소프트웨어 오류개수에 근거한 최적 출시시점 결정
Determination of Optimal Software Release Time Based on Number of Errors

유 영 관* · 이 종 무* · 박 철 수*
Young-Kwan Yoo · Jong-Moo Lee · Cheol-Soo Park

Abstract

In this paper, a software release model is presented to determine the optimum testing time with consideration of software error type. The software errors are classified into two types, major and minor errors. The software testing is continued until the Nth major error is discovered and corrected. The total cost needed before and after testing time is modeled under nonhomogeneous Poisson error correction model. Numerical examples are presented to demonstrate the results.

1. 서 론

정보기술과 관련 산업의 눈부신 발전에 따라 현대 사회에서는 다방면에 걸쳐 다양한 형태로 컴퓨터가 응용되고 있다. 항공관제 시스템, 핵반응 통제 시스템, 항공기, 군사 무기시스템, 생산 공정 통제 시스템, 그리고 병원에서의 환자 관찰 시스템 등은 컴퓨터 응용의 대표적인 예들이다. 또한 컴퓨터의 기능이 점점 필수적으로 요구되고 응용 소프트웨어의 크기와 복잡성이 증가함에 따라 컴퓨터 소프트웨어의 신뢰도에 대한 요구가 크게 증가하고 있다.

소프트웨어는 노후화하지 않는다는 점에서 하드웨어 신뢰도와 근본적인 차이점이 있다. 또한 소프트웨어 자체는 고장 나지 않으며, 소프트웨어 내의 결함이나 오류(fault, error)가 프로그램 작동으로 인해 드러나야만 고장(failure)으로 이어진다[14].

* 한라대학교 경영학과

소프트웨어가 복잡해지고 거대화함에 따라 소프트웨어 개발 비용은 크게 증가하고 있다. 그리고 소프트웨어의 크기가 증가함에 따라 소프트웨어의 결함은 지수적으로 증가하며, 이러한 결함의 수정에 막대한 비용이 소요된다. 한 조사에 의하면 전문적인 프로그래머는 1,000 LOC(lines of code) 당 평균 6개의 소프트웨어 결함을 저지르는 것으로 나타났는데, 이러한 결함 비율이라면 350,000 LOC를 갖는 보통의 상용 소프트웨어의 경우 2,000개 이상의 결함을 갖고 있는 것으로 계산된다[14].

소프트웨어의 고장으로 인한 손실을 방지하기 위해서는 소프트웨어의 운용에 앞서 충분한 시험이 필요하다. 일반적으로 소프트웨어 개발 과정은 분석, 설계, 구현, 시험, 운용단계로 나누어지는데, 소프트웨어의 목적에 따라 설계와 코딩이 완성되면 실제 운용에 들어가기에 앞서 소프트웨어의 결함을 찾아내기 위한 시험이 실시되며 이 작업은 시험 단계에서 이루어진다.

시험을 오래 할수록 소프트웨어 내의 결함을 찾아내어 제거할 수 있는 확률은 늘어나므로 소프트웨어의 신뢰성은 증가한다. 그러나 이 경우 시험 비용의 증가로 전체적인 개발 비용이 증가하게 되며, 또한 고객에게 인도해야 할 납기일 때문에 무한정 시험 기간을 늘릴 수도 없다. 반대로 시험 기간을 줄이면 전체적인 개발 비용은 줄어들지만 납품하는 소프트웨어의 신뢰성은 줄어들게 되므로 실제 운용 단계에서 빈번한 고장이 발생할 가능성이 커진다. 운용단계에서의 소프트웨어 고장은 시험단계에서의 고장에 비해 일반적으로 훨씬 큰 금전적 손실을 야기할 뿐만 아니라 고객의 신용을 잃게 되므로 시험기간을 단축하는 것도 바람직하지 않다. 따라서 전체적인 소프트웨어 개발 관련 비용을 최소화 해줄 수 있는 적절한 시험기간의 결정은 매우 중요한 일이며 소프트웨어 개발자들의 주요 관심사라고 할 수 있다. 소프트웨어의 시험 시간을 얼마로 정하는 것이 소프트웨어의 신뢰성이나 수명주기 총 비용 면에서 최적인가의 문제를 최적 소프트웨어 출시문제(optimal software release problem) 또는 최적 소프트웨어 시험시간 문제(optimal software testing time problem)라고 한다.

이 분야의 고전적 연구는 Okumoto & Goel[13]의 연구를 들 수 있으며, 이는 그들의 비제차 포아송과정(NHPP) 소프트웨어 신뢰도 성장모형[6]을 기반으로 한 것이다. 그들은 소프트웨어 출시시점을 결정하기 위한 기준으로 소프트웨어 신뢰도와 수명주기 비용을 각각 제시하였으며, 후에 Yamada & Osaki[19]는 이 두 기준을 동시에 충족할 수 있는 방안을 제시하였다. 이후부터 많은 연구자들에 의해 다양한 상황과 요소들을 고려한 소프트웨어 시험시간 결정 모형들이 개발되어왔다[1, 2, 4, 5, 7-12, 15, 16, 18, 20].

그러나 이전의 연구들은 소프트웨어 오류의 유형을 구분하지 않고 동일하다고 가정하고 있다. 소프트웨어의 운용 중 고장 유형은 일반적으로 다음과 같이 분류될 수 있다[14].

1)파국적(catastrophic) 고장: 재앙적인 결과를 야기하는 고장. 예를 들면 항공관제 소프트웨어의 고장 등.
2)치명적(critical) 고장: 재앙 적이지만 회복 가능한 고장. 재산상의 피해만 야기하거나, 인간에게 끼친 피해가 회복될 수 있는 정도의 고장.

3)중대한(major) 고장: 심각한 고장이긴 하지만 인간에 대한 신체적인 피해나 다른 시스템에 대한 피해는 없는 고장.
4)경미한(minor) 고장: 소프트웨어 시스템이나 사용자에게 부분적인 불편함을 야기하는 고장.

본 연구에서는 상기의 네 가지 오류 유형을 단순하게 중대오류와 경미오류의 두 가지 유형으로 크게 나누고 이를 소프트웨어 출시시점의 결정에 이용하고자 한다. 소프트웨어의 신뢰도는 Goel & Okumoto[6]의 NHPP 모형을 따른다고 가정하며, 소프트웨어의 총 오류는 중대오류와 경미오류의 일정 비율로 구성된다. 소프트웨어의 시험은 N번째 중대오류가 발견되어 수정될 때까지 지속되며 이후 소프트웨어는 출시된다. 이때 소프트웨어의 수정은 즉시 이루어진다고 가정한다.

본 모형은 이전의 대부분의 연구들이 시험시간(time)을 결정변수로 이용한 것과는 달리 오류개수를 결정변수로 이용하고 있다. 오류개수를 결정변수로 이용한 이전의 연구로는 Bai & Yun[3]과 Shinohara et al.[17]을 들 수 있다. Bai & Yun[3]은 Jelinski-Moranda 모형과 DFR(감소 고장률) 모형에 대해 Koch & Kubat[11]의 average gain 기준을 이용하여 최적 오류개수를 구하였다. Shinohara et al.[17]은 Jelinski-Moranda 모형과 LPET(logarithmic Poisson execution time)모형을 이용하여 수명주기의 총비용을 최소화하는 최적 시험시간과 오류개수를 각각 구하고 이들의 비용을 비교하였다. 본 연구에서는 소프트웨어의 오류를 두 가지 유형으로 분류하고, Goel-Okumoto 모형을 이용하여 수명주기의 총비용을 최소화하는 최적 중대오류 개수를 구하는 모형을 제시한다.

2. 중대오류 개수 모형

2.1 Goel-Okumoto 모형

본 연구에서 사용하는 Goel-Okumoto 소프트웨어 신뢰도 성장모형[6]은 다음과 같이 정리된다. 소프트웨어의 시험에 따라 오류가 검출되며 이 오류는 검출 즉시 수정된다. 이때 오류의 도래(arrival)는 NHPP를 따르며, 소프트웨어의 시험이 진행됨에 따라 신뢰도는 증가하게 된다. 이제 $N(t)$를 시간 t까지의 오류 개수라고 하자. 그러면 $\{N(t), t \geq 0\}$는 NHPP이므로

$$P\{N(t)=k\}=\frac{[m(t)]^k e^{-m(t)}}{k!}, k=0,1,2,... \quad (1)$$

이고, NHPP의 mean value function $m(t)$와 intensity function $\lambda(t)$는 각각

$$m(t) = E[N(t)] = a(1 - e^{-bt}) \tag{2a}$$
$$\lambda(t) \equiv m'(t) = abe^{-bt} \tag{2b}$$

로 주어진다. 여기에서 $a = m(\infty)$, 즉 a는 이 소프트웨어의 총 오류개수의 기대값을 나타내며, b는 오류 당 고장률(hazard rate of an error)을 의미한다.

2.2 오류검출과정

본 연구의 모형개발을 위해 소프트웨어의 오류 중 중대오류의 비율을 p, 경미오류의 비율을 $q = 1 - p$라고 하자. 또한 다음과 같이 기호를 정의하자.

$N_1(t)$ = 시간 t까지의 중대오류의 개수
$N_2(t)$ = 시간 t까지의 경미오류의 개수

그러면 Poisson과정의 특성[21]에 의해 $\{N_1(t), t \geq 0\}$와 $\{N_2(t), t \geq 0\}$는 각각 독립적인 NHPP가 되며 $N(t) = N_1(t) + N_2(t)$이고, 이들의 mean value function은 각각 다음과 같다.

$$E[N_1(t)] = pm(t) = pa(1 - e^{-bt}) \tag{3a}$$
$$E[N_2(t)] = qm(t) = qa(1 - e^{-bt}) \tag{3b}$$

이제 T_N을 N번째 중대오류가 발생할 때까지의 시간이라고 하자. 그러면

$$\{T_N > t\} \Leftrightarrow \{N_1(t) < N\} \tag{4}$$

이므로 T_N의 생존함수(survival function)는 다음과 같이 구해진다.

$$P\{T_N > t\} = P\{N_1(t) < N\} = \sum_{k=0}^{N-1} P\{N_1(t) = k\} = \sum_{k=0}^{N-1} \frac{[pm(t)]^k e^{-pm(t)}}{k!} \tag{5}$$

따라서 N번째 중대오류의 검출까지의 평균시간은 다음과 같다.

$$E[T_N] = \int_0^\infty P\{T_N > t\} dt = \sum_{k=0}^{N-1} \int_0^\infty \frac{[pm(t)]^k e^{-pm(t)}}{k!} dt \tag{6}$$

한편 식(5)로부터 T_N의 밀도함수를 다음과 같이 구할 수 있다.

$$f(t) = -\frac{d}{dt}P\{T_N > t\} = \sum_{k=0}^{N-1}\frac{1}{k!}[p\lambda(t)e^{-pm(t)}[pm(t)]^{k-1}(pm(t)-k)]$$
$$= p\lambda(t)e^{-pm(t)}\left[\sum_{k=0}^{N-1}\frac{[pm(t)]^k}{k!} - \sum_{k=1}^{N-1}\frac{[pm(t)]^{k-1}}{(k-1)!}\right]$$
$$= \frac{[pm(t)]^{N-1}e^{-pm(t)}}{(N-1)!}p\lambda(t),\ t>0 \qquad (7)$$

2.3 수명주기 비용모형

소프트웨어의 시험은 N번째 중대오류를 검출할 때까지, 즉 T_N까지 지속된 후 출시된다. 이러한 출시정책하의 수명주기 기대비용을 구하기 위해 다음과 같이 기호를 정의하자.

c_{m1} = 시험기간 중 중대오류의 개당 수정비용
c_{m2} = 운용기간 중 중대오류의 개당 수정비용
c_{n1} = 시험기간 중 경미오류의 개당 수정비용
c_{n2} = 운용기간 중 경미오류의 개당 수정비용
c_r = 시험기간 중 단위시간당 시험비용

1) $(0, T_N]$동안의 시험비용

시험은 N번째 중대고장까지, 즉 T_N까지 지속되므로 이 기간 동안의 평균 시험비용은 식(6)으로부터 다음과 같다.

$$c_rE[T_N] = c_r\sum_{k=0}^{N-1}\int_0^\infty \frac{[pm(t)]^k e^{-pm(t)}}{k!}dt \qquad (8)$$

2) $(0, T_N]$동안의 오류 수정비용

중대오류의 수정비용은 단순히 $c_{m1}N$이다. 경미오류의 수정비용을 구하기 위하여 먼저 이 기간 동안의 경미오류 개수의 기대값이 필요한데 이는 식(3b)와 (7)로부터 다음과 같이 구해진다.

$$\text{E}[(0, T_N]\text{의 경미오류 개수}] = \int_0^\infty qm(t)f(t)dt = qp^N\int_0^\infty \frac{\lambda(t)m(t)^N e^{-pm(t)}}{(N-1)!}dt \qquad (9)$$

따라서 이 기간 동안의 오류수정비용은 다음과 같다.

$$c_{m1}N+c_{n1}qp^N\int_0^\infty \frac{\lambda(t)m(t)^N e^{-pm(t)}}{(N-1)!}dt \tag{10}$$

3) (T_N,∞)동안의 오류 수정비용

소프트웨어의 총 오류개수의 기대값은 $a=m(\infty)$이므로 중대오류의 수정비용은 $c_{m2}(ap-N)$이다. 비슷하게 경미오류에 대한 수정비용은 식(9)를 이용하면 T_N 이후의 경미오류 개수를 구할 수 있으므로 이 기간 중의 총 오류 수정비용은 다음과 같이 구해진다.

$$c_{m2}(ap-N)+c_{n2}q\left[a-p^N\int_0^\infty \frac{\lambda(t)m(t)^N e^{-pm(t)}}{(N-1)!}dt\right] \tag{11}$$

식(8)과 (10), 그리고 (11)을 더하면 소프트웨어 수명주기 상의 총비용이 다음과 같이 정리된다.

$$\begin{aligned} C(N)=c_r\sum_{k=0}^{N-1}\int_0^\infty \frac{[pm(t)]^k e^{-pm(t)}}{k!}dt+c_{n1}qp^N\int_0^\infty \frac{\lambda(t)m(t)^N e^{-pm(t)}}{(N-1)!}dt \\ +c_{n2}q\left[a-p^N\int_0^\infty \frac{\lambda(t)m(t)^N e^{-pm(t)}}{(N-1)!}dt\right]+c_{m2}ap+N(c_{m1}-c_{m2}) \end{aligned} \tag{12}$$

이때 중대오류의 비율 p값이 1인 경우, 즉 $q=0$이라면 식(12)는 다음과 같이 간략화 된다.

$$C(N)=c_r\sum_{k=0}^{N-1}\int_0^\infty \frac{m(t)^k e^{-m(t)}}{k!}dt+c_{m2}a+N(c_{m1}-c_{m2}) \tag{13}$$

식(13)은 오류의 유형을 구분하지 않고 N번째 오류수정에서 시험을 종결하고 출시하는 정책하의 비용함수를 표현하는 것으로 해석할 수 있다. 단, 이 경우 각 비용모수들의 기호를 새로이 정의해야한다.

3. 최적해를 위한 수치분석

식(12)는 결정변수 N에 대해 복잡한 형태로 되어 있어 해석적으로 분석하기에는 어려움이 있다. 본 절에서는 식(12)에 대해 컴퓨터를 이용한 수치분석을 통하여 최적해를 구해본다.

식(12)를 수치분석하기 위해서는 모형의 모수들에 대한 값이 필요하다. 본 연구에서는 이진승 등[2]과 박일광 · 공명복[1] 등 기존의 연구들을 참고하여 다음과 같은 모수

값들을 수치예제에 이용하였다.

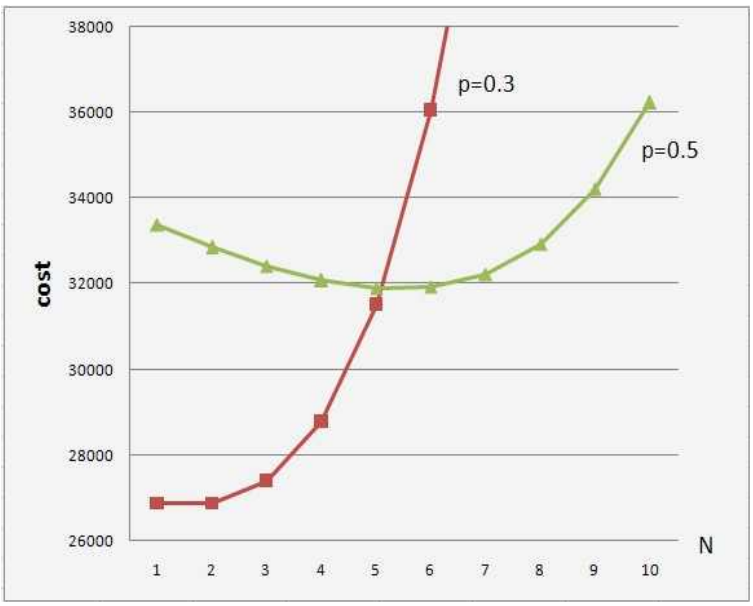
$$a = 34,\ b = 0.00579,\ c_r = 100,\ \ c_{m1} = 200,\ c_{n1} = 100,\ c_{m2} = 1500,\ c_{n2} = 500$$

식(12)의 수치분석을 위해 MATLAB을 이용하였으며, 다음의 <표 1>은 중대오류의 비율 p값을 변화시켜 가면서 그 계산 결과를 정리한 것이다.

<표 1> 중대오류의 비율에 따른 최적해의 변화

p	N^*	중대오류 기대개수(ap)	시험기간 (기대값)	$C(N^*)$
0.3	2	10.2	41.4	26878.0
0.5	5	17.0	64.0	31903.6
0.7	10	23.8	100.2	36107.6
1.0	18	34.0	138.6	41460.8

N의 값을 1부터 중대오류의 기대개수(ap)까지 변화시켜 가면서 식(12)를 계산한 결과 식(12)는 N에 대해 정수볼록(integer convex)임을 관찰할 수 있었다. 일부의 예로 다음의 <그림 1>은 $p = 0.3$과 $p = 0.5$인 경우에 대해 총비용의 변화를 관찰한 결과이다. 중대오류의 비율이 증가할수록 N의 최적값은 당연히 증가하며 이에 따라 총 비용도 증가함을 알 수 있다. 이 표에서 시험기간의 기대값은 식(6)으로부터 계산된 것이다.



<그림 1> N 값의 변화에 대한 총비용의 변화

4. 결 론

본 연구에서는 소프트웨어에 존재하는 오류들에 대해 중대오류와 경미오류의 두 가지 유형으로 분류하고 중대오류의 검출과 수정에 기반을 두는 소프트웨어 시험 및 출시정책을 제시하였다. 소프트웨어에 대한 시험은 일정 개수의 중대오류가 발견되어 검출될 때까지 지속되며 일정 개수의 중대오류 수정 후 즉시 출시된다. 이때 오류의 검출은 Goel-Okumoto NHPP 신뢰도 성장모형을 따르는 것으로 가정하였다.

본 연구에서 제시한 출시정책은 기존의 시간기준 정책과는 달리 오류의 개수가 결정변수라는 것과 오류의 유형을 분류하여 접근함으로써 좀 더 현실적이고 다양한 상황을 고려할 수 있다는 점이 특징이다. 그러나 시험기간의 종료시점을 미리 알 수 없고 단지 예상(기대)값만을 제시할 수 있다는 점이 단점이라고 하겠다.

한편 본 연구에서는 오류의 유형을 두 가지로 간단히 나누었으나 좀 더 현실적인 접근을 위해서는 세 가지 이상의 다유형(multi-type)으로 분류하는 것이 필요할 것이다. 이 경우 phase-type 분포가 유용한 분석도구로 사용될 수 있을 것이며, 이러한 부분과 함께 본 연구의 오류개수 기준 모형과 기존의 시간기준 모형과의 비용측면에서의 비교 분석은 차후 과제로 남긴다.

5. 참 고 문 헌

[1] 박일광, 공명복, “사용단계에서 주기적 서비스 팩 배포와 불확실한 패치 배포를 고려한 소프트웨어의 최적 출시시기”, 대한산업공학회지, 제33권, 제4호, pp487-493, 2007.

[2] 이진승, 나일용, 홍정식, 이창훈, “출시 후 보수를 고려한 소프트웨어의 최적 출시시기”, 대한산업공학회지, 제30권, 제4호, pp261-266, 2004.

[3] Bai. D., Yun, W., "Optimum number of errors corrected before releasing a software system", IEEE Tr. Reliability, Vol.37, No.1, pp41-44, 1988.

[4] Boland, P., Chuiv, N., "Optimal times for software release when repair is imperfect", Statistics and Probability Letters, Vol.77, pp1176-1184, 2007.

[5] Chatterjee, S., Misra, R., Alam, S., "Joint effect of test effort and learning factor on software reliability and optimal release policy", International J. of Systems Science, Vol.28, pp391-396, 1997.

[6] Goel, A., Okumoto, K., "Time-dependent error detection rate model for software reliability and other performance measures", IEEE Tr. Reliability, Vol.28, pp206-211, 1979.

[7] Hou, R., Kuo, S., Chang, Y., "Optimal release policy for hyper-geometric distribution software reliability growth model", IEEE Tr. Reliability, Vol.45, No.4., pp646-651, 1996.

[8] Hou, R., Kuo, S., Chang, Y., "Optimal release times for software systems with scheduled delivery time based on the HGDM", IEEE Tr. Computers, Vol.46, No.2, pp216-221, 1997.
[9] Huang, C., "Optimal release time for software systems considering cost, testing-effort, and test efficiency", IEEE Tr. Reliability, Vol.54, pp583- 591, 2005.
[10] Huang, C., "Cost-reliability-optimal release policy for software reliability models incorporating improvements in testing efficiency", J. of Systems and Software, Vol.77, pp139-155, 2005.
[11] Koch, H., Kubat, P., "Optimal release time of computer software", IEEE Tr. Software Engineering, Vol.9, No.3, pp323-327, 1983.
[12] Kumar, S., Zeephongsekul, P., Balasubramanian, S., "Modeling and analysis of a software system with improvements in the testing phase", Mathematical and Computer Modelling, Vol.22, pp183-191, 1995.
[13] Okumoto, K., Goel, A., "Optimal release time for software system based on reliability and cost criteria", J. of System and Software, Vol.1, pp315-318, 1980.
[14] Pham, H., Software Reliability, Springer, Singapore, 2000.
[15] Pham, H., Zhang, X., "A software cost model with warranty and risk costs", IEEE Tr. Computers, Vol.48, pp71-75, 1999.
[16] Pham, H., Zhang, X., "NHPP software reliability and cost models with testing coverage", European J. of Operational Research, Vol.145, pp443-454, 2003.
[17] Shinohara, Y., Dohi, T., Osaki, S., "Comparison of optimal release policies for software systems", Computers & Industrial Engineering, Vol.33, pp813-816, 1997.
[18] Yamada, S., Hishitani, J., Osaki, S., "Software- reliability growth with a Weibull test-efforts: A model & application", IEEE Tr. Reliability, Vol.42, pp100-106, 1993.
[19] Yamada, S., Osaki, S., "Cost-reliability optimal release policies for software systems", IEEE Tr. Reliability, Vol.34, No.5, pp422-424, 1985.
[20] Xie, M., Hong, G., "Software release time determination based on unbounded NHPP model", Computers & Industrial Engineering, Vol.37, pp165-168, 1999.
[21] Ross, S., Stochastic Processes, John Wiley & Sons, New York, 1983