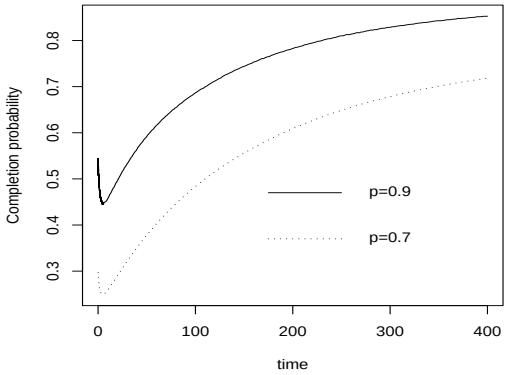

그림 2: *소프트웨어 업무의 완전처리 확률 $p(t)$ 비교 ($p = 0.9, 0.7$)*

<단계 2> $\beta_{\lceil \gamma N \rceil} \geq \beta_0$이면 $A(t) \geq a_0$를 만족하는 시점, t_2 결정 후 $t^* = \max(t_1, t_2)$. 그렇지 않으면 $\lceil \gamma N \rceil = \lceil \gamma N \rceil + 1$로 설정 후 단계 1 실행.

5. 수치예제

소프트웨어는 2개의 모듈로 구성되었으며, 소프트웨어의 유형 j의 i번째 고장이 발생할 때 고장의 원인이 되는 결함을 제거하는 시간은 평균 $1/\theta_{i,j}$인 지수분포를 따르고, $\theta_{i,j}$는 $\theta_{i,j} = \theta_{0,j}[1 + (1 - \exp(-\sqrt{i}))l_j]$의 형태를 가정한다. 여기서 l_j는 학습효과를 나타내며, i가 증가할 때 $\theta_{i,j}$는 증가하나, 증가폭은 점차 줄어드는 형태를 갖게 된다. 수치예제를 위하여 $N = 10$, $\mu'_{1,1} = 0.05$, $\mu'_{1,2} = 0.02$, $\mu'_{2,1} = 0.1$, $\mu'_{2,2} = 0.08$, $\theta_{0,1} = 0.9$, $\theta_{0,2} = 0.75$, $l_1 = 0.1$, $l_2 = 0.05$ 및 $s_1 = s_2 = 0.7$으로 설정하며, Y_u의 확률밀도함수는 $f(y) = 0.3^2 y \exp(-0.3y)$를 사용한다.

그림 1은 $p = 0.9$와 0.7일 때 소프트웨어의 가용도를 나타내고 있다. 우선, 소프트웨어의 완전수리 확률이 0.9일때 0.7인 경우에 비하여 가용도가 높은 것으로 나타나고 있으며, 소프트웨어의 테스팅 기간 초기에는 내재된 결함들이 많이 발견되어 소프트웨어의 고장이 빈번하게 발생하므로 소프트웨어 가용도는 초기 테스팅 기간에 낮아지다가 이후 높아지고 있다. 또한 그림 2는 소프트웨어에 주어진 업무의 완전처리 확률 $p(t)$을 나타내고 있으며, 그림 1과 동일한 경향을 나타내고 있다. 다만, 소프트웨어의 특성상 테스팅 초기시점에서도 주어진 업무를 처리할 수 있으므로 완전처리의 확률은 0이 아닌 값을 갖으며, $p = 0.9$이고 $t = 0.02$의 경우 소프트웨어 업무의 완전처리 확률은 0.542를 갖는다.

최적의 출하시점을 결정하기 위하여 $p = 0.9$, $\gamma = 0.7$, $\beta_0 = 0.9$, $a_0 = 0.95$인 경우를 고려하면, <단계 1>로 부터 $\mathrm{EPD}_{(0,0)}(t) \geq 7$을 만족하는 시점 t_1은 137.5으로 얻어진다. <단계 2>에서 $\beta_7 = 0.876$으로 주어진 β_0보다 작으므로 소프트웨어의 결함을 추가로 1개 되도록 <단계 1>을 다시 실행하여, $\mathrm{EPD}_{(0,0)}(t) \geq 8$을 만족하는 시점 t_1을 199.6으로 결정한다. <단계 2>에서 $\beta_8 = 0.936$으로 주어진 β_0보다 크고 $A(t) \geq 0.95$를 만족하는 시점 t_2는 204.2로 얻게 되어, $t^* = 204.2$가 소프트웨어 최적의 출하시점으로 결정된다.

6. 결론

본 연구에서는 소프트웨어가 여러개의 모듈로 구성되어 개발되는 현장의 상황을 반영하여, 소프트웨어의 업무처리 능력평가모형 및 출하정책을 제안하였다. 이 중 소프트웨어의 업무처리 모형은 Lee와 Park (2003)의 마코프 불완전 수리모형을 기반으로 개발되었으며, 이와 관련된 측도로써 소프

트웨어에 주어진 업무들의 완전처리하는 확률 및 완전처리되는 업무의 기대수를 유도하였다. 또한 소프트웨어의 출하시점은 소프트웨어의 완전수리의 기대수, 주어진 소프트웨어 업무의 완전처리 확률 및 소프트웨어의 가용도를 함께 고려하여 결정되도록 하였다.

제안된 소프트웨어의 업무처리 능력평가모형에서, 소프트웨어를 구성하는 각 모듈에 주어진 보조업무들의 처리시간들이 서로 영향을 미치도록 확장하는 부분 및 모듈개발단계를 모형화하고 각 모듈의 성능을 평가하는 모형을 제안하는 것이 향후의 연구주제로 고려된다.

참고 문헌

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.

Goel, A. L. and Soenjoto, J. (1981). Models for hardware-software operational- performance evaluation, *IEEE Transactions on Reliability*, **30**, 232–239.

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.

Lee, C. H. and Park, D. H. (2003). Markovian imperfect software debugging model and Its performance, *Stochastic Analysis and Applications*, **21**, 849–864.

Moranda, P. B. (1979). Event-altered rate models for general reliability analysis, *IEEE Transactions on Reliability*, **R-28**, 376–81.

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. (1998). Operational software availability measurement with two kind of restoration actions, *Journal of Quality in Maintenance Engineering*, **4**, 273–283.

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.

2010년 7월 접수; 2010년 9월 채택

Evaluation of Software Task Processing Based on Markovian Imperfect Debugging Model and Its Release Policy

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

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

Abstract

In real software development fields, software is unified by several modules that are developed before the software testing period. For the evaluation of software task processing performance, this paper considers the software imperfect debugging model that is proposed by Lee and Park (2003) and presents the measures of a unified software, such as the completion probability of a task which is completed in a time interval and the expected number of the completed tasks. In addition, we suggest a software release policy that satisfies the required level of the expected perfect debugging, completion probability, and availability.

Keywords: Software, module, imperfect debugging, completion probability, release policy.

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