Journal of the Korean Institute of Industrial Engineers
Vol. 30, No. 4, pp. 261-266, December 2004.

출시 후 보수를 고려한 소프트웨어의 최적 출시시기

이진승[1] · 나일용[2] · 홍정식[3] · 이창훈[1†]

[1]서울대학교 산업공학과 / [2]한국국방과학연구소 / [3]서울산업대학교 산업정보시스템공학과

Optimal Software Release Time Considering Maintenance during Operation

Chin-Seung Lee[1] · Il-Yong Na[2] · Jung-Sik Hong[3] · Chang-Hoon Lie[1]

[1]Department of Industrial Engineering, Seoul National University, Seoul, 151-742

[2]Agency for Defense Development, Daejeon, 305-600

[3]Department of Industrial and Information Systems Engineering, Seoul National University of Technology, Seoul, 139-743

In this paper, the software reliability growth model which incorporates the periodic maintenance after the release is proposed. Using the proposed model, the debugging and periodic maintenance cost subject to the required level of the software reliability are investigated. An optimal software release time is derived for a fixed interval of periodic maintenance. To validate the proposed model, release times obtained in this study are compared with examples. The proposed investigation is expected to be served as one of factors in determining the release time of the software where periodic maintenance is considered.

***Keywords*:** software reliability growth model, software release time, maintenance

1. 서 론

컴퓨터와 전기, 전자, 정보통신의 발달로 과거 하드웨어로만 작동하던 시스템의 많은 부분들이 소프트웨어의 제어 및 관리 시스템으로 대체되고 있다. 또한 소프트웨어의 적용 분야와 범위가 급속히 확장되고 있으며, 해당 시스템에서의 중요성도 계속 증가하고 있다. 따라서 소프트웨어의 정확한 신뢰성 분석과 이를 바탕으로 한 설계 · 개발뿐 아니라 출시시기 결정에 반영하는 것이 필요하다. 사용자가 요구하는 신뢰도를 만족시키면서 소프트웨어의 수명주기(life cycle)를 고려한 출시시기 결정은 상용(commercial) 소프트웨어의 경제성 분석에 매우 중요한 사항이 된다. 그러나 기존의 출시시기 결정에 관한 연구들(Kimura *et al*., 1999; Yamada and Osaki, 1985; Yang and Xie, 2000; Xie and Hong, 1999)을 살펴보면 소프트웨어 테스트 기간 동안의 신뢰성 평가에만 초점을 맞추고 있으며, 출시 이후의 보수에 관한 연구 및 보수 정책에 대해서는 제시되고 있지 않다. 일반적으로 소프트웨어는 출시 이후에도 패치(patch), 서비스팩(service pack) 등을 통해 지속적인 보수가 이루어지기 때문에, 출시 이후의 보수를 고려하여 출시시기를 결정하는 방법론이 필요하다.

하드웨어 시스템의 경우 시간이 지남에 따라 노화(aging)현상이 일어나기 때문에 시스템의 고장률은 점차 증가하는 경향을 가지게 된다. 그러나 소프트웨어의 경우에는 개발 및 테스트 기간 동안 오류 발견(debugging) 작업이 계속적으로 이루어지기 때문에, <Figure 1>과 같이 시간이 지남에 따라 고장률은 계속해서 감소하는 경향을 가지게 된다. 이러한 소프트웨어에서 가지는 신뢰성의 특수성을 모형화한 것을 소프트웨어 신뢰성 성장 모형(Software Reliability Growth Model; SRGM)이라고 한다. 이 모형에서 출시 후 보수가 진행되지 않는다는 가정하에서는 출시 후 소프트웨어가 사용에 들어갔을 때는 상수 고장률을 가지게 된다. 즉, 특정 시점에서 테스트가 완료되고 출시된 후에는 출시 시점에서의 고장률이 시간이 지나더라도

본 연구는 2001학년도 연구년제(서울대) 기간에 수행한 연구 결과임.
† 연락저자 : 이창훈 교수, 151-742 서울시 관악구 신림동 산56-1번지 서울대학교 산업공학과, Fax : 02-873-6486, E-mail : chl@cybernet.snu.ac.kr
2004년 4월 접수; 2004년 6월 수정본 접수; 2004년 7월 게재 확정.

일정하게 유지되는 것이다. 아래 <Figure 1>에서와 같이 출시 이후에는 A의 궤적을 따라 고장률이 변하는 것이 아니라 B와 같이 일정한 고장률을 가지게 되는 것이다. 그렇기 때문에 출시 이후의 신뢰성을 분석하는 경우 A가 아닌 B의 고장률로써 분석을 해야 올바른 결과를 얻어낼 수 있다. 따라서 Yamada and Osaki(1985)가 A의 고장률 함수를 이용하여 구한 최적 출시 시점에서는 의도한 만큼의 신뢰성을 확보하지 못하는 결과를 가져오게 된다. Yang and Xie(2000)는 Yamada and Osaki(1985)가 비용 및 신뢰성을 고려한 최적 출시시기 결정에 대하여 이러한 문제점을 지적하고 있으며, 올바른 비용 및 신뢰성 분석을 하기 위해서는 소프트웨어의 고장률 함수로 테스팅 단계의 고장률 함수가 아닌 실제 사용자가 사용하는 단계의 고장률 함수를 사용해야 한다고 주장하였다. Yang and Xie(2000)의 모형(<Figure 1>의 B)을 사용하여 최적 출시시기를 결정하게 되면 Yamada and Osaki(1985)가 제시했던 모형보다 최적 출시 시점이 늦어지게 된다. 이러한 테스팅 단계의 신뢰성과 사용 단계의 신뢰성을 구분하기 위해서 출시 이전의 고장률 함수를 통해 분석되는 신뢰성을 테스팅 신뢰성(Testing Reliability)이라 하고, 출시 이후 실제 사용자가 느끼게 되는 신뢰성을 사용 단계의 신뢰성(Operational Reliability)이라고 한다(Yang and Xie, 2000).

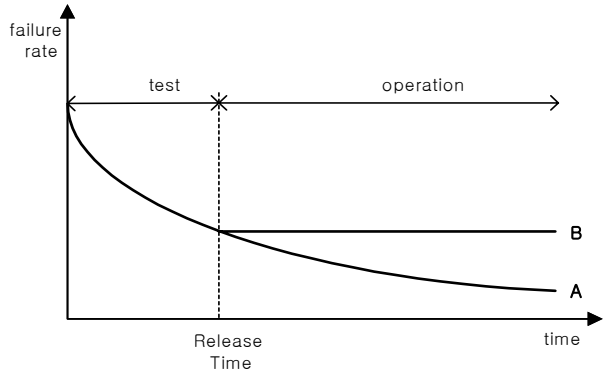

Figure 1. Failure rate of software during testing and operational phase.

그러나 현재의 많은 상용 소프트웨어들을 고려해 볼 때, 소프트웨어라고 할지라도 출시 이후 보수가 이루어지지 않으며 일정한 고장률을 가진다는 가정은 무리가 있다. 대부분의 상용 소프트웨어들이 출시 이후에도 일정한 시점까지, 이를테면 소프트웨어의 수명주기가 다 할 때까지, 서비스 팩이나 패치 등을 통해서 보수를 해주고 있다. 비단, 상용 소프트웨어가 아니라고 하더라도 정도의 차이는 있을지언정 보수가 한 번도 이루어지지 않는다고는 할 수 없다.

따라서 본 논문에서는 위에서 언급한 문제점을 해결하기 위하여 출시 이후의 고장률이 보수에 의해 변화되는 모형을 제시하였다. 출시 이후 보수로 인한 고장률의 변화가 직접적으로 비용 및 신뢰성에 영향을 주기 때문에 좀더 정확한 출시시기 결정이 가능해진다. 그리고 제시된 모형의 신뢰성 함수와 비용 함수를 통해 최적 출시시기를 결정함으로써, 소프트웨어 출시에 관한 보다 현실적인 정책 제시를 할 수 있게 된다.

2. 보수를 고려한 소프트웨어 신뢰성 성장 모형

기호 정의

$C(t)$: 비용 함수
$R(t)$: 신뢰성 함수
$m(t)$: 소프트웨어 고장 횟수의 평균값 함수
$\lambda(t)$: 소프트웨어의 고장률 함수
T_R : 소프트웨어 출시시기
R_0 : 최소 신뢰성 기준
M : 출시 후 보수 횟수
W : 소프트웨어 수명주기(소프트웨어 신뢰성 보증 기간)
c_1 : 소프트웨어의 오류 하나당 수정 비용
c_2 : 소프트웨어의 1회 보수비용
c_3 : 소프트웨어 출시 후 1회 고장으로 인한 손실 비용
c_4 : 소프트웨어 출시 전 단위 시간당 테스트 비용
c_5 : 소프트웨어 출시 후 단위 시간당 테스트 비용

소프트웨어 신뢰성 성장 모형으로 가장 많이 사용되고 있는 Goel and Okumoto 모형(이하 G-O 모형으로 표기)은 소프트웨어의 고장(failure)이 남아있는 오류(error)의 개수에 비례해서 발생한다(식 (2) 참조)는 기본 가정 하에 만들어진 비균일 포아송 과정(nonhomogenous Poisson process; NHPP) 모형이며, t 시점까지의 고장 횟수를 $N(t)$, 고장 횟수의 평균값 함수를 $m(t)$라고 하면 아래 식과 같다(Goel and Okumoto, 1979).

$$\Pr(N(t)=n)=\frac{(m(t))^n}{n!}\exp(-m(t)),\ n=0,1,2,\cdots \tag{1}$$

평균값 함수 $m(t)$와 고장률 함수 $\lambda(t)$는 식 (3)과 (4)로 표현된다. 여기서 a는 소프트웨어가 테스트를 시작할 때 가지고 있는 오류의 평균 개수, b는 남아 있는 오류 하나당 고장을 유발시킬 가능성을 나타내는 비례상수이다.

$$\frac{dm(t)}{dt}=b(a-m(t)),\ \ m(0)=0 \tag{2}$$

$$m(t)=a(1-e^{-bt}) \tag{3}$$

$$\lambda(t)=\frac{dm(t)}{dt}=abe^{-bt} \tag{4}$$

본 논문에서 제시되는 보수를 고려한 신뢰성 성장 모형은 G-O 모형의 가정에 다음의 세 가지 가정을 더하여 만들어진다.

가정

① 소프트웨어는 출시 이후 일정 시간 W동안 품질(신뢰성)을 보증한다.

② 출시 후 M번의 보수(패치) 작업을 시행한다. 즉, 일정 시간 간격 $L=\left(\frac{W}{M+1}\right)$을 두고 보수(패치) 작업을 한다.

③ 출시 후에도 소프트웨어 개발자는 오류 발견(debugging) 작업을 계속 진행하며, 보수 후 소프트웨어 고장률의 감소량은 G-O 모형을 따른다.

출시 후 보수를 고려한 소프트웨어 신뢰성 성장 모형의 고장률 함수는 <Figure 2>와 같다. 주기 L마다 보수가 이루어지는 것을 가정하고 있기 때문에 소프트웨어 출시 후에는 보수 주기 L마다 고장률이 감소하는 계단 함수의 모양을 가지게 된다. 즉, 소프트웨어 출시시기 T_R 이전의 테스트 기간 동안 고장률 함수를 $\lambda_0(t)$로, 출시 후 $i-1$번째 보수 후부터 i번째 보수가 시행되기 직전까지의 고장률 함수를 $\lambda_i(t)(i=1, \cdots, M+1)$로 정의하면, 소프트웨어의 고장률 함수는 아래와 같이 주어진다.

$$\lambda_0(t)=abe^{-bt},\ 0\le t\le T_R \tag{5}$$

$$\lambda_i(t)=abe^{-b(T_R+(i-1)L)},\quad T_R+(i-1)L<t\le T_R+iL,\ i=1,\cdots,M+1 \tag{6}$$

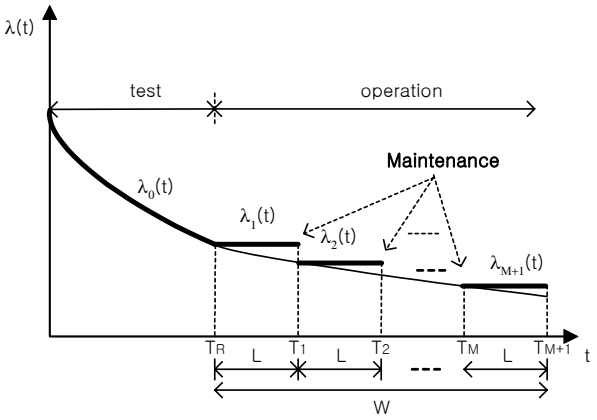

Figure 2. Failure rate function of software considering maintenance.

소프트웨어 신뢰성이란 '특정 시점에서부터 일정 시간 동안 소프트웨어의 고장이 발생하지 않을 확률'을 의미한다(Musa *et al.*, 1987). 즉, T시점 이후 x시간 동안의 소프트웨어 신뢰성은 식 (7)과 같이 표현된다.

$$R(x|T)=\exp(-[m(T+x)-m(T)]) \tag{7}$$

그런데 앞 식에서 $m(T+x)-m(T)$는 고장률 $\lambda(t)$의 그래프에서 $[T, T+x]$ 사이의 적분값이 된다. 즉, $\lambda(t)$의 $[T, T+x]$ 사이의 넓이를 S라 할 때 신뢰성은 아래와 같이 나타낼 수 있다(Yang and Xie, 2000).

$$R(x|T)=\exp(-S) \tag{8}$$

따라서 출시 후 보수를 고려한 보증 기간(W) 동안의 소프트웨어의 신뢰성은 식 (6)-(8)을 이용하면 다음과 같이 얻어진다.

$$R(W|T_R)=\exp\left[-\sum_{i=1}^{M+1}\left(\int_{T_R+(i-1)L}^{T_R+iL}\lambda_i(t)\,dt\right)\right]$$

$$=\exp[-L\cdot(abe^{-bT_R})\cdot(\frac{1-e^{-bW}}{1-e^{-bL}})] \tag{9}$$

3. 보수를 고려한 소프트웨어 최적 출시시기의 결정

비용 및 신뢰성을 고려한 소프트웨어의 최적 출시시기는 아래의 식 (10)을 만족시키는 T_R이 된다. 즉, 출시 후 품질 보증 기간 동안 사양에서 요구하는 최소한의 신뢰도를 만족시키면서 관련 비용을 최소화하는 T_R이 소프트웨어의 최적 출시시기가 된다.

$$\min\ C(T_R) \tag{10}$$

$$s.t.\ R(W|T_R)\ge R_0$$

먼저 식 (10)의 비용 함수를 도출해 보자. 소프트웨어의 테스트 및 오류 수정, 출시 후 보수 및 출시 후 고장으로 인한 손실에 관련된 비용은 아래와 같이 구성된다.

비용 = (T_R+W까지의 오류 수정 비용) + (출시 후 보수(패치) 비용) + (출시 후 발생하는 고장으로 인한 손실 비용) + (출시 전 테스트 비용) + (출시 후 테스트 비용)

비용 함수의 첫째 항인 T_R+W까지의 오류 수정 비용은 소프트웨어의 개발자가 테스트 기간이나 출시 후 보증 기간에서 오류를 발견해 이를 수정하는 데 필요한 비용이며, 가정 ③에 의하여 오류를 찾아낼 발견율(error detection rate)은 G-O 모형을 따르게 된다. 셋째 항의 출시 후 발생하는 고장으로 인한 손실 비용은 소프트웨어가 출시 후 실제 사용을 하다가 고장이 발생함으로써 야기되는 비용이며, 이 때 사용 단계에서의 고장률은 <Figure 2>의 계단 모양의 함수로 주어진다. 또한 넷째 및 다섯째 항인 테스트 비용의 경우에는 오류 발견율이 G-O 모형을 따른다고 가정을 하고 있기 때문에 출시 후에는 사용자의 증가로 인하여 테스트 비용을 적게 투입하더라도 출시 전과 동일한 효과를 가져오게 되므로 테스트 비용을 출시 전후로 구분하였다. 이에 근거하여 비용 함수를 도출하면 다음과 같다.

$$C(T_R) = c_1 a(1-e^{-b(T_R+W)}) + c_2 M + c_3 \sum_{i=1}^{M+1} Labe^{-b(T_R+(i-1)L)} + c_4 T_R + c_5 W \quad (11)$$

다음으로 비용 함수 $C(T_R)$의 특성에 대해 알아보자. 비용함수(11)의 T_R에 관한 일차 미분식 및 이차 미분식은 아래와 같이 얻어진다.

$$\frac{dC(T_R)}{dT_R} = c_1 abe^{-bT_R} \times \left[e^{-bW} - \frac{c_3 bL}{c_1}\left(\frac{1-e^{-bW}}{1-e^{-bL}}\right)\right] + c_4 \quad (12)$$

$$\frac{d^2C(T_R)}{dT_R^2} = c_1 ab^2 e^{-bT_R} \times \left[-e^{-bW} + \frac{c_3 bL}{c_1}\left(\frac{1-e^{-bW}}{1-e^{-bL}}\right)\right] \quad (13)$$

식 (13)에서 $c_1 ab^2 e^{-bT_R} > 0$이므로, 식 (13)의 [ ] 안의 수식에만 주목해 보자. [ ]안의 수식을 W에 대한 함수 $f(W)$라고 하면 아래와 같은 정리가 성립한다.

(정리 1)

함수 $f(W) = -e^{-bW} + \frac{c_3 bL}{c_1}\left(\frac{1-e^{-bW}}{1-e^{-bL}}\right)$는 단조 증가 함수이다.

(증명) $\frac{df(W)}{dW} = e^{-bW}\{b + \frac{c_3 b^2 L}{c_1(1-e^{-bL})}\} > 0$ 이 성립한다. 따라서 $f(W)$는 단조 증가 함수이다.

W'이 $f(W)=0$을 만족시키는 값이라 하면, (정리 1)에 의해 다음과 같은 결론을 얻을 수 있다.

- $W > W'$인 경우, $\frac{d^2C(T_R)}{dT_R^2} > 0$ 가 성립하므로 $C(T_R)$은 convex 함수이며, $\frac{dC}{dT_R} = 0$가 되는 $T_R = T_R^1$에서 최소값을 갖는다.
- $W < W'$인 경우, $\frac{d^2C(T_R)}{dT_R^2} < 0$ 가 성립하므로 $C(T_R)$은 concave 함수이고, $\frac{dC}{dT_R} > 0$가 성립하므로 $T_R = 0$에서 최소값을 갖는다.

식 (9)에서 주어진 신뢰성 함수의 특성을 살펴보면 신뢰성 함수는 출시시기 T_R에 대한 단조 증가 함수이다(증명 생략).

위에서 알아본 비용함수 및 신뢰성 함수의 특성을 고려하여 최적 소프트웨어 출시시기를 유도해 보자. 비용 함수를 최소로 만드는 출시시기를 T_R^1, 신뢰성 하한을 만족시키는 출시시기를 T_R^2라 할 때, $T_R^1 > T_R^2$이 성립한다면 T_R^2 시점 이후에는 소프트웨어 사양에서 설정한 신뢰성이 만족되기 때문에 비용을 최소화하는 T_R^1에서 출시를 하는 것이 최적이 된다. 반대로 $T_R^1 < T_R^2$라면 비용이 최소인 T_R^1 시점에서 신뢰성 하한을 만족시키지 못하기 때문에 T_R^1 시점이 아닌 T_R^2 시점까지 기다려야 출시가 가능해 진다. 따라서 최적 출시시기 T_R^*은 아래와 같이 결정된다.

$$T_R^* = \max(T_R^1, T_R^2) \quad (14)$$

여기서 T_R^1 및 T_R^2는 식 (9)-(13)을 통해서 아래와 같이 얻어진다.

$$T_R^1 = \begin{cases} 0, & W < W' \\ -\frac{1}{b}\ln\left[\frac{c_4}{c_3 Lab^2\left(\frac{1-e^{-bW}}{1-e^{-bL}}\right) - c_1 abe^{-bW}}\right], & W < W' \end{cases} \quad (15)$$

$$T_R^2 = -\frac{1}{b}\ln\left[\frac{(e^{-bL}-1)\ln R_0}{abL(1-e^{-bW})}\right] \quad (16)$$

4. 실험 예제

실험 예제에 사용된 데이터는 Goel and Okumoto(1979), Yang and Xie(2000) 등에서 사용된 소프트웨어 고장 자료를 이용하였다. 이 자료는 소프트웨어의 테스트 기간 동안 얻어진 26개의 고장 시간 간 자료로 구성되어 있으며, 이 고장 자료에 대한 G-O 모형의 모수인 a, b의 최대 우도 추정치는 다음과 같다.

$$\hat{a} = 34, \quad \hat{b} = 0.00579$$

비용 함수에서 사용된 비용과 관련된 모수는 기존의 연구와 비교를 하기 위해 Yang and Xie(2000)에서 제시된 것을 사용하였고, 새롭게 도입된 c_2의 경우에는 출시 후 소프트웨어의 고장으로 인한 손실 비용보다는 작게 설정을 하고, 출시 후 단위 시간당 테스트 비용인 c_5의 경우에는 소프트웨어 출시로 인한 사용자의 증가가 테스트 비용을 감소시키는 역할을 하게 되므로 c_4보다는 작게 설정을 하였다.

$c_1 = 200$, $c_2 = 400$, $c_3 = 1500$, $c_4 = 10$, $c_5 = 1$, $W = 400$, $R_0 = 0.7$

출시 후 보수 횟수(M)가 5회일 때 위의 모수가 주어진 소프트웨어의 출시시기를 결정해 보았다. 이 예제에서 식 (15)에 나타나 있는 W'은 21.4가 되고 $W > W'$이 성립하므로, 비용 함수는 convex 함수가 된다. <Figure 3>의 굵은 실선이 본 논문에서 제시된 모형의 비용 함수와 신뢰성 함수이다. 비용을 최소화하는 T_R^1은 597이 되고, 신뢰성 하한 0.7을 만족시키는 T_R^2는 803이 된다. 따라서 최적 출시시기 T_R^*는 803이 된다. 또한, 이 때의 비용은 17747이 소요된다. 반면에, Yang and Xie(2000)의 논문에서 제시된 출시 이후 보수를 고려하지 않는 모형($M=0,\ c_5=0$)을 사용할 경우는 <Figure 3>에서 점선으로 나타난 부분이며, 비용 측면에서는 $T_R^1=729$, 신뢰성 측면에서는 $T_R^2=933$이 되어 최적 출시시기는 933이 된다. 그리고 이 때의 비용은 17052로 나타난다.

두 결과를 비교해 보면 출시 후 보수를 고려하는 경우에는 출시 시점은 130을 앞당길 수 있지만 비용은 695가 증가하는 것을 알 수 있다. 앞의 실험 예제에서는 1회 보수비용인 c_2를 400으로 설정하여 비용이 증가하는 결과를 가져오지만 이 비용이 400보다 더 작게 설정하는 경우에는 비용도 감소하는 결과를 가져올 수 있게 된다. 또한 본 논문에서의 비용 함수는 출시시기 단축에 따른 이익을 고려하고 있지 않지만, 이를 고려하는 경우에도 비용 감소 결과를 가져올 수 있게 된다. 즉, 동일한 신뢰성 조건을 만족시키는 조건하에서 출시시기를 앞당김으로써 시장 선점의 효과와 매출액의 증가를 고려하게 된다면 좀더 정확한 비용 비교를 할 수 있을 것으로 판단된다.

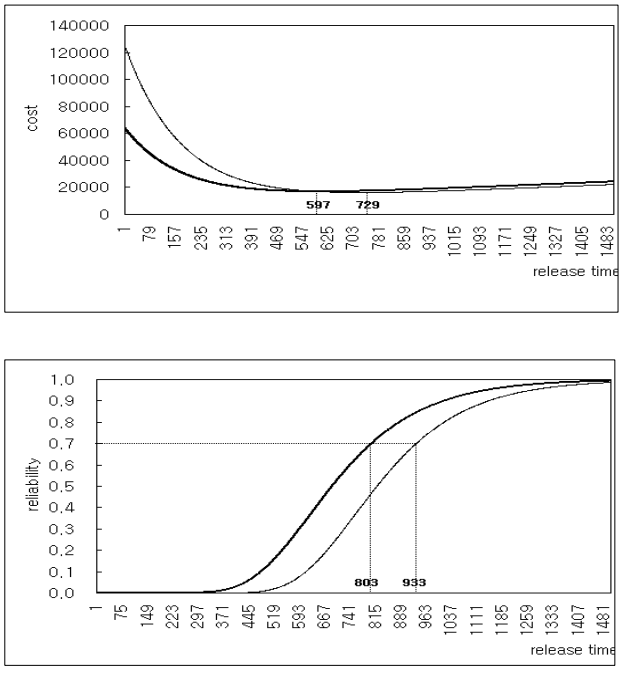

Figure 3. The optimal software release time with cost and reliability requirement.

<Table 1>에서는 소프트웨어의 보증 기간(W) 및 최소 요구 신뢰성(R_0)의 변화에 따른 최적 출시시기와 비용의 변화를 보여주고 있다. 보증 기간 및 요구 신뢰성이 증가할수록 최적 출시시기는 늦어지고 비용은 증가함을 알 수가 있다.

Table 1. Optimal release time and cost for W & R_0

W \ R_0	0.6		0.7		0.8		0.9	
	T_R^*	Cost	T_R^*	Cost	T_R^*	Cost	T_R^*	Cost
100	593	15071	655	15497.1	736	16138.9	866	17290.4
300	721	16654.5	783	17049.3	846	17664.2	994	18791.7
500	763	17288.7	825	17679.3	906	18290.4	1036	19414.6
700	788	17739.8	850	18130	931	18740.9	1060	19855.7

<Figure 4>는 보수 횟수(M)의 변화에 따른 최적 출시시기의 변화 추이를 나타낸다. 출시 후 보수 횟수를 크게 설정할수록 최적 출시시기를 앞당길 수 있게 된다. 그러나 어느 정도의 보수 횟수 이상에서는 출시시기 감소 효과가 미미해짐을 알 수가 있다. 예를 들어, 보증 기간이 200인 경우에는 보수 횟수가 9인 경우에 최적 출시시기는 733이며, 보수 횟수가 30인 경우에도 최적 출시시기는 726이 된다. 즉, 보수 횟수가 9회 이상인 경우에는 보수 횟수를 더 크게 잡아도 출시시기는 거의 변화하지 않음을 알 수가 있다.

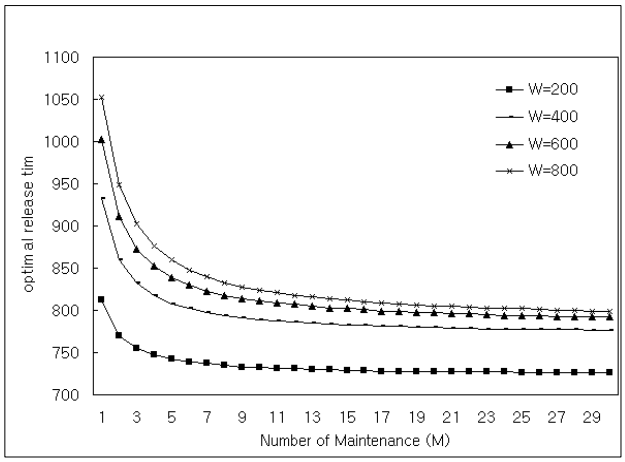

Figure 4. The optimal software release time versus number of maintenance.

다음으로 a 값의 변화에 따른 최적 출시시기와 비용의 변화에 대해 조사해 보았다. 소프트웨어가 테스트를 시작할 때 가지고 있는 오류의 평균 개수로 정의되는 a 값은 출시 시점 이전까지 얻은 테스트 데이터로부터 추정된다. 실험 조건은 a의 값을 제외하고는 앞의 실험 조건과 동일하고 a값만 5, 10, 20, 40, 80, 160으로 변화시켜 보았다. 이에 따른 결과는 <Table 2>와 같다.

Table 2. a Optimal release time and cost for a

a	5	10	20	40	80	160
T_R^1	267	386	505	625	744	865
T_R^2	472	592	711	831	951	1070
T_R^*	472	592	711	831	951	1070
Cost	8637	10836	14028	19227	28426	45618

<Table 2>를 보면 a값이 두 배 증가할 때마다 일정 시간씩 최적 출시시기는 늦어지게 된다. 출시시기 면에서는 $a=5$에서 $a=160$까지 32배 증가했음에도 불구하고 출시시기는 두 배 정도가 증가하였다. 하지만, 비용 측면에서는 5배 이상 많은 비용이 증가하였다. 따라서 초기 오류의 평균 개수는 출시시기를 증가시키는 것보다는 비용의 증가에 더 많은 영향을 준다고 할 수 있다.

다음으로 b값의 변화에 따른 최적 출시 시점과 비용의 변화에 대해 알아보았다. b값은 소프트웨어에 존재하고 있는 오류 하나가 고장을 발생시킬 가능성을 나타내는 비례 상수이다. a값과 마찬가지로 출시 시점 이전에 얻은 테스트 데이터로부터 추정된다. b값을 0.002, 0.004, 0.008, 0.016, 0.032, 0.064로 변화시키면서 비용 및 신뢰성 추이를 조사해 보았다.

Table 3. Optimal release time and cost for b

b	0.002	0.004	0.008	0.016	0.032	0.064
T_R^1	841	723	490	306	187	114
T_R^2	2015	1117	598	317	172	96
T_R^*	2015	1117	598	317	187	114
Cost	29818	23249	19298	12885	11379	10488

<Table 3>을 보면 b값이 증가함에 따라 최적 출시시기 및 비용 함수의 최소값이 감소하고 있다. 하지만 점차 감소 폭이 작아지는 것을 알 수 있다. 또한, <Table 3>에서 b의 변화에 따른 최적 출시 시점 T_R^*의 변화를 보면 b의 값이 작은 경우에는 신뢰성 하한에 의해서 최적 출시 시점이 결정되지만 b의 값이 커지면 비용 함수가 최소가 되는 시점에 의해 최적 출시 시점이 결정된다. a값의 경우에는 항상 신뢰성 사양에 의해 최적 출시 시점이 결정되던 것과는 다른 양상을 보이고 있다.

5. 결론 및 추후 연구 과제

본 논문에서는 기존에 소프트웨어 출시시기를 결정하는 문제에 있어서 고려되지 않았던 출시 이후의 보수라는 개념을 도입하여 최적 출시시기를 결정하여 보았다. 이를 위해, 출시 이후의 보수 정책을 반영한 소프트웨어 신뢰성 성장 모형(software reliability growth model)을 도출하고, 이 모형을 기반으로 하여 비용 및 신뢰성에 근거한 소프트웨어 최적 출시시기를 결정해 보았다. 제시된 모형의 타당성을 검증하기 위해 실험 예제를 통해 기존의 논문에서 제시되었던 결과들과 비교분석을 수행해 보았다. 본 연구의 결과는 기존 연구에서 고려되지 않았던 보수 정책을 고려함으로써, 보다 현실에 가까운 모형을 제시하였다는 점에서 의의를 찾을 수 있고, 출시 후 보수 시점, 보수 정책 등의 결정에도 기여할 수 있을 것으로 판단된다.

본 논문에서는 출시 이후 보수 주기 동안의 오류 발견율 변화는 출시 전 테스트 환경 내에서와 동일한 것으로 가정을 하고 분석을 하였다. 하지만 출시가 된 이후에는 소프트웨어의 사용 환경이 달라지고, 테스트할 때보다는 훨씬 많은 사용자에게 노출되기 때문에 고장의 발생 및 감소 패턴에 변화가 있게 될 것이다. 따라서 추후 연구 과제로는 출시 이후 환경 변화를 고려한 모형화를 생각해 볼 수 있다.

참고문헌

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.

Kimura, M., Toyata, T. and Yamada, S. (1999), Economic analysis of software release problems with warranty cost and reliability requirement, Reliability Engineering and System Safety, **66**, 49-55.

Musa, J. D., Iannino, A. and Okumoto, K. (1987), *Software reliability: measurement, prediction, application*, McGraw-Hill, New York.

Ross, S. M. (1997), *Introduction to Probability Models*, Academic Press, San Diego.

Yamada, S. and Osaki, S. (1985), Cost-Reliability optimal release policies for software systems, *IEEE Transactions on Reliability*, **34**, 422-424.

Yang, B. and Xie, M. (2000), A study of operational and testing reliability in software reliability analysis, *Reliability Engineering and System Safety*, **70**, 323-329.

Xie, M. and Hong, G. Y. (1999), Software release time determination based on unbounded NHPP model, *Computer & Industrial Engineering*, **37**, 165-168.