OPTIMAL SOFTWARE RELEASE POLICY BASED ON WARRANTY AND RISK COSTS

이 종 형
건양대학교 컴퓨터학과
Email : chlee@konyang.ac.kr
장 규 범
한림대학교 수리정보과학부 정보통계학과
Email : tzarm@hallym.ac.kr
박 동 호
한림대학교 수리정보과학부 정보통계학과
Email : dhpark@hallym.ac.kr

요 약

컴퓨터 소프트웨어는 이제 우리의 일상적인 삶에서 필수불가결한 요소이며 시스템의 운용에 중요한 요인이 되었다. 최근에 들어서는 소프트웨어 비용이 하드웨어 비용을 초과하게 되면서 소프트웨어를 개발하는데 필요한 비용과 더불어 소프트웨어 고장에 의한 비용의 중요성이 더 커지게 되었다. 본 논문에서는 Non-Homogeneous Poisson Process(NHPP)에 기초한 소프트웨어 비용 모형을 제안하려고 한다. 개발초기단계에서 출시 전까지의 소프트웨어 개발비용과 테스트비용, 출시이후의 보증기간동안의 제반비용, 소프트웨어 고장에 의한 위험비용 등을 포함하는 소프트웨어 비용 모형을 제안하고 소프트웨어의 최적 출시시기를 결정하는 효과적인 정책을 제시하려고 한다.

1. 소 개

컴퓨터의 고장은 가전제품의 고장을 일으켜 생활에 불편을 줄 수가 있으며 은행체계 등의 혼잡을 야기해 경제적인 손실을 주기도 하며, 나아가서 항공체계나 의학 소프트웨어 고장으로 생명까지 위협할 수가 있다. 따라서 소프트웨어의 신뢰성은 현대의 소프트웨어 개발에 중요한 관심사중의 하나이다.

소프트웨어의 개발과정은 설계, 디자인, 코딩 그리고 테스팅의 네 가지로 구성되어 있다. 처음 세 단계에서 나타난 많은 소프트웨어 고장들이 테스팅 단계에서 대부분 발견되고 수정되어진다. 이러한 의미에서 테스팅 단계는 사용자들에 대한 소프트웨어의 질을 보증하고 확증하는데 매우 중요하다. 소프트웨어 질을 나타내는 특징들은 기능성, 신뢰성, 유용성, 효율성, 유지 보수성으로 알려져 있는데 그 중에서도 소프트웨어의 신뢰성은 특히 중요한 특징이다. 이것은 일정한 기간 동안에 소프트웨어가 고장이 없이 작동하는 확률로서 정의된다. 소프트웨어의 고장은 소프트웨어에 내재된 에러에 의해서 야기되는 것으로 정상적인 프로그램 운용에서 받아들여질 수 없는 이탈로서 정의된다. 만약 소프트웨어에 남아있는 에러의 총수가 시험단계에서 정확하게 추정될 수 있다면 소프트웨어의 신뢰성은 양적으로 측정되어질 수 있다. 이것은 시험의 향상을 조절할 수 있고 소프트웨어가 출하되는 시점을 예측 가능하게 한다.

이러한 이유로 시험단계 동안에 관찰된 데이터를 분석하고 개발된 소프트웨어의 신뢰성을 평가하기 위한 소프트웨어 신뢰성에 대한 연구가 활발하게 진행되고 있으며 이러한 연구를 위한 가장 유용한 모델중의 하나가 소프트웨어 신뢰성 성장 모형이다. 이 모형은 시험단계 동안에 고장출현 현상 또는 에러 발견 현상을 나타내기 위한 소프트웨어 신뢰성의 성장과정에서 발견된 에러의 수와 소프트웨어 시험의 시간 사이의 관계를 의미한다. 소프트웨어 신뢰성 모형 하에서 기대된 초기 고장의 수, 시험동안에 임의의 시간에서 남아있는 에러의 기대수, 고장들간의 평균 간격시간, 소프트웨어 신뢰성 등을 예측하고 추정하는 여러 가지 방법들이 문헌에 소개되어 있다.

일반적으로 소프트웨어는 오류가 전혀없이 완전하게 출하하는 것이 불가능하기 때문에 소프트웨어의 출하시기를 결정하는 문제는 프로그램 개발자들에게 있어서 대단히 중요한

관심사 중의 하나라고 할 수 있다. 시험기간이 너무 짧을 경우에는 낮은 신뢰성으로 인하여 오류가 많이 발생하여 출하이후의 오류를 수정하기 위해 막대한 비용을 감수해야 할 것이고, 시험기간이 너무 길게 지속될 경우는 신뢰성은 증가하겠지만 시험비용의 과다 소요와 제품의 출하가 늦어지게 되므로 최적의 출하시기를 결정하는 것은 매우 중요하다.
최적의 출하시기를 찾는 문제는 다양한 비용모형과 소프트웨어 성장 모형에서 연구가 진행되어지고 있다. Okumoto & Goel (1979)은 NHPP에 기초한 소프트웨어 성장 모형을 사용하여 최적 출하시기 결정문제를 다루었다. Yamada & Osaki (1984)는 계획된 소프트웨어 운반 지연에 따른 비용을 비용모형에 첨부하여 최적 출하시기에 대한 문제를 다루었다. Leung (1992)는 세 가지 서로 다른 소프트웨어 신뢰성 성장 모형에서 이 문제를 다루었다. 본 논문에서는 소프트웨어 신뢰성 성장 모형 중 많이 사용되어지고 있는 Goel-Okumoto NHPP 모형에 기초하여 제시된 Pham & Zhang (1999) 논문의 비용모형을 이용하여 최적의 출하시기를 결정하는 방법을 설명하려고 한다. 3 절에서는 Goel-Okumoto 소프트웨어 신뢰성 모형과 Pham & Zhang (1999)이 제안한 소프트웨어 비용모형에 대해 간략하게 알아보고 4 절에서 최적 출하시기를 결정하는 방법을 제시하고 5 절에서 수치 예제를 제공하였다.

2. 소프트웨어 비용 모형

2.1 Goel-Okumoto 소프트웨어 신뢰성 모형

소프트웨어 신뢰도 함수 $R(x|T)$는 테스트 시간 t ($t\geq 0, x>0$)에서 고장이 발생했다고 주어져 있을 때 시간구간 $[t, t+x)$에서 다음 고장이 발생하지 않는 확률로 정의된다. Nonhomogeneous Poisson process(NHPP) 모형에서의 신뢰도 함수는

$$R(x|T) = e^{-[m(t+x)-m(t)]}.$$

NHPP 모형에서 중요시하는 점은 어떤 시점까지 발생한 평균 고장수를 나타내는 mean value function을 적절하게 결정하는 것이다. 여기서 평균 고장수 $m(T) = E[N(t)]$로 정의되며 Goel-Okumoto NHPP 모형에서의 $m(T)$는 다음과 같다.

$$m(T) = a(1-e^{-bT}).$$

여기서, $m(T)$는 시간 T까지 감지된 오류의 평균수, a는 테스트하기 전에 소프트웨어에 존재하는 전체 오류의 평균수이며, b는 fault의 장강도이다. 따라서 사용되는 신뢰도함수는

$$R(x|T) = e^{-[m(t+x)-m(t)]} = e^{-a[e^{-bT}-e^{-b(T+x)}]}.$$

Goel-Okumoto NHPP 모형을 사용한 이유는 간단한 수식형태를 가지고 있으며 응용하기 쉽고 많은 응용에서 잘 작동하기 때문이다.

2.2 소프트웨어 비용모형

소프트웨어의 품질은 얼마나 많은 시간동안 테스트를 했는지와 무슨 테스트 방법을 사용했는지에 의존한다. 일반적으로 테스트 기간을 많이 잡으면 잡을수록 소프트웨어에 존재하는 오류를 더 많이 제거할 수 있으면 이를 통해 소프트웨어 신뢰성을 높일 수 있다. 그런데 소프트웨어의 테스트 비용 또한 증가할 것이다. 한편 만일 테스트 기간을 짧게 잡으면 소프트웨어의 비용은 감소할 수 있지만 이를 사용하는 소비자들은 신뢰성이 낮은 소프트웨어를 구입할 위험이 커지게 될 것이다. 또한 운용단계 동안의 비용이 증가하게 되는데 그 이유는 테스트 단계에서보다는 운용단계에서 오류를 제거하는데 더 많은 비용이 들기 때문이다. 이번 절에서는 Pham & Zhang (1999)논문에서 제안한 비용모델에 대해 간단히 알아보겠다.
Pham & Zhang (1999)논문에서 제시된 소프트웨어의 비용모델은 다음과 같은 부분으로 구성되어 있다.
(1) 테스트를 하기 위한 비용 $E_1(T)$은 시간 T의 멱함수(power function)으로 결정한다. 테스트 비용은 테스트 시간에 비례하며, 테스트 과정에서의 learning process을 반영한 결과이다.

$$E_1(T) = C_1 T^{\alpha}, (0<\alpha<1)$$

(2) 시간 T까지 감지된 오류를 제거하는데 걸린 평균 전체시간은

$$E[\sum_{i=1}^{N(T)} Y_i] = E[N(T)] \cdot E[Y_i] = m(T)\mu_y.$$

여기서 $N(T)$는 T까지 발생한 고장수이며 μ_y는 각각의 오류를 제거하는데 걸린 평균

시간이다. 따라서 시간 T까지 모든 오류를 제거하는게 필요한 비용은

$$E_2(T) = C_2 m(T)\mu_y.$$

(3) 보증기간 $[T, T+T_w]$동안에 감지된 모든 오류를 제거하는데 걸리는 평균 전체시간은

$$E[\sum_{i=1}^{N(T)} W_i] = E[N(T)] \cdot E[W_i]$$
$$= [m(T+Tw) - m(T)]\mu_w.$$

여기서 μ_w는 각각의 오류를 제거하는데 걸린 평균시간이다. 따라서 시간 T까지 모든 오류를 제거하는데 필요한 비용은

$$E_3(T) = C_3[m(T+T_w) - m(T)]\mu_w.$$

(4) 소프트웨어제품 출하 이후에 소프트웨어 고장으로 인한 위험 비용은

$$E_4(T) = C_4[1 - R(x \mid T)].$$

위에서 얻어진 비용들을 전부 합하면 소프트웨어 전체비용에 대한 기대값은 다음과 같이 표현할 수 있다.

$$E(T) = C_0 + C_1 T^{\alpha} + C_2 m(T)\mu_y + C3[m(T+Tw) - m(T)]\mu_w + C4[1 - R(x \mid T)].$$

3. 소프트웨어 최적 출시정책

이번 절에서는 소프트웨어의 전체운용비용의 기대값을 최소화하는 소프트웨어 출시 시간 T^*을 결정하는 방법을 새롭게 제시한다.

$$y(T) = \frac{dE(T)}{dT}$$

$$\frac{d^2E(T)}{d^2T} = e^{-bT}[u(T) - C]$$

$$u(T) = \alpha(\alpha-1)C_2 T^{(\alpha-2)}e^{bt} + ab^2C_4(1-e^{-b\alpha})R(x \mid T) \times [1 - ae^{-bT}(1-e^{-b\alpha})]$$

$$C = C_2\mu_y ab^2 - \mu_w C_3 ab^2(1-e^{-bT_w})$$

함수 $u(T)$는 시간 T에 따라 증가하다가 감소하는 형태의 함수임을 알 수 있다.

정 리 *C_0, C_1, C_2, C_3, C_4, x, μ_y, μ_w, T_w가 주어져 있다면 소프트웨어의 평균 전체운용비용을 최소화하는 T의 최적값, T^*는 다음과 같다.*

(1) 만일 $u(0) < C$,

(a) 만일 $y(0) > 0$이면, $T \in (0, T_a]$인 경우는 $y(T) > 0$고 $T \in (T_a, T_b]$인 경우는 $y(T) < 0$,

$T \in (T_b, \infty)$인 경우는 $y(T) > 0$이므로

만일 $E(0) < E(T_b)$이면 $T^* = 0$이고,

만일 $E(0) > E(T_b)$이면 $T^* = T_b$.

(b) 만일 $y(0) < 0$이면, $T \in (0, T')$인 경우는 $y(T) < 0$이고 $T \in (T', \infty)$인 경우는 $y(T) > 0$이므로 $T^* = T'$, 여기서 $T' = y^{-1}(0)$.

증명. 여기에서 함수 $u(T) = u_1(T) + u_2(T)$라고 하자.

$$u_1(T) = \alpha(\alpha-1)C_2 T^{(\alpha-2)}e^{bt}$$

$$u_2(T) = ab^2C_4(1-e^{-b\alpha})R(x \mid T) \times [1 - ae^{-bT}(1-e^{-b\alpha})].$$

함수 $u_1(T)$는 $T < (2-\alpha)/b$인 경우 $du_1(T)/dT > 0$이며 $T > (2-\alpha)/b$인 경우 $du_1(T)/dT < 0$이고 $T > 2(2-\alpha)/b$인 경우 $d^2u_1(T)/d^2T < 0$이므로 증가하다가 감소하는 형태를 보인다. 그리고 $T \to \infty$이면 $u_1(T) \to -\infty$이다. 함수 $u_2(T)$는 증가하다가 일정한 값으로 수렴함을 알 수 있다. 따라서 함수 $u(T)$는 증가하다가 감소하는 형태임을 알 수 있다. 함수 $y(T)$의 경우는 T의 값이 커짐에 따라 0으로 수렴하게 된다.

(1) 만일 $u(0) < C$이면, $T \in (0, T_1]$인 경우는 $u(T) < C$이므로 $y(T)$는 감소하고 $T \in (T_1, T_2]$인 경우는 $u(T) > C$이므로 $y(T)$는 증가, $T \in (T_2, \infty]$의 경우는 $u(T) > C$, $y(T)$는 다시 감소하게 된다. 따라서,

(a) 만일 $y(0) > 0$이면, $T \in (0, T_a]$인 경우에

$y(T)>0$이기 때문에 $E(T)$는 증가하고 $T\in(T_a, T_b]$인 경우는 $y(T)<0$으로 $E(T)$는 감소하고, $T\in(T_b,\infty)$인 경우는 $y(T)>0$이므로 $E(T)$는 증가한다. 따라서

만일 $E(0)<E(T_b)$이면 $T^*=0$이고,

만일 $E(0)>E(T_b)$이면 $T^*=T_b$.

(b) 만일 $y(0)<0$이면, $T\in(0, T'']$인 경우는 $y(T)<0$이기 때문에 $E(T)$는 감소하고 $T\in(T'',\infty)$인 경우는 $y(T)>0$이므로 $E(T)$는 증가한다. 따라서 $T^*=T''$, 여기서 $T''=y^{-1}(0)$.

4. 수치 예제

이번 절에서 사용된 데이터는 Misra (1983) 논문에서 사용한 시간당 고장수이다. 최우도 추정법을 사용하여 Goel-Okumoto 모델의 파라미터 값을 추정하면

$$\hat{a}=142.32, \qquad \hat{b}=0.1246.$$

따라서 Mean value function은 다음과 같이 얻어진다.

$$m(T)=142.32(1-e^{-0.1246T}).$$

비용계수는 $C_0=100$, $C_1=50$, $C_2=25$, $C_3=100$, $C_4=1000$로 하고 $\mu_w=0.5$, $\mu_y=0.1$, $x=0.05$, $T_w=20$, $\alpha=0.95$로 가정하였을 경우에 평균 전체운용 비용의 기대값을 최소로 하는 출시 시간 T^*은 표 1과 그림 1을 통해 나타내었다. 이 수치예제는 앞에서 제시한 최적 출시시기를 결정하는 정리 (b)에 해당되는 것이다.

표 1 : 최적 출시 시간

출시시간	평균 전체 비용, $E(T)$
23.0	1839.40
23.5	1835.37
24.0	1833.17
24.5*	1832.12
25.0	1832.32
25.5	1833.71
26.0	1836.20

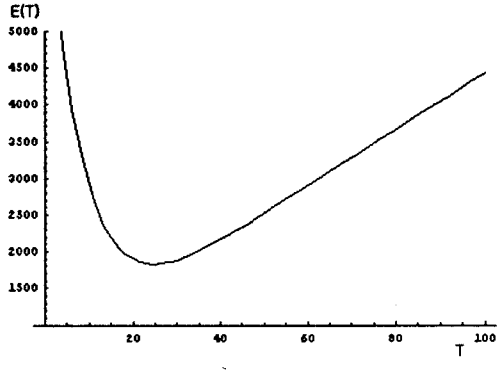

그림 1 : 시간 t 에 따른 평균 전체운용 비용

참고 문헌

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

[2] H. Pham, X. Zhang(1998), A software cost model with error removal times and risk costs, International Journal of Systems Science, 29, 435-442.

[3] H. Pham, X. Zhang(1998), A software cost model with warranty cost, error removal times and risk costs, IIE Transactions, 30, 1135-1142.

[4] H. Pham, X. Zhang(1999), A software cost model with warranty and risk costs, IEEE Transactions on Computer, 48, 71-75.

[5] P.N. Misra(1983), Software reliability analysis, IBM Systems Journal, 22, 262-270.

[6] Y.W. Leung(1992), Optimum software release time with a given cost budget, Journal of Systems Software, 17, 233-242.

[7] S. Yamada, H. Narihisa, S. Osaki(1984), Optimum release polices for a software system with a scheduled software delivery time, International Journal of Systems Science, 15, 905-914.