Journal of the Korean Institute of Industrial Engineers
Vol. 33, No. 4, pp. 487-493, December 2007.

사용단계에서 주기적 서비스 팩 배포와 불확실한 패치 배포를 고려한 소프트웨어의 최적 출시시기

박일광† · 공명복

울산대학교 산업정보경영공학부

Optimal Release Time for Software Considering Distribution of Periodic Service Packs and Uncertain Patches during Operational Phase

Il Gwang Park · Myung Bock Kong

Department of Industrial Engineering, University of Ulsan, Ulsan 680-749, Korea

In this paper, we deal with an optimal software-release problem of determining the time to stop testing and release the software system to the user. The optimal release time problem is considered from maintenance like the periodic distribution of service packs and the unpredictable distribution of patches after the release. Moreover, the environment of software error-detection during operation differs from the environment during testing. This paper proposes the software reliability growth model which incorporates periodic service packs, unpredictable patches and operational environment. Based on the proposed model, we derive optimal release time to minimize total cost composed of fixing an error, testing and maintenance. Using numerical examples, optimal release time is determined and illustrated.

Keywords: Software Reliability Growth, Periodic Service Packs, Uncertain Patches, Accelerated Life Testing, Release Time for Software

1. 서 론

소프트웨어 시스템 개발과정에서 개발 책임자는 어느 한 시점에서 시스템 테스팅을 멈추고 시스템의 출시를 결정해야 한다. 이러한 결정 문제를 소프트웨어의 최적 출시시기(Optimal release time for software)문제라 부르며 이전 연구로는 Okumoto and Goel(1980), Koch and Kubat(1983), Yamada and Osaki (1985), Ohtera and Yamada(1990), Lee *et al*.(2004), Huang and Lyu(2005) 등이 있다. 특히 이들 중 Okumoto and Goel(1980)와 Koch and Kubat(1983) 그리고 Yamada and Osaki(1985)는 소프트웨어 시스템을 출시하기 전 테스트단계(testing phase)와 출시 후 사용단계(operational phase)에서의 오류(error) 발견 상황이 동일하다는 가정 하에 최적 출시시기를 결정하였다. 그러나 일반적으로 두 단계의 상황이 같지 않으므로, Ohtera and Yamada(1990)와 Huang and Lyu(2005)은 두 단계를 자원(resource) 소비의 크기에 따라 구분하였다. 즉 테스트단계에서 소비되는 자원(CPU 작동시간, 인력, 테스팅 케이스)이 사용단계에서 소비되는 자원(CPU 작동시간)보다 더 많다는 가정 하에 최적 출시시기를 결정하였다. 또한 사용단계의 신뢰성 평가 방법의 연구에서도 사용 환경을 고려한 연구가 진행되었는데 Yamada *et al*.(1989)와 Yamada(1993)는 테스트단계에서 얻은 통계 데이터에 의한 신뢰성 평가 방법을 연구하였고, Okamura *et al*.(2001)는 사용단계 내의 환경을 고려한 하드웨어의 가속수명시험(accelerated life testing) 개념을 적용한 평가 방법을 연구하였다.

한편 이전 연구들에서는 다루지 않았던 사용단계 내의 보수(maintenance) 문제를 Lee *et al*.(2004)는 서비스 팩(service packs), 패치(patches) 형태의 보수를 고려한 최적 출시시기를 결정하

† 연락저자 : 박일광, 680-749 울산광역시 남구 무거2동 산29 울산대학교 산업정보경영공학부, Tel : 052-259-2173, Fax : 052-259-2180
E-mail : idapark@mail.ulsan.ac.kr
2007년 07월 접수; 2007년 09월 수정본 접수; 2007년 10월 게재 확정.

였다. 일반적으로 고장률(failure rate)변화가 직접적으로 비용 및 신뢰성에 영향을 주기 때문에 이들은 출시 후 보수를 고려한 고장률 변화 모형을 제시하였다. 즉 사용단계 내의 고장률 함수의 변화가 테스트단계 내와 동일한 것으로 가정함으로써, 고장률 감소정도가 Goel and Okumoto(1979)의 고장률 함수(<Figure 1>의 $\alpha=1$)를 따른다는 가정하에, 사용단계에서의 보수정책을 고려한 최적 출시시기를 결정하였다.

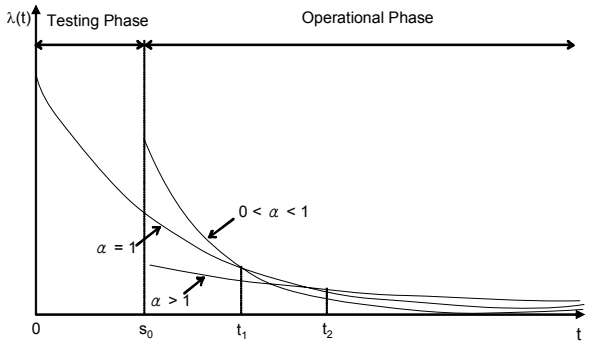

Figure 1. Failure rate function from environmental coefficient(α)

그러나 앞서 언급하였듯이 두 단계에서의 오류 발견 환경이 동일하지 않기 때문에 고장률 함수의 변화도 달라야 한다. 따라서 사용단계의 환경을 나타낼 수 있는 환경계수(environmental coefficient) α를 사용하면 <Figure 1>에서처럼 다양한 형태의 고장률 함수가 나타난다. 여기서 α는 Okamura *et al.*(2001)가 사용단계에서의 사용 환경을 나타내기 위해서 하드웨어 가속수명시험의 가속계수(accelerated factor)와 유사한 개념으로 사용한 바가 있다. α에 따른 사용단계에서의 고장률 함수를 살펴보면 <Figure 1>에서 $\alpha>1$인 경우, 구간$(s,\ t_2]$사이에서는 $\alpha=1$(사용단계 내의 환경이 테스트단계 내의 환경과 동일)인 경우의 고장률 함수보다 더 작고, 구간$(t_2,\ t)$사이에서는 더 큼을 알 수 있다. 반면 $0<\alpha<1$인 경우, 구간$(s,\ t_1]$사이에서 $\alpha=1$인 경우의 고장률 함수보다 더 크고, 구간$(t_1,\ t)$사이에서 더 작음을 알 수 있다. 여기서 우리는 직관적으로 사용단계에서는 $0<\alpha<1$인 경우의 고장률 감소 형태가 될 것으로 판단할 수 있다. 즉 시스템 출시 후 많은 사용자들에게 노출되기 때문에 어느 시점(가령 <Figure 1>에서 t_1)까지는 고장률이 클 것이고 $t\to\infty$일 때 α가 가지는 값에 상관없이 최종적으로 발견될 전체 평균 고장수가 a라고 할 때(식 (3) 참조) 그 시점을 지나면 작아지는 형태가 될 것이다. 이 때 고장률 함수를 $\lambda_o(t)$로 정의하면 다음과 같다.

$$\lambda_o(t)=\frac{ab}{\alpha}e^{-b(s+\frac{t-s}{\alpha})},\quad t\geq s \tag{1}$$

또한 Lee *et al.*(2004)는 보수를 서비스 팩과 패치 형태로 예를 들면서, 이들을 구분하지 않았다. 그러나 Scott(2007)이 "Why service packs are better than patches"에서 언급하였듯이 일반적으로 두 형태가 구분된다. 첫 번째 형태인 서비스 팩은 전략적 접근으로써 출시 전 철저한 계획 하에 배포하게 되고, 두 번째 형태인 패치는 전술적 접근으로써 서비스 팩을 배포하기 전 심각한 문제가 야기될 것으로 판단되는 고장들, 가령 보안과 관련된 고장들에 대해서 지체 없이 실시하게 된다. 그러므로 서비스 팩은 정해진 주기에 따라 배포가 가능하지만, 패치 배포의 주기는 예측이 불가능하다.

본 논문은 테스트단계 내의 환경과 구분되는 사용단계 내의 환경, 그리고 주기적 서비스 팩과 불확실한 패치를 고려한 소프트웨어 신뢰성 성장 모형을 도출하고, 이 모형을 기반으로 출시시기를 결정함으로써 좀 더 정확한 최적 출시시기 결정이 가능하도록 연구하였다. 또한 기존 논문의 모형을 일반화하였다는 점에서 본 논문의 의의를 찾을 수 있을 것으로 생각된다. 아울러 본 논문의 모형은 소프트웨어의 초기 개발에서 출시와 관련된 전략을 모색하는데 도움이 될 것으로 생각한다.

본 논문의 제 2장에서는 기존에서 다룬 보수를 고려한 소프트웨어 신뢰성 성장 모형을 검토한 후, 본 논문에서 제시하는 사용단계에서의 보수정책을 고려한 소프트웨어 신뢰성 성장 모형을 제시한다. 제 3장에서는 제 2장에서 제시한 모형을 기반으로 테스트단계 내에서의 비용(오류수정비용, 테스트비용)과 사용단계 내에서의 비용(오류수정비용, 서비스 팩 배포 비용, 패치비용, 손실비용)을 더한 총비용을 최소로 하는 최적 출시시기 결정 문제를 다루고, 제 4장에서는 수치예제를 통해 제 3절에서 제시한 최적 출시시기 결정 모형을 설명한다.

2. 소프트웨어 신뢰성 성장 모형

기호 정의

$N(t)$	: 시간 t까지 발견된 누적 고장수
$m(t)$	: 테스트단계에서의 평균값 함수, $E[N(t)]$
$m_o(t)$	: 소프트웨어 시스템 출시 후 사용단계에서의 평균값 함수
$\lambda(t)$	: 테스트단계에서의 고장률 함수, $m'(t)$
$\lambda_o(t)$	: 소프트웨어 시스템 출시 후 사용단계에서의 고장률 함수
$\lambda_{pi}(t)$	: 소프트웨어 시스템 출시 후 서비스 팩, 패치 배포를 고려한 고장률 함수
a	: 소프트웨어 시스템에서 최종적으로 발견될 고장수
b	: 오류 하나당 오류 발견율
α	: 사용단계에서 사용 환경을 나타내는 환경계수 (environmental coefficient)
β_i	: i번째와 $i+1$번째 서비스 팩 배포 사이에서의 오류 하나당 패치율
s	: 소프트웨어 출시시기 또는 시스템 테스트 시간
s_0	: s를 만족하는 유일한 해, $dC(s)/ds=0$
W	: 소프트웨어 시스템 출시 후 품질 보증기간
T	: 소프트웨어 수명주기, $T=s+W$

M : 서비스 팩 배포 횟수
L : 서비스 팩 배포 주기, $L = W/(M+1)$

본 논문에서 제시한 소프트웨어 신뢰성 성장 모형은 Okumoto and Goel(1980)과 Lee *et al.*(2004) 모형의 가정과 아래 가정 하에 만들어진다. 단 Lee *et al.*(2004) 모형에서의 가정 ③을 수정하여 출시 후에 소프트웨어 개발자는 오류 발견 작업을 계속 진행하지 않으며, 서비스 팩 배포와 패치 배포 후의 고장률 감소 정도는 <Figure 1>의 $0 < \alpha < 1$인 경우의 고장률 함수를 따르는 것으로 가정한다.

가정

① 사용단계에서 발생한 오류는 서비스 팩 또는 패치 배포 시점에서만 제거한다.
② 사용단계에서 발생한 오류 중 심각한 문제를 야기시킬 오류에 대하여 패치를 시행한다.
③ 서비스 팩과 패치 배포 시점에서 고장률 감소 정도는 테스트단계에서 보다 더 가혹한 환경이다. 단 패치 이후 고장률 감소 정도는 고장 하나당 패치율인 β_i가 고려된다.

소프트웨어 시스템 출시 이후 사용단계에서 서비스 팩과 패치 형태의 보수를 고려하였을 때, Lee *et al.*(2004)는 사용단계에서 오류 발견율 변화 형태로 <Figure 2>을 제시하였다. 즉 보증기간 W동안 보수 주기 L마다 고장률이 감소하는 계단 함수 모양이 된다는 것이다. 그러나 이들은 테스트단계와 사용단계 내에서의 환경이 동일하다는 가정 하에 사용단계에서 고장률 감소 정도가 Goel and Okumoto(1979)의 고장률 함수를 따른다고 하였다(<Figure 2>의 점선).

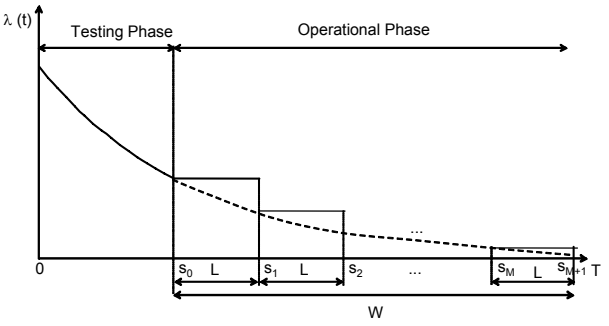

Figure 2. Failure rate function considering maintenance

소프트웨어의 최적 출시시기 결정 문제에서 기존의 많은 연구들(Okumoto and Goel 1980; Yamada and Osaki 1985; Ohtera and Yamada 1990; Lee. *et al* 2004; Huang and Lyu 2005 등)은 Goel and Okumoto(1979)의 소프트웨어 신뢰성 성장 모형을 기반으로 최적 출시시기를 결정하였다. 이 모형은 비균일 포아송 과정(nonhomogeneous Possion process; NHPP) 모형으로, t 시점까지의 고장 횟수를 $N(t)$, 고장 횟수의 평균값 함수를 $m(t)$라고 정의하면 아래 식과 같다.

$$P\{N(t) = y\} = \frac{m(t)^y e^{-m(t)}}{y!},\ y = 0, 1, 2, \cdots \tag{2}$$

이들은 평균값 함수 $m(t)$를 식 (3)과 같이 기술하였고, 식 (3)을 미분한 고장률 함수 $\lambda(t)$는 식 (4)와 같다.

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

$$\lambda(t) \equiv m'(t) = abe^{-bt} \tag{4}$$

위 식에서 a는 소프트웨어 시스템에서 최종적으로 발견될 평균 고장수이고, b는 오류 하나당 오류 발견율을 나타내는 비례상수이다.

일반적으로 테스트단계 내와 사용단계 내에서의 오류 발견 상황이 동일하지 않기 때문에 Lee *et al.*(2004)의 가정과는 달리 두 단계 내에서 각각 다른 고장률 변화 형태가 될 것이다. 따라서 테스트단계 내에서는 식 (4)의 고장률 함수를 따른다고 하고, 사용단계 내에서는 식 (1)을 따른다고 하였을 때, 보수를 고려한 고장률 감소 형태는 <Figure 3>과 같게 될 것이다. 여기서 사용 환경을 나타내는 환경계수 α값의 범위는 구간 $(0, 1)$이다.

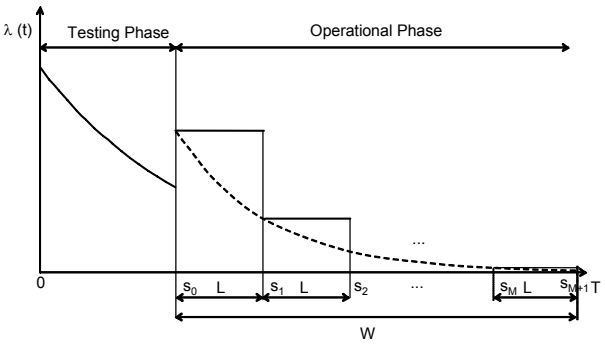

Figure 3. Failure rate function considering maintenance and the differences environment between during testing and operating

<Figure 4>에서 고장률 감소 형태를 살펴보면 s_{i+1}, $i = 0, \cdots, M$시점에서 고장률이 감소함을 알 수 있고 감소 정도는 $\lambda_o(t)$와 만나는 지점까지임을 알 수 있다. 그리고 구간 (s_i, s_{i+1}), $i = 0, \cdots, M$ 사이에는 보수를 하지 않음을 알 수 있다. 그러나 이들 각 구간에서 심각한 문제를 야기시킬 오류들, 가령 보안 문제 등은 지체 없이 패치 형태의 보수를 하게 된다(Scott, 2007). 따라서 이 상황에서는 패치 간격을 예측하기가 불가능하고, 무엇보다 패치 후 고장률 감소 정도의 예측이 불가능하다. 따라서 여러 형태의 고장률 감소가 존재할 것임을 직관적으로 알 수 있다. 예를 들면 <Figure 4>에서처럼 구

간(s_0, s_1) 사이에서 $\lambda_o'(t)$, $\lambda_{p0}(t)$, $\lambda_o(t)$와 같은 감소 형태가 존재할 수 있으며, 특히 여기서 $\lambda_o'(t)$는 패치를 전혀 하지 않는 경우의 고장률 함수(Lee *et al*. 2004의 고장률 함수와 동일)이고 $\lambda_o(t)$는 구간 (s_0, s_1)에서 발생하는 모든 오류를 패치 하는 경우의 고장률 함수가 된다.

따라서 사용단계에서 서비스 팩 배포와 패치를 고려한 고장률 함수는 구간 (s_i, s_{i+1}), $i=0, \cdots, M$ 사이에서 오류 하나당 패치율 β_i, $i=0, \cdots, M$에 따라서 여러 형태의 고장률 함수가 존재할 것이고, 한 예로 <Figure 4>의 $\lambda(t)$, $\lambda_{p0}(t)$, $\cdots$, $\lambda_{pM}(t)$인 형태를 볼 수 있다. 이때 출시 전 테스트단계 내의 고장률 함수를 $\lambda(t)$로 정의하고 출시 후 사용단계 내의 고장률 함수를 $\lambda_{pi}(t)$, $i=0, \cdots, M$으로 정의하면 다음과 같다.

$$\lambda(t) = abe^{-bt}, \qquad 0 \le t < s \tag{5}$$

$$\lambda_{pi}(t) = \frac{ab}{\alpha}\, e^{-b\left(s+\frac{s_i-s}{\alpha}+\beta_i\frac{t-s_i}{\alpha}\right)} \tag{6}$$

$$s+iL < t \le s+(i+1)L,$$
$$0 < \alpha \le 1, \quad 0 \le \beta_i \le 1, i=0, \cdots, M$$

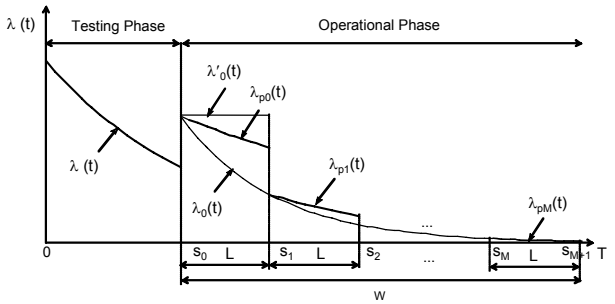

Figure 4. Failure rate function considering distribution of periodic service packs, uncertain patches and the differences environment between during testing and operating

3. 소프트웨어의 최적 출시시기 결정

Okumoto and Goel(1980)은 소프트웨어 신뢰성 성장 모형을 기반으로 신뢰성 또는 비용 척도에 기반을 둔 소프트웨어의 최적 출시시기 결정 문제를 다루었다. 즉 신뢰성 모형은 테스트단계 내(구간 $(0, s)$)에서 발견된 오류가 k개일 때, 사용단계 내의 구간 $[s, x)$에서 발견된 오류가 $k+1$개일 조건부 신뢰성 $R(x|s) \ge R_0$(R_0는 소프트웨어 시스템에 요구되는 최소한의 신뢰성)를 만족하는 s를 최적 출시시기로 결정하는 모형이다. 다음 비용모형의 설명에 앞서, 일반적으로 사용단계 내에서의 시스템 성능은 테스트 시간이 길어질수록 좋아진다. 그러나 테스트 시간이 길어지게 되면 출시 지연이 발생하게 되고 그 원인으로 인해 추가비용이 발생하게 된다. 따라서 이것이 테스트 시간을 줄이고 출시를 서두르도록 하는 이유가 된다. 이와 같은 상황을 고려하여 비용함수를 만들고, 이 함수 값이 최소가 되는 시기를 최적 출시시기로 결정하는 모형이 비용모형이다. 이후 Yamada and Osaki(1985)는 신뢰성모형과 비용모형을 확장한 모형을 제시하였다. 즉 제약조건 $R(x|s) \ge R_0$하에 비용함수의 값이 최소가 되는 시기를 최적 출시시기로 결정하는 모형이다. 본 논문은 소프트웨어 시스템이 이미 R_0를 만족(제약조건 $R(x|s) \ge R_0$를 만족)한다는 가정 하에서 최적 출시시기를 결정하였다.

다음 비용함수를 도출하기 위한 비용 구성을 살펴보면 <Figure 5>에서 보는 봐와 같이 테스트단계에서는 오류수정비용 $c_1 m(s)$와 테스트비용 $c_6 s$ 그리고 시스템 출시 후 사용단계에서는 오류수정비용 $c_2 m_o(T)$과 서비스 팩 배포 비용 $c_3 M$ 그리고 본 논문에서 새롭게 추가된 패치 배포 비용 $c_4 \sum_{i=0}^{M} \beta_i \int_{s_i}^{s_{i+1}} \lambda_o(t)\, dt$으로 구성된다. 또한 Lee *et al*.(2004)가 제시한 손실비용 $c_5 \sum_{i=0}^{M} \int_{s_i}^{s_{i+1}} \lambda_{pi}(t)\, dt$이 추가된다.

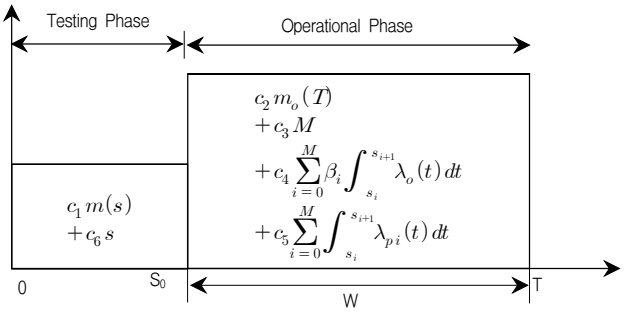

Figure 5. Cost components

부연하여 설명하면, 테스트단계에서의 오류 수정비용은 이 단계 내에서 테스팅하여 발견된 평균 오류 횟수인 $m(s)$를 수정하는 비용이고, 사용단계에서의 오류수정비용은 소프트웨어 시스템 출시 이후 수명주기 동안 발견된 평균 오류 횟수인 $m_o(T)$를 수정하는 비용이다. 그리고 패치 배포 비용은 서비스 팩 배포 구간 (s_i, s_{i+1}), $i=0, \cdots, M$ 사이에서 발견되는 오류에 대하여 패치를 시행함으로써 발생하는 비용이고, 손실비용은 실제로 사용하다가 고장이 발생함으로써 야기되는 비용이다. 이들을 고려한 비용함수는 다음과 같다.

$$C(s) = c_1 m(s) + c_2 m_o(T) + c_3 M + c_4 \sum_{i=0}^{M} \beta_i \int_{s_i}^{s_{i+1}} \lambda_o(t)\, dt + c_5 \sum_{i=0}^{M} \int_{s_i}^{s_{i+1}} \lambda_{pi}(t)\, dt + c_6 s \tag{7}$$

다음 비용함수 식 (7)이 최소가 되는 소프트웨어의 최적 출시시기 s^*를 찾아본다. 그러기 위해 비용함수 $C(s)$를 s에 대해서 일차미분과 이차미분을 하면 아래 식이 얻어진다. 이때

$dC(s)/ds = 0$인 s를 s_0라고 하였을 때 s_0는 식 (9)와 같다.

$$\frac{dC(s)}{ds} = abe^{-bs}\{c_1 - c_2(1-e^{-bW/\alpha}) - c_4A - c_5B\} + c_6 \quad (8)$$

$$s_0 = \frac{1}{b}\ln\left[\frac{ab\{c_2(1-e^{-bW/\alpha}) + c_4A + c_5B - c_1\}}{c_6}\right] \quad (9)$$

$$\frac{d^2C(s)}{ds^2} = ab^2e^{-bs}\{c_2(1-e^{-bW/\alpha}) + c_4A + c_5B - c_1\} \quad (10)$$

여기서 $A = \sum_{i=0}^{M}\beta_i e^{-biL/\alpha}(1-e^{-bL/\alpha})$,

$$B = \sum_{i=0}^{M}\frac{e^{-biL/\alpha}}{\beta_i}(1-e^{-b\beta_i L/\alpha})$$

다음 식 (8)의 우변항을 0이라 두면 $abe^{-bs} = c_6/\{c_2(1-e^{-bW/\alpha}) + c_4A + c_5B - c_1\}$으로 표현되며, 여기서 $c_2(1-e^{-bW/\alpha}) + c_4A + c_5B - c_1$를 W에 대한 함수 $f(W)$라고 하고, abe^{-bs}를 식 (5)에 의하면 다음과 같다.

$$\lambda(s) = \frac{c_6}{f(W)} \quad (11)$$

여기서 $f(W)$는 단조증가함수(증명 생략)임을 알 수 있다. 그리고 만약 $ab \le c_6/f(W)$라고 한다면, $\lambda(s)$는 단조감소함수(증명 생략)이고, $\lambda(0) = ab$이므로 식 (11)의 해는 존재하지 않는다. 그리고 $s \ge 0$에서 $dC(s)/ds > 0$이 성립하므로 비용함수 $C(s)$는 $s^* = 0$에서 최소값을 가진다.

그렇지 않고 만약 $ab > c_6/f(W)$이면 식 (11)의 해가 존재하며 유일하다. 그리고 $0 < s < s_0$에서 $dC(s)/ds < 0$이고, $s > s_0$에서 $dC(s)/ds > 0$이 성립하므로 비용함수 $C(s)$는 $dC(s)/ds = 0$이 되는 $s^* = s_0$에서 최소값을 가진다. 이상을 정리하면 소프트웨어의 최적 출시 시기는 다음과 같다.

$$s^* = \begin{cases} \frac{1}{b}\ln\left[\frac{abf(W)}{c_6}\right], & ab > c_6/f(W) \\ 0, & ab \le c_6/f(W) \end{cases} \quad (12)$$

여기서 $f(W) = \{c_2(1-e^{-bW/\alpha}) + c_4A + c_5B - c_1\}$,

$$A = \sum_{i=0}^{M}\beta_i e^{-biL/\alpha}(1-e^{-bL/\alpha}),$$

$$B = \sum_{i=0}^{M}\frac{e^{-biL/\alpha}}{\beta_i}(1-e^{-b\beta_i L/\alpha})$$

한편 테스트단계와 사용단계의 환경이 동일($\alpha = 1$)하고, 패치 배포를 시행하지 않는($\beta_i = 0,\ i = 0, \cdots, M$) 경우라면, $A = 0$, $B = Lb\sum_{i=0}^{M}e^{-biL}$이 되므로 s_0는 다음과 같다.

$$s_0 = \frac{1}{b}\ln\left[\frac{ab\left\{c_2(1-e^{-bW}) + c_5Lb\sum_{i=0}^{M}e^{-biL} - c_1\right\}}{c_6}\right] \quad (13)$$

이는 Lee *et al.*(2004)가 제시한 최적 출시시기 결정과 같기 때문에, 본 논문의 모형은 Lee *et al.*(2004)의 일반 모형임을 알 수 있다.

4. 수치예제

이 절에서는 테스트단계에서 발생하는 비용(오류수정비용, 테스트비용)과 출시 후 사용단계에서 발생하는 비용(오류수정비용, 서비스 팩 배포비용, 패치 비용, 손실비용)을 더한 총비용을 최소로 하는 소프트웨어의 최적 출시시기를 결정한다. 이를 위해 기존 논문(Goel and Okumoto, 1979; Koch and Kubat, 1983; Lee *et al.* 2004)과 저자 경험에 의해 아래 예제를 제시하였으며, 이 예제를 통해 본 논문의 모형을 설명하고, 최적 출시시기를 결정하였다. 아래 예제는 사용단계가 테스트단계보다 더 가혹한 환경인 경우와 사용단계에서 패치를 시행해야할 만큼 심각한 오류가 빈번하게 발생하는 경우를 나타내고 있다. 또한 1회 서비스 팩 배포비용보다 1회 패치 배포 비용이 더 큼을 나타낸다.

$a = 34,\ b = 0.00579$

$\alpha = 0.7$

$\beta_0 = 0.8,\ \beta_1 = 0.9,\ \beta_2 = 0.5$

$W = 1500,\ M = 2$

$c_1 = 200,\ c_2 = 1500,\ c_3 = 400,\ c_4 = 600,\ c_5 = 1500,\ c_6 = 10$

이상에서 $ab = 0.1969$이고 $c_6/f(W) = 0.0028$이므로 $ab > c_6/f(W)$이 성립한다. 그리고 식 (9)에 의해서 $s_0 = 737$이므로 식 (12)에 의해 $s^* = 737$이다. 즉 시스템을 737일 동안 테스팅한 후 출시하게 되고, 출시 후 1500일 동안 품질을 보증하므로 소프트웨어 수명주기는 만 6년이 된다. 이때 비용은 $C(s^*) = 16693$이다(<Figure 6> 참조).

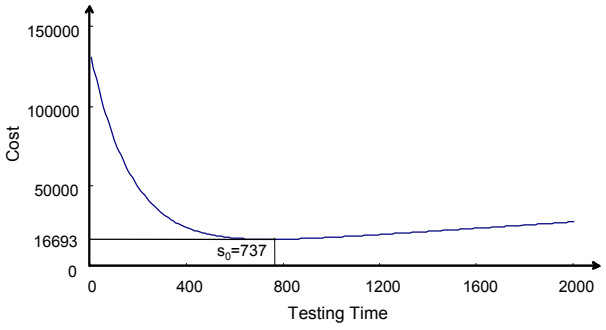

Figure 6. The cost optimal software release time

위 예제에서 테스트단계 내에서 오류 발견 환경과 사용단계 내에서 오류 발견 환경이 동일($\alpha=1$)하다는 조건하에 패치 배포를 시행하지 않는 경우($\beta_i=0,\ i=0,\cdots,M$), 식 (13)에 의해 최적 출시시기는 $s_0=821$이 된다. 따라서 앞 단락에서의 s^*와 비교했을 때, 테스팅을 85일 더 시행하게 되고, 이때 비용은 845가 더 소요됨을 알 수 있다.

다음 위 예제를 통해 본 논문의 모형을 좀 더 자세히 살펴보자. 우선 패치 배포를 시행해야 할 만큼 심각한 오류가 여러 경우($\beta=0.1,\ 0.15,\ 0.2,\ 0.5,\ 0.8$)로 발생할 때, 사용 환경에 따른 최적 출시시기의 변화 상태를 그래프(<Figure 7> 참조)에 의해 살펴보았다. 또한 패치 배포를 여러 경우($\beta=0.1,\ 0.5,\ 0.9$)로 시행할 때, 서비스 팩 배포 횟수에 따른 최적 출시시기의 변화를 그래프(<Figure 8> 참조)에 의해 살펴보았고, 시스템 출시 후 품질 보증기간이 여러 경우($W=200,\ 600,1200,\ 2000$)일 때, 환경계수에 따른 최적 출시시기의 변화를 살펴보았다(<Figure 9> 참조).

<Figure 7>은 여러 형태의 패치 배포를 시행하는 경우에 사용 환경의 변화에 따른 최적 출시시기의 변화 상태를 나타낸다. 이를 자세히 살펴보면, 테스트단계 내의 환경보다 사용단계 내의 환경이 더 가혹할수록 테스트 시간이 더 길어짐을 알 수 있다. 예를 들어 $\beta=0.1$인 경우, $\alpha=0.9$에서 $s^*=814$일이고, 이 보다 더 가혹한 환경인 $\alpha=0.3$에서 $s^*=923$일이므로 109일 더 테스팅함을 알 수 있다. 또한 시스템을 실제로 사용하면서 패치를 시행할 만큼 심각한 오류가 빈번하게 발생하는 상황일수록 테스트 시간이 줄어드는 것을 알 수 있다. 예를 들어 $\alpha=0.3$에서 $\beta=0.1$인 경우 $s^*=923$일인 반면, 심각한 오류가 더 빈번하게 발생하는 $\beta=0.5$에서는 $s^*=777$일로 테스트 시간이 146일 줄어드는 것을 알 수 있다. 그리고 사용단계에서 발견된 오류의 반 이상을 패치 하는 경우, 사용 환경이 변하더라도 테스트 시간의 차이가 크지 않음을 알 수 있다. 예를 들어 $\beta=0.8$인 경우 $\alpha=0.9$에서 $s^*=735$일이고, 사용 환경이 더 가혹한 $\alpha=0.3$에서 $s^*=738$일로 그 차이(3일)가 크지 않음을 알 수 있다.

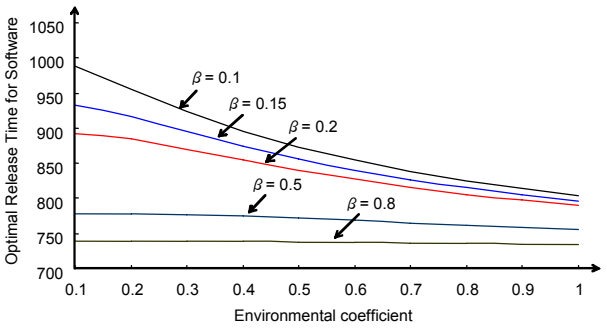

Figure 7. Dependence on the patch rate per error in optimal release time for software

<Figure 8>은 여러 경우의 패치 배포 시행에서 서비스 팩 배포 횟수에 따른 최적 출시시기의 변화 상태를 나타낸다. 이를 자세히 살펴보면, 서비스 팩 배포 횟수가 늘어날수록 테스트 시간이 줄어드는 것을 알 수 있다. 이러한 원인은 서비스 팩 배포 횟수가 늘어날수록 배포 비용은 증가할 것이나, 손실비용이 감소하기 때문에 결국 테스트 시간이 줄어드는 것으로 판단할 수 있다. 마찬가지로 패치 배포를 빈번하게 시행할수록 테스트 시간이 줄어드는 것을 알 수 있다. 예를 들어 서비스 팩 배포를 2회 시행한다고 하였을 때, $\beta=0.1$인 경우 $s^*=839$일이며 $\beta=0.9$인 경우 $s^*=731$일이므로 테스트 시간이 108일 줄어드는 것을 알 수 있다. 또한 서비스 팩 배포 횟수가 어느 (가령 <Figure 8>에서 14회) 이상이 되었을 때 패치 배포 횟수에 상관없이 테스트 시간이 비슷함을 알 수 있다. 예를 들어 서비스 팩 배포 횟수가 14회인 경우 $\beta=0.1$일 때 $s^*=730$일이며 $\beta=0.5$일 때 $s^*=726$일이고 $\beta=0.9$일 때 $s^*=726$일로 그 차이가 크지 않음을 알 수 있다.

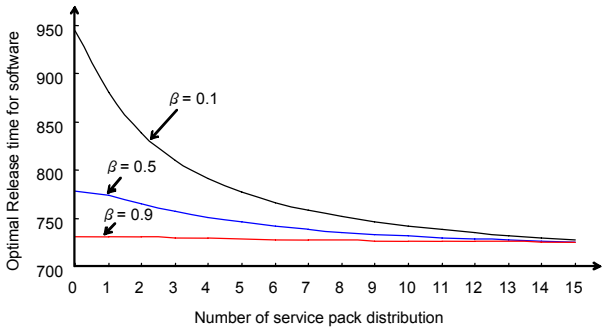

Figure 8. Optimal release time versus number of service pack distribution

<Figure 9>는 시스템 출시 후 품질 보증기간이 여러 경우일 때 환경계수에 따른 최적 출시시기의 변화를 나타낸다. 이를 자세히 살펴보면, 품질보증기간이 늘어날수록 테스트 시간이 길어짐을 알 수 있고 사용단계 내의 환경이 가혹할수록 테스트 시간이 길어짐을 알 수 있다. 예를 들어 출시 후 품질보증기간이 200일인 경우 $\alpha=0.9$에서 $s^*=663$일이고 품질보증기간이 2000일인 경우에는 $s^*=737$일이 되어 테스팅 시간이 74일 더 길어짐을 알 수 있다.

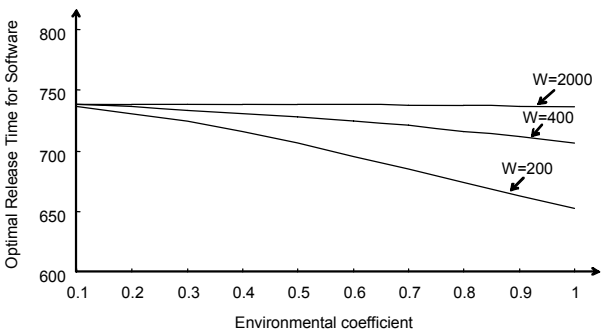

Figure 9. Dependence on warranty in optimal release time for software

이상을 정리하면, 사용단계의 환경이 가혹할수록 최적 출시시기가 늦어짐을 알 수 있고 패치 배포의 시행 횟수가 증가할수록 그 시기가 앞당겨짐을 알 수 있다. 그리고 사용단계에서 발생한 오류의 반 이상에 대해 패치를 실시하는 경우에는 그 시기가 비슷함을 알 수 있고, 서비스 팩 배포 횟수가 늘어날수록 최적 출시시기가 앞당겨짐을 알 수 있다. 또한 서비스 팩 배포 횟수가 어느 정도(가령 14회 이상, <Figure 8> 참조) 이상이 된다면 패치 배포 횟수에 상관없이 그 시기가 비슷함을 알 수 있고, 패치 배포 시행이 빈번한 경우(가령 <Figure 8>의 $\beta = 0.9$) 에는 서비스 팩 배포 횟수에 상관없이 그 시기의 차이가 비슷함을 알 수 있다. 또한 품질보증기간이 길어질수록 최적 출시시기가 늦어짐을 알 수 있다.

5. 결 론

본 논문의 목적은 소프트웨어 시스템 개발자가 개발과정에서 최적의 시점에서 테스팅을 멈추고 사용자에게 시스템을 출시하도록 하는데 있다. 이를 위해 기존에 연구된 보수를 고려한 최적 출시시기 결정 문제에서 다루지 않았던, 사용단계에서의 사용 환경을 고려하고 동시에 주기적 서비스 팩 배포와 심각한 고장 발생시 패치를 시행하는 경우를 고려함으로써 좀 더 정확한 최적 출시시기를 결정할 수 있도록 하였다.

본 논문은 기존 논문의 보수정책과 구별되는 정책에 의해서 신뢰성 성장 모형을 도출하였고, 이 모형을 기반으로 소프트웨어 시스템을 출시하기 전 테스트단계(testing phase) 내에서의 비용(오류수정비용, 테스트비용)과 출시 후 사용단계(operational phase) 내에서의 비용(오류수정비용, 서비스 팩 배포 비용, 패치 비용, 손실비용)을 더한 총비용이 최소가 되는 시기를 최적 출시시기로 결정하는 모형을 제시하였다. 아울러 본 논문에서는 기존 모형을 일반화한 모형을 제시하였다. 그리고 제시된 모형을 사용해 본 논문에서 제시한 수치예제를 풀어 소프트웨어의 최적 출시시기를 찾았고, 그 결과 사용단계의 환경이 가혹할수록 최적 출시시기가 늦어짐과 서비스 팩 배포 횟수가 증가할수록 또한 패치를 시행해야할 만큼 심각한 오류 발생이 증가할수록 그 시기가 앞당겨짐을 알 수 있었다.

향후 테스트단계와 사용단계가 구분되는 상황에서 시스템 출시 이후 주기적 서비스 팩 배포와 패치 배포가 불확실한 경우의 소프트웨어 최적 출시시기를 결정하는 문제에 있어서, 본 논문의 모형을 적용해 보는 것은 상당한 의미가 있다고 생각된다.

참고문헌

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

Huang, C. Y. and Lyu, M. R. (2005), Optimal release time for software systems considering cost, testing-effort, and test efficiency, *IEEE Transaction on Reliability*, **54**, 583-591.

Koch, H. S. and Kubat, P. (1983), Optimal release time for computer software, *IEEE Transaction on Software Engineering*, **SE-9**, 323-327.

Lee, C. S., Na, I. Y., Hong, J. K. and Lie, C. H. (2004), Optimal software release time considering maintenance during operation, *Journal of the Korean Institute of Industrial Engineers*, **30**, 261-266.

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

Okamura, H. K, Dohi, T., and Osaki, S. (2001), A reliability assessment method for software products in operational phase-proposal of an accelerated life testing model, *Electronics and Communications in Japan*, **84**, 25-33.

Okumoto, K. and Goel, A. L. (1980), Optimum release time for software systems based on reliability and cost criteria, *The Journal of systems and Software*, **1**, 315-318.

Ohtera, H. and Yamada, S. (1990), Optimal software-release time considering an error-detection phenomenon during operation, *IEEE Transaction on Reliability*, **39**, 596-599.

Phan, H. and Zhang, X. (1999), A software cost model with warranty and risk costs, *IEEE Transaction Computer*, **48**, 71-75.

Ross, S. M. (1985), Software reliability: the stopping rule problem, *IEEE Trans. Software Engineering*, **11**, 1472-1475.

Scott, C. (2007), Why service packs are better than patches, *Microsoft TechNet*, www.microsoft. com/technet/ archive/community/columns/security/essays/srvpatch.mspx.

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

Yamada, S., Tanio, Y., and Osaki, S. (1989), A software reliability evaluation method during operation phase, *Trans IEICE,* **J72-D-I**, 797-803.

Yamada, S. (1993), Software reliability measurement during operational phase and its application, *J Comput Software Eng* **1**, 389-402.

Yamada, S., Hishitani, J., and Osaki, S. (1993), Software-reliability growth with a weibull test-effort: a model & application, *IEEE Transaction on Reliability*, **42**, 100-106.

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.S.