Journal of the KSQC
Vol. 17, No.2, 1989

소프트웨어의 테스트 回數를 이용한 最適放出政策
Optimal Release Policy for a Software System using Number of Software Test

高 賢 祐*
黃 義 徹**

ABSTRACT

Software developers often allocate a significant amount of effort to software testing. But, for most business-related software system it is natural to expect the continued discovery of defects after the software system is released into field. Such defects are usually very expensive to fix and have the potential to cause great damages to the users. It is important to stop testing the software and release it to the users at the correct time.

In this paper, we propose the determination of the optimal number of software test by minimizing a total expected software cost.

A numerical example is used when the criterion is the expected profit. The result indicates that the proposed software release policy based on the number of software test can be a good alternative to the existing policy.

1. 序 論

새로운 소프트웨어 시스템이 개발되었을 때 소프트웨어 시스템에 남아있는 에러는 소프트웨어의 실행시에 고장을 유발시켜 소프트웨어의 品質을 떨어뜨리게 된다. 그래서 소프트웨어의 品質은 매우 중요하므로 고장을 방지하기 위해서는 재설계나 중복설계를 하는 것이 타당한 것

* 한양대학교 산업공학과
** 한양대학교 산업공학과 교수

이나, 여러 제약때문에 어려운 실정이다. 그러므로 재설계나 중복설계 없이 品質을 높이는 두가지 방법으로 첫째는 개발과정에서 더욱 주의를 하는 것과 둘째로는 엄격한 테스트 기준을 세워 이에 맞는 소프트웨어만을 선택하는 방법이 있다.

개발된 소프트웨어는 테스트단계를 거쳐 필드(field)로 방출하게 되는데 이때에 費用과 信賴度는 매우 중요한 요인이 된다. 일반적으로 수명이 중지되는(life-critical) 소프트웨어 시스템 — 항공시스템, 핵발전시스템, 인공위성시스템 — 에서는 소프트웨어의 信賴度는 가장 중요한 요인으로서 작용하며 가능한한 높은 信賴度를 갖게하기 위해 많은 노력과 시간과 비용을 필요로 한다. 그러나 사회에서 유통되는 대부분의 시스템은 商業用(business-related) 소프트웨어로서 총비용이 가장 중요한 요인이다. 즉 얼마간의 에러를 허용하는 것이 모든 에러를 제거하기 위해서 시스템을 테스트하는 것 보다 더욱 경제적이기 때문이다. 그래서 소프트웨어를 어느 정도까지 테스트를 하는 것이 타당한지를 결정하는 放出政策이 필요하게 된다.

소프트웨어 放出政策이란 소프트웨어 시스템의 테스트를 언제 멈추고서 사용자에게 放出하는냐 하는 테스트 중지규칙(stopping rule)을 결정하는 것이다[11].

소프트웨어의 最適放出政策은 여러사람에 의해 연구되어 왔는데 1983년에 Shanthikumar와 Tufekel[12]은 Binomial 信賴度모형을 사용하여 PDP(Probabilistic Dynamic Programming)로 소프트웨어 放出문제를 정식화하고 이 문제와 관련된 비용함수의 성질을 이용하여 最適放出시간을 결정하는 最適信賴度의 앨고리듬을 개발하였다. 같은 해 Koch와 Kubat[7]는 J-M(Jelinski-Moranda)모형을 사용하여 소프트웨어를 放出할 시점을 결정하는 의사 결정절차를 비용-이익 분석을 바탕으로해서 결정하였는데 지정된 시점이 지나서 소프트웨어 시스템을 放出할 때에는 그에 대한 벌칙비용이 부과되도록 하였고 그 시점 이전에 放出 할 때에는 그 시점까지의 費用을 할인하여 이익을 총비용에 고려하여 총비용이 最小가 되는 시점을 最適放出시점으로 하였다. 1984년 Yamada, Narihisa와 Osaki[14]는 NHPP(Non-Homogeneous Poisson Process) 모형을 사용하여 납기를 어길경우에 부과되는 벌칙비용이 납기후 지연된 放出시간에 선형으로 그리고 지수적으로 증가할 때 납기시간이 상수인 경우와 임의의 분포를 갖는 확률 변수일 때 총 기대비용을 最小로 하는 放出政策을 발표하였다. 1985년 Yamada와 Osaki[15]는 NHPP 모형을 사용하여 총 기대비용이 最小가 되고 요구되는 소프트웨어 信賴度를 만족하는 最適 소프트웨어 放出政策을 발표하였다. 1986년 Yamada와 Osaki[16]는 NHPP 모형을 사용하여 총 기대 소프트웨어 비용과 소프트웨어 信賴度를 평가 기준으로 해서 테스트 기간과 시스템의 放出이후에 발생하는 에러의 종류를 Type I 에러(에러의 검출이 쉬운 것)와 Type II 에러(에러의 검출이 어려운 것)로 나누어 두가지 평가기준을 이용한 소프트웨어 放出政策을 발표했다. 1988년 Bai와 Yun[1]은 J-M 모형과 DFR모형을 이용하여 수정에러갯수로서 소프트웨어의 最適放出시점을 결정하는 政策을 발표하였다. 같은 해 Catuneanu, Moldouan, Popentiu 그리고 Gheorghiu[2]는 Yamada와 Osaki의 소프트웨어 信賴度모형에 대한 最適放出政策을 SLUMT기법을 사용하여 결정하였다.

이상의 연구들을 살펴보면 소프트웨어 시스템의 最適放出政策을 결정하는데 있어서 여러 고장발생모형에 평균고장시간이나 검출에러수등을 고려하면서 여러가지 비용요소와 信賴度등을 결합한 政策들을 제시하였다. 본 연구에서는 이런 商業用(business-related) 소프트웨어 시스템을 대상으로 테스트를 얼마나 실시한 후에 放出하는 것이 테스트단계에서의 소요비용-테스트비용과 테스트단계에서의 에러수정비용과 필드에서 에러 발생으로 초래되는 費用을 고려해서 총 기대 소프트웨어비용을 最小로 하는 最適 테스트횟수를 결정하므로써 소프트웨어 시스템

放出政策을 제시한다.

2. 模型의 設定 및 用語說明

2.1 模型의 設定

소프트웨어 시스템이 상품화되어 필드로 내보내지기 전에 테스트단계를 거치게 되는데 이 단계는 소프트웨어 개발단계중에서 검출된 소프트웨어 에러를 수정하기 위해 인력, 시간, 장비, 기술 등이 투입되기 때문에 상당한 비중을 차지하게 된다. 또한 이 때에 테스트 방법은 중요하며 선택되는 입력데이타 집합(input data set)에 따라 테스트 결과에 영향을 미치게 된다. 그래서 검사자는 입력데이타 집합의 선택에 있어서 실제로 필드로 소프트웨어 시스템이 放出되었을 때 실행이 예상되는 모든 경우의 입력데이타 집합을 테스트 대상으로 삼아야 한다. 그런데 이런 입력데이타 집합 중에는 소프트웨어 시스템에 에러를 유발시킬 수 있는 데이타의 집합이 있을 것이다. 그래서 검사자는 에러를 발생시킬 데이타의 집합을 선택해서 소프트웨어의 에러를 발견하고 수정하는 것이 바람직할 것이다. 그러나 에러를 유발할 입력데이타 집합들은 전체 입력데이타 집합에 랜덤하게 분포되어 있고, 또한 입력데이타 집합의 수는 상당히 많기 때문에 에러를 발견위해 모든 경우의 입력 데이타를 전부 테스트하는 것은 소프트웨어 시스템의 品質은 높아지겠지만 테스트비용과 에러 발생으로 인한 수정비용이 증가하게 되므로 비현실적일 것이다. 또한 테스트를 느슨하게 한다면 테스트단계에서의 비용은 감소하지만 品質이 떨어져서 필드에 에러 발생이 증가함에 따라 초래되는 費用이 증가하게 되는 절충관계(trade-off)가 발생한다.

검사자는 전체 입력데이타 집합중에서 임의로 하나의 입력데이타 집합을 선택해서 테스트를 하고 도중에 에러를 발견하게 되면 에러수정을 하게된다. 테스트된 입력데이타 집합은 다음 선택에서 제외시키게 될 것이다. 이러한 테스트 작업이 계속해서 독립적으로 실행되고서 소프트웨어 시스템은 사용자(user)에게 放出될 것이다. 필드에서 사용자의 특성은 선택한 입력 데이타가 테스트단계에서 사용된 것인지에 대해서는 관심이 없고 테스트단계에서 실행했던 것과 같은 방법으로 입력데이타 집합을 사용하여 소프트웨어 시스템을 실행 할 것이다.

費用의 측면에서 보면 테스트단계에서 소요되는 비용으로서 소프트웨어가 放出될 때까지 테스트에 소요되는 비용으로 CPU사용 비용과 인건비 등을 포함하는 테스트비용과 테스트중에 에러가 발생했을 때 시스템이 원래와 같이 동작되도록 에러를 수정하는데 소요되는 에러수정비용 그리고 소프트웨어 시스템이 필드로 放出된 후에 에러의 발생으로 인해서 초래되는 비용으로 에러수정비용과 에러 발생으로 인한 피해보상비용 등을 포함하는 필드에서 발생하는 費用으로 구분할 수 있다. 본 연구에서 필요한 가정은 다음과 같다.

1) 1회의 테스트에는 하나의 입력데이타 집합이 사용된다.

2) 1회의 테스트에서 발견될 수 있는 에러 수는 최대1개이다.

3) 발견된 에러는 즉시 수정되며 수정 도중 또 다른 에러의 유입은 없다.

4) 테스트는 중복 테스트를 허용치 않는다.

5) 입력데이타 집합들은 무한하며 서로 독립이다.

2.2 用語說明

본 논문에서 사용되는 용어는 다음과 같다.

$C(n)$: 총 기대 소프트웨어비용

C_1 : 테스트단계에서 1회 테스트할 때 소요되는 비용

C_2 : 테스트단계에서 1개의 에러를 수정하는데 소요되는 비용

C_3 : 필드에서 1개의 에러발생으로 초래되는 비용

N : 서로 다른 총 입력데이타 집합의 수

E : 총 입력데이타 집합들 중에서 에러를 유발시킬 입력데이타 집합의 수

E' : 필드초기에 입력데이타 집합들 중에서 에러를 유발시킬 입력데이타 집합의 수

n : 테스트 횟수

M : 필드에서 소프트웨어가 중복되지 않고 사용되는 횟수

p_m : 검사자가 테스트단계에서 에러를 발견하지 못할확률

f_n : n회 테스트 후에 기대 에러 확률

k : 비용비율

u : 테스트단계에서 검출되는 기대 에러 수

u_n : 필드에서 수명기간 동안 발생할 기대 에러 수

T : 테스트 시간

θ : 비례상수

3. 模型의 展開

3.1 費用分析

검사자는 비용요소들을 고려하면서 테스트를 하게 되고 어느시점에서 사용자에게 放出할 것인가를 결정하게 된다.

이 소프트웨어 放出시점에서의 총 기대 소프트웨어 비용함수 $C(n)$을 정식화하면 다음과 같다.

(1) 테스트費用

테스트비용은 소프트웨어 放出시점까지의 테스트비용으로서 1회의 테스트에 소요되는 비용 c_1에다 실행된 테스트 횟수 n을 곱한 것으로 표시된다. 즉,

$$c_1 \cdot n \quad \cdots\cdots(1)$$

이다.

(2) 테스트단계에서의 에러수정비용

테스트단계에서 검사자는 총 입력데이타 집합 중에서 한번 테스트 할 때마다 하나의 입력데이타 집합만을 선택해서 테스트를 하고 거기에서 에러가 발견되면 바로 수정하게 된다. 따라서 1회 테스트에서 에러가 발견될 확률 p는 총 입력데이타 집합 N중에서 에러를 유발시킬 입력데이타 집합 E를 선택하는 경우이므로

$$p = E/N \quad \cdots\cdots(2)$$

로 나타낼 수 있다[13].

그리고 여기서 선택된 입력데이터 집합은 테스트 후에 전체 입력데이타 집합에 다시 복원시키지 않는다. 이와 같은 특성은 초기하 분포를 모형화할 수 있으므로 N이 상당히 크다면 이항분포로 근사하게 된다. 이때, n회의 연속독립 테스트를 했을 때 기대에러 수를 u라 하면

$$u = n \cdot p = n \cdot (E/N) \quad \cdots\cdots(3)$$

가 된다.

따라서 테스트단계에서의 에러수정비용은 하나의 에러를 수정할 때의 비용 c_2에다 테스트단계에서 발견될 기대에러수를 곱한 것이 된다. 즉, 식(2)와 (3)에 의해

$$c_2 \cdot (E/N) \cdot n = c_2 \cdot p \cdot n \quad \cdots\cdots(4)$$

로 나타낼 수 있다.

(3) 필드에서 발생하는 費用

검사자가 n번의 연속 독립 테스트에서 에러를 발견하지 못할 확률 p_m은 식(2)에 의해

$$p_m = (1 - E/N)^n \quad \cdots\cdots(5)$$

으로 나타낼 수 있다[13].

그리고 n회의 테스트가 행해진 후 j개의 에러가 있을 확률은 $p(j|n)$인 조건부 확률로 나타낼

수 있다. 이 조건부 확률은 다음과 같이 근사시킬 수 있다.

β_n를 n회 테스트 후의 에러의 발생확률이라 하면 식(3)에 의해

$$\beta_n = (E-u)/N$$

$$= (E-(E/N)\cdot n)/N$$

$$= p\cdot(1-(n/N)) \quad \cdots\cdots(6)$$

으로 나타낼 수 있다.

또한 M을 필드에서 소프트웨어가 그 수명기간동안 반복되지 않고 사용될 수 있는 횟수라고 하면 소프트웨어의 수명기간동안 발생할 기대에러 수 u_n은 M이 매우 크게 될 때, 식(6)으로부터

$$u_n = M\cdot\beta_n$$

$$= M\cdot p(1-(n/N)) \quad \cdots\cdots(7)$$

으로 나타낼 수 있다.

그래서 n회 테스트 후에 j개의 필드 에러가 잔존해 있을 조건부 확률은 다음과 같이 poisson 분포에 근사시킬 수 있다. 즉,

$$p(j|n) = (u_n^j/j!)\cdot e^{-u_n},\ j=0,1,2,\cdots,E' \quad \cdots\cdots(8)$$

단, $E' = E-u$이다.

여기서 기대되는 에러수는 $\sum_{j=0}^{E'} j\cdot p(j|n)$이 되고 이것은 r, v, j의 조건부 기대치를 나타내고 거의 u_n와 같다.

그러므로 필드에서 발생하는 비용은 하나의 에러 발생으로 인해 초래되는 비용 c_3와 테스트 단계에서 n회의 독립 테스트에서 에러를 검출하지 못할 확률, 그리고 n회 테스트가 행해졌을 때 수명기간동안 발견될 기대 에러 수를 곱한 것이 된다. 즉, 식(5), (6), (7), (8)에 의해

$$c_3\cdot p_m\cdot\sum_{j=0}^{E'} j\cdot p(j|n)$$

$$= c_3\cdot u_n\cdot(1-p)^n$$

$$= c_3\cdot M\cdot p\cdot\{1-(n/N)\}\cdot(1-p)^n \quad \cdots\cdots(9)$$

이다.

이 결과 총 기대 소프트웨어비용 $C(n)$은 식(1), (4), 그리고 (9)를 합한 것이 된다. 즉,

$$C(n) = c_1\cdot n + c_2\cdot(E/N)\cdot n + c_3\cdot p_m\cdot\sum_{j=0}^{E'} j\cdot p(j|n)$$

$$= c_1\cdot n + c_2\cdot p\cdot n + c_3\cdot u_n\cdot(1-p)^n$$

$$= c_1\cdot n + c_2\cdot p\cdot n + c_3\cdot M\cdot p\cdot\{1-(n/N)\}\cdot(1-p)^n \quad \cdots\cdots(10)$$

이다.

3.2 소프트웨어 시스템의 最適放出政策

소프트웨어 시스템의 最適放出政策은 총 기대 소프트웨어비용을 最小로 하는 테스트 횟수를 구하는 것이다.

이에 대한 목적함수식은

$$\text{Minimize } C(n) \quad \cdots\cdots(11)$$

subject to $n \geq 0$, $c_3 > c_1, c_2$

$c_1, c_2, c_3 > 0$

$0 < p < 1$

이때 n^*를 $dC(n)/dn = 0$을 만족시키는 점이라 하고, n^*점에서 $d^2C(n)/dn^2 > 0$이면, n^*는 식(11)의 목적식값을 最小로 하는 점이 된다.

그래서 $dc(n)/dn$, $d^2C(n)/dn^2$을 구해보면 다음과 같다.

식(10)과 (11)에 의해

$$C(n) = c_1 \cdot n + c_2 \cdot p \cdot n + c_3 \cdot M \cdot p \cdot \{1-(n/N)\} \cdot (1-p)^n$$

과 같이 된다. 이것을 1차 미분하면

$$\begin{aligned} dc(n)/dn &= c_1 - c_2 \cdot p + c_3 \cdot M \cdot p[\{1-(n/N)\} ln(1-p) \cdot (1-p)^n - (1/N) \cdot (1-p)^n] \\ &= c_1 + c_2 \cdot p + c_3 \cdot M \cdot p \cdot (1-p)^n \cdot [\{1-(n/N)\} \cdot ln(1-p) - (1/N)] \quad \cdots\cdots(12) \end{aligned}$$

여기서 식(12)를 0으로 하는 n^*의 값은 c_1, c_2, N, E, M, 이 주어질 때 수치해법에 의해 구할 수 있다.

다시 2차 미분하면

$$\begin{aligned} d^2C(n)/dn^2 &= c_3 \cdot M \cdot p \cdot [\{1-(n/N)\} (1-p)^n \cdot \{ln(1-p)\}^x - (1/N) \cdot (1-p)^n \cdot ln(1-p) - (1-N) \cdot (1-p)^n \cdot ln(1-p)] \\ &= c_3 \cdot M \cdot p \cdot [\{1-(n/N)\} (1-p)^n \cdot \{ln(1-p)\}^x - (2/N) \cdot (1-p)^n \cdot ln(1-p)] \\ &= c_3 \cdot M \cdot p \cdot (1-p)^n \cdot [\{1-(n/N)\} \{ln(1-p)\}^x - (2/N) \cdot ln(1-p)] \quad \cdots\cdots(13) \end{aligned}$$

과 같이 된다.

식(13)이 0보다 크므로, n^*에서 비용을 最小로 하는 最適해를 구할 수 있음을 알 수 있다.

4. 數値例

본 연구에서는 여러 파라메터의 값이 다음과 같이 주어졌을 때 총 기대 소프트웨어비용이 最小로 되게 하는 最適테스트횟수 n^*를 구하고 기존의 연구와 비교하는 수치예를 보인다.

파라메터들의 값이 다음과 같을 때, n^*와 그 때의 총 기대 소프트웨어비용 $C(n^*)$의 값을 구해 보면,

$c_1 = 10$ $c_2 = 2000$ $c_3 = 3500$ $N = 3000$ $E = 50$ $M = 2540$ 일 때

이 값을 식(12)에 대입하여 풀면 $n^* = 2150$이 되고 이 때의 총 기대 소프트웨어비용은 식(11)에 의하여 $C(2150) = 93310$이 된다.

이 결과를 Koch와 Kubat[7]의 예를 대상으로 기존의 政策과 제안된 政策을 이익함수를 사용하여 비교해 보자.

먼저 이익함수를 $G(n)$이라 하고 이 함수식을 구해보면, 필드에서 에러 발생으로 초래되는 費用과 테스트단계에서의 에러수정 費用과의 차이 즉, 단위이익에 n회 테스트를 했을 때 기대되는 에러 수를 곱하면 총 기대이익이 된다. 이것에서 테스트단계에서 총 테스트비용 즉, n회 테스트 했을 때 기대 에러 수가 발견될 때까지의 기대시간에 테스트 단계에서의 비용비율을 곱한 것과의 차로서 기대이익을 얻을 수 있다. 즉,

$$G(n)=(c_3-c_2)\cdot n\cdot(E/N)$$

$$-k\cdot E\{T_{n\cdot\frac{E}{N}}\}$$

$$c\cdot n\cdot p-k\cdot E\{T_{n\cdot p}\} \quad (14)$$

단, $c=c_3-c_2$이다.

Koch와 Kubat[7]의 예에서 다음과 같이 수치가 주어졌을 때, $k=21500$, $c_2=2000$, $c_3=3500$, $E=50$, $\theta=1.6$ 기대이익을 구해보면 다음 표와 같은 결과를 얻게 된다.

Koch와 Kubat 政策		제안된 政策	
T^*	0.77	$E\{T_{n^*\cdot p}\}$	0.796
$E\{1-\exp(-T^*\theta)\}$	35.41	$n^*\cdot p$	36
$G(T^*)$	36564.3	$G(n^*)$	36886.0

단, $G(T)=cE\{1-\exp(-T\theta)\}-k\cdot T$[1]

이 결과 Koch와 Kubat의 政策에 의한 이익 $G(T^*)$은 36564.3인데 비해 제안된 政策에 의한 이익 $G(n^*)$은 36886.0으로 보다 크게 나타남을 알 수 있다.

5. 結 論

본 연구에서는 소프트웨어 시스템에 사용되는 입력데이터 집합을 이용하여 테스트단계에서 테스트 케이스(Case)로 사용되는 입력데이터의 집합을 바탕으로한 소프트웨어 最適放出政策에 대해 연구 하였다.

현재까지 소프트웨어의 放出政策에 대한 기존의 연구방향은 중지규칙(Stopping rule)으로서 평균고장시간이 어느 정도되었을 때 放出할 것인가 하는 것과 몇개의 에러가 검출될 때까지 테스트를 하고 放出할 것인가 하는 문제로 요약된다.

본 연구에서는 테스트단계와 필드에서 발생하는 소프트웨어 비용에 대한 비용분석을 하였으며 이를 이용하여 총 기대 소프트웨어비용을 最小로 하는 중지규칙을 정립하였다. 즉, 입력데이터 집합이 테스트되었을 때 총 기대 소프트웨어비용이 最小가 되도록 하는 중지규칙을 最適테스트횟수를 수리적으로 구하였고 이때의 이익함수를 도출하여 기존의 政策과 비교하였다. 그 결과 본 연구에서 제시한 放出政策에 의한 기대이익값이 기존의 연구에서의 기대이익값 보다 더 크게 됨을 보였다.

앞으로의 연구과제로는 제시된 政策에 에러수정 시 에러가 유입되는 경우, 또는 1회 테스트에서 발견될 수 있는 최대 에러수가 1개 이상인 경우등에 대한 연구가 계속 되어야 할 것이다.

參 考 文 獻

1. Bai, D. S., Yun, W. Y. (1988), "*Optimum Number of Errors Corrected before Releasing a Software System*," IEEE Trans, Reliability, Vol. 37, No. 1, pp. 41-44.
2. Catuneanu, V. M., Moleouan, C., Popentiu, FL., Gheorghiu, M., (1988), "*Optimal Software Release Polices Using SLUMT*," MicroelectronReliability, Vol. 28, No. 4, pp. 547-549.
3. Forman, E. H., Singpurwalla, N. D, (1979), "*Optimal Time Intervals for Testing Hypotheses on Computer Software Errors*," IEEE Trans. Reliability, Vol. R-28, No. 3, pp. 250-253.
4. Goel, A. L., Okumoto, K., (1979), "*Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures*," IEEE Trans. Reliability, Vol. R-28, No. 3, pp. 206-211.
5. Goel, A. L., (1985), "*Software Reliability Models : Assumptions, Limitation, and Applicability*," IEEE Trans, Software Engineering, Vol. SE-11, No. 12, pp120-134.
6. Jelinski, Z., Moranda, P., (1972), "*Software Reliability Research*," in Statistical Computer Performance Evaluation, W. Freiberger, Ed. New York ; Academic, pp. 465-485.
7. Koch, H. S., Kubat, P., (1983), "*Optimal Release Time of Software*," IEEE Trans, Software Engineering, Vol. SE-9, No. 3, pp. 323-327.
8. Kubat, P., Koch, H. S., (1983), "*Pragmatic Testing Protocols to Measure Software Reliability*," IEEE Trans, Reliability, Vol. R-32, No. 4, pp. 338-341.
9. Lipow, M., (1982), "*Number of Faults per Line of Code*," IEEE Trans, Software Engineering, Vol. SE-8, No. 4, pp. 437-439.
10. Musa, J. D., Iannino, A., Okumoto, Kazuhira., (1987), Software Reliability, Mcgraw-Hill Book Co..
11. Ross, S. M., (1985), "*Software Reliability : The Stopping Rule Problem*," IEEE Trans, Software Engineering, Vol. SE-11, No. 12, pp. 1172-1476.
12. Shanthikumar, J. G., Tufekci, S., (1983), "*Application of a Software Reliability Model to Decide Software Release Time*," Microelectron Reliability, Vol. 23, No. 1, pp. 41-59.
13. Shooman, M. L., (1983), Software Engineering, Mcgraw-Hill Book.
14. Yamada, S., Narihisa, H., Osaki, S., (1984), "*Optimum Release Policies for a Software System with a Scheduled Software Delivery Time*," Int. J. Systems Sci., Vol. 15, No. 8, pp. 905-914.
15. Yamada, S., Osaki, S., (1985), "*Cost-Reliability Optimal Release Policies for Software System*," IEEE Trans, Reliability, Vol. R-34, No. 5, pp. 422-424.
16. Yamada, S., Osaki, S., (1986), "*Optimal Software Release Policies for a Non-Homogeneous Software Error Detection Rate Model*," Microelectron Reliability, Vol. 26, No. 4, pp. 691-702.
17. Yang, M. C. K., Wackerly, D. I., .

Rosalsky, A., (1982), "*Optimal Stopping Rules In Proofreading*," J, Appl. Prob., Vol. 19, pp. 723-729.

18. Yu, T., Shen, V. Y., (1988), "*An Analysis of Several Software Defect Models*," IEEE Trans, Software Engineering, Vol. 14, No. 9, pp. 1261-1269.
19. Zhi, H. X., (1984), "*The Hypergeometric Distribution Model for Predicting the Reliability of Softwares*," Microelectron Reliability, Vol. 24, No. 1, pp. 11-20.