▨응용논문

Musa-Okumoto 대수 포아송 실행시간 모형에 근거한 비용-신뢰성 최적정책

김대경
전북대학교 통계학과

Cost-Reliability Optimal Policies Based on Musa-Okumoto Logarithmic Poisson Execution Time Model

Dae-Kyung Kim
Dept. of Statistics, Chonbuk National University

Abstract

It is of great practical interest to decide when to stop testing a software system in the development phase and transfer it to the user. This decision problem- called an optimal software release one- is discussed to specify the appropriate release time. In almost all studies, the software reliability models used are nonhomogenous Poisson process(NHPP) model with bounded mean value function. HNPP models with unbounded mean value function are more suitable in practice because of the possibility of introducing new faults when correcting or modifying the software. We discuss optimal software release policies which minimize a total average software cost under the constraint of satisfying a software reliability requirement. A numerical example illustrates the results.

1. 서론

최근 몇 년 동안 운영체제, 관리 프로그램 그리고 응용 프로그램들과 같은 소프트웨어 시스템들이 전보다 더욱 커지고 복잡해져가고 있다. 소프트웨어 고장들로 인한 컴퓨터 시스템의 고장은 우리 사회에 엄청난 손실을 초래할 수도 있다. 따라서 소프트웨어 신뢰도는 현대의 소프트웨어 생산품 개발에서 중요한 문제들 중의 하나이다. 소프트웨어 신뢰도 엔지니어링에서의 연구 활동은 지난 20년 넘게 행해졌고 많은 신

뢰도 성장 모델들이 소프트웨어에 남아 있는 고장들의 수와 소프트웨어 신뢰도의 추정을 위해서 제안되었다.

일반적으로 소프트웨어의 개발과정은 다음과 같은 연속적인 네 단계 즉, 설계단계, 디자인, 코딩 그리고 시험으로 구성된다. 그 중에서도 소프트웨어 시험은 보통 총 개발비용의 50% 이상의 비용이 들기 때문에 소프트웨어를 언제 방출해야 하는가는 소프트웨어 개발자들에게는 중요한 관심사 중의 하나이다. 만약 발견되지 않은 많은 고장들이 방출한 후 운용할 때 일어난다면 그들의 보전비용(maintenance cost)은 증가한다.

실제적으로 소프트웨어 시스템의 시험을 마치고 그것을 사용자에게 넘기는 시기를 결정하는 것은 중요하다. 시험이 너무 일찍 끝내면 많은 고장들이 남게되어 개발 회사는 후에 결점을 제거하는데 많은 비용을 감수해야 하고, 시험이 너무 길게 지속된다면 신뢰도는 증가하나 시험 비용이 너무 많이 들고 생산품 소개가 그만큼 늦어지게 된다. 따라서 최적 소프트웨어 방출 시간을 결정하는 것은 중요하다.

소프트웨어 방출 시간을 결정하는 문제에 관한 많은 문헌들[1, 2, 3, 7, 8, 9]이 있다. 대부분의 연구에서 순기 비용이 주어진 신뢰도 요구를 조건으로 하여 최소화되었다. 그러한 논문들에서는 소프트웨어가 유한개의 고장이 있다는 것이고 고장 제거는 새로운 고장을 초래하지 않는다고 가정하였다. 또한 한계가 있는 평균값 함수를 가진 NHPP(Nonhomogenous poission process)모형을 가정하였다. 그러나 이 가정은 소프트웨어 시스템을 수정하고 고칠 때에 또 다른 고장을 초래하기 때문에 의문시된다[10].

실제로 소프트웨어 시스템의 수정하고 고칠 때에 고장들을 초래하는 것은 특히 항상 완전한 고장 제거가 거의 불가능한 큰 소프트웨어에 대해서 일어난다. 따라서 한계가 없는 평균값 함수를 가진 NHPP 모델들이 실제에서는 더욱 알맞다.

이 논문에서는 한계가 없는 NHPP 모델들에 관한 최적 방출 시간을 결정하는 문제를 연구하고자 한다. 2 절과 3 절에서는 신뢰도 요구와 비용 최소화에 각각 관하여 최적 시험 시간을 토의한다. 위의 두 가지의 기준을 동시에 고려한 최적 시험 시간이 4 절에서 연구되었다. 마지막으로 예제가 최적 방출 시간 절차의 결정을 하는 것을 보이기 위하여 주어졌다.

2. 신뢰도 요구에 관한 방출시간

신뢰도 요구가 주어졌을 때 소프트웨어 신뢰도 모형을 사용해서 최적 방출 시간을 결정하는 것은 간단하다. 소프트웨어 신뢰도인 $R(x \mid T)$는 마지막 고장이 시험 시간 $t(t \geq 0, x > 0)$에서 일어났다는 조건하에서 소프트웨어 고장이 $(t, t+x)$에서는 일어나지 않을 확률로서 정의된다. 그러면 소프트웨어가 방출 시간에서의 신뢰도가 $R_0 = R(x \mid T)$가 되도록 요구되어진다고 가정하자. 시험 시간 t까지 발견될 수 있

는 고장의 기대 누적 갯수인 평균값 함수 $m(t)$를 가진 NHPP 모형에 대해서 방출시간은 다음에 의하여 결정되어진다.

$$\begin{aligned} R(x \mid T) &\equiv P(X_i > x \mid S_{i-1} = t) \\ &= \exp[-\{m(t+x) - m(t)\}], \quad (t \geq 0, x \geq 0) \end{aligned} \tag{2.1}$$

여기에서 X_i는 $i-1$번째 고장과 i번째 고장간의 시간을 그리고 S_{i-1}은 $i-1$번째 고장이 발견되기까지의 시간을 나타내는 확률변수이다.

유한이 아닌 평균값 함수를 가지는 많은 신뢰도 성장 모형들이 있다. 가장 잘 알려진 모형들은 Duane 모형 그리고 Musa - Okumoto의 대수적인 포아송 실행시간 모형(이하 Musa - Okumoto 모형)이다[6]. 또한 log-power 모형도 최근 많은 데이터에 대해서 좋은 모형이라고 보여졌다[11]. 우리는 이 논문에서는 예시적인 목적을 위해서 Musa - Okumoto 모형을 사용하였다.

우리는 다음과 같은 평균값 함수를 가지는 Musa - Okumoto 모형을 사용해서 신뢰도 요구를 만족시키기에 필요한 시간을 유도하였다.

$$m(t) = \frac{1}{\theta} \ln(\lambda_0 \theta t + 1) \tag{2.2}$$

여기에서 λ_0는 초기고장강도(initial failure intensity)이고 $\theta(>0)$는 고장강도쇠퇴모수(failure intensity decay parameter)이다. 소프트웨어가 방출 시간에서 신뢰도 $R_0 = R(x \mid t)$를 가져야 한다면 Musa - Okumoto 모델에 대한 소프트웨어 신뢰도는 식 (2.1)에 의해서 구하면 다음과 같다.

$$R(x \mid t) = \exp[\frac{1}{\theta} \ln(\lambda_0 \theta t + 1) - \frac{1}{\theta} \ln\{\lambda_0 \theta (t+x) + 1\}]$$

그리고 최적 방출시간 T_R은 다음 방정식의 해이다.

$$\ln R_0 = \frac{1}{\theta} \ln(\lambda_0 \theta T_R + 1) - \frac{1}{\theta} \ln\{\lambda_0 \theta (T_R + x) + 1\} \tag{2.3}$$

3. 비용 최소화에 근거한 방출시간

비용 최소화에 근거한 최적 방출시간은 신뢰도 모형과 함께 비용 모형에 의해서 결정되어진다. 소프트웨어 방출 시간을 T로 나타내고 $m(T)$와 $m(\infty)$는 각각

$(0, T]$와 $(0, \infty)$의 기간에 발견된 기대된 고장들의 갯수로 표현된다. $C(T)$를 소프트웨어 순기 동안에 기대되는 소프트웨어 비용이라고 한다면 $C(T)$는 다음처럼 표현된다.

$$C(T) = c_1 m(T) + c_2[m(\infty) - m(T)] + c_3 T$$

여기에서 c_1은 시험 동안에 하나의 고장을 수리하는 비용이고 c_2는 가동 중에 하나의 고장을 수리하기 위한 비용 ($c_2 > c_1$), 그리고 c_3는 단위 시간당 시험 비용이다.

총 비용이 최소화되어질 때 문제는 유한이 아닌 평균값 함수를 가진 NHPP 모형에 대해서 일어날 수 있다. 무한 수명에 대한 비용함수 $C(T)$는 모델의 이 형태에 대해서 $m(\infty)$로서 이 경우에 직접 사용되어질 수 없다. 이와 같은 형태의 비용 함수를 사용할 때 소프트웨어의 수명시간인 T_{LC}를 지정하여 분석하였다. T_{LC}는 소프트웨어마다 서로 다른 임의의 값이기 때문에 유한 NHPP 모형이라고 확신할 수는 없다.

비용함수를 고려하고 소프트웨어의 모든 수명에서 총 비용을 최소화함으로서 최적 검사시간을 결정할 수 있다. 이것은 다음에 의하여 얻어질 수 있다.

$$\frac{dC(T)}{dT} = 0 \quad \text{그리고} \quad \frac{d^2C(T)}{d^2T} > 0$$

따라서 비용함수 $C(T)$는 유일한 최소값을 갖는다.

따라서 Musa-Okumoto 모형을 사용해서 시험과 수정에 따른 평균 비용 $C(T)$는 다음과 같이 주어진다.

$$\begin{aligned} C(T) &= c_1 m(T) + c_2[m(\infty) - m(T)] + c_3 T \\ &= (c_1 - c_2)\frac{1}{\theta}\ln(\lambda_0 \theta T + 1) + \frac{c_2}{\theta}\ln(\lambda_0 \theta T_{LC} + 1) + c_3 T \end{aligned} \tag{3.1}$$

여기에서 T_{LC}는 소프트웨어 지정된 수명이다.

T에 관해서 비용함수 $C(T)$를 미분하면 다음과 같은 방정식의 해인 최적 방출시간 T_C를 얻는다.

$$\frac{\lambda_0(c_1 - c_2)}{1 + \lambda_0 \theta T_C} + c_3 = 0 \tag{3.2}$$

최적 방출시간은 우리가 사용한 비용 모형의 형태에 대해서 지정된 수명, T_{LC}에 의존하지 않는다는 것을 알 수 있다.

수명이 방출시간에 영향을 끼치지 않는다는 것은 또한 유한인 평균값 함수를 가진 NHPP 모델에 대해서도 또한 사실이다. 그러한 모형일지라도 우리는 역시 무한 수명을 가정할 수 있다. 유한이 아닌 평균값 함수를 가진 NHPP 모형들을 사용할 때 함수적인 변화가 만들어질 때 새로운 결점들이 초래됨으로서 몇 개의 고장이 야기될 수 있다는 것은 중요하다.

4. 결합된 기준에 근거한 방출시간

실제로 만족할 만한 신뢰도를 얻어야 하고 동시에 시스템 고장과 연계된 기대 총 비용을 최소화하기 위해서 필요하다면 시험을 계속해야 한다. 3 절과 4 절로부터 얻어진 결과들을 합치면 결합된 기준에 근거한 최적 시험 시간을 결정하는 것은 쉽다. 즉, 신뢰성 요구를 만족하는 총 순기 비용을 최소화 할 수 있다.

Musa-Okumoto 모형을 사용한 최적 방출시간 T_{op}는 T_R과 T_C에 관해서 다음처럼 표현된다.

$$T_{op} = \max(T_C, T_R) \tag{4.1}$$

여기에서 T_R과 T_C는 다음 두 방정식에 의해서 계산되어질 수 있다.

$$R(x \mid t) = \left(\frac{\lambda_0 \theta T_R + 1}{\lambda_0 \theta (T_R + x) + 1} \right)^{\frac{1}{\theta}}$$

$$\frac{\lambda_0 (c_1 - c_2)}{1 + \lambda_0 \theta T_C} + c_3 = 0$$

유한이 아닌 평균값 함수를 가진 NHPP 모형에 대해서 유한 수명을 규정하는 것이 필요하다고 할지라도 수명이 최적 방출시간보다 길다면 이것은 최적 방출시간과 무관하다는 것이다.

5. 예제

데이터는 두 가지의 형태 즉, 고장 구간들 혹은 구간별 고장들의 갯수로 얻어질 수 있으나 본 예제에서는 고장 시간 데이터(Musa [5])를 다루어 보고자 한다. 다음 <표 5.1>은 시스템 T1의 수집된 고장시간 데이터의 일부이다.

< 표 5.1 > 시스템 T1에 대한 고장시간 (CPU sec)

3	33	146	227	342	351	353	444	556	571	709	759	836	860	968	1056	1726	1846	1872
1986	2311	2366	2608	2676	3098	3278	3288	4434	5034	5049	5085	5089	5089	5097				
5324	5389	5565	5623	6080	6380	6477	6740	7192	7447	7644	7837	7843	7922	8738				
10089	10237	10258	10491	10625	10982	11175	11411	11442	11811	12559	12559							
12791	13121	13486	14708	15251	15261	15277	15806	16185	16229	16358	17168							
17458	17758	18287	18568	18728	19556	20567	21012	21308	23063	24127	25910							
26770	27753	28460	28493	29361	300085	32408	35338	36799	37642	37654	37915							
37915	40580	42015	42045	42188	42296	42296	45406	46653	47596	48296	49171							
49416	50145	52042	52489	52875														

관찰된 데이터의 모수를 추정하기에 앞서 관찰된 데이터가 통계적으로 Musa-Okumoto 모형에 잘 적합되는지를 점검하기 위해서 적합도 검정을 하는 것이 중요하다. NHPP 모형에 대해서 Kolmogorov-Smirnov (K-S) 검정 통계량에 근거한 적합도 검정을 하였다. 이 적합도 검정은 관찰 데이터의 표본크기가 작을지라도 유용한 것으로 알려져 있다. 고장 발생 시간 데이터에 대한 Kolmogorov-Smirnov (K-S) 검정 통계량은 다음에 의하여 주어진다.

$$D = \max_{1 \le i \le n-1} \{D_i\} \tag{5.1}$$

$$D_i = \max \left\{ \left| \frac{\hat{m}(s_i)}{\hat{m}(s_n)} - \frac{i}{n-1} \right|, \left| \frac{\hat{m}(s_i)}{\hat{m}(s_n)} - \frac{i-1}{n-1} \right| \right\} \tag{5.2}$$

만약 유의수준 α가 주어졌을 때 (5.1)에 의해서 계산된 D값이 기각값인 $D_{n-1;\alpha}$보다 작다면 주어진 데이터가 Musa-Okumoto 모형에 잘 적합된다고 해석할 수 있다. 계산을 하면 $D = 0.064 \le D_{113;0.05} = 0.1274$이기 때문에 Musa-Okumoto 모형을 적용할 수 있다.

식(2.3)과 (3.2)는 모두 모르는 모수 λ_0, θ에 의존하기 때문에 이를 추정해야 한다. 가정한 시험 종결시간인 $\tau_e = 53280$(CPU 초)까지 $m = 114$개의 고장이 관찰되었다고 가정하여 추정하기로 한다.

$\hat{\theta}$과 $\hat{\lambda_0}$의 최우추정량은 다음과 같이 구할 수 있다[6].

$$\hat{\theta} = \frac{1}{m} \ln(\hat{\phi}\tau_e + 1) \tag{5.3}$$

$$\hat{\lambda}_0 = \hat{\phi} / \hat{\theta} \tag{5.4}$$

여기에서 $\hat{\phi}$는 $\frac{\partial L}{\partial \phi} = \frac{m}{\phi} - \sum_{i=1}^{m} \frac{\tau_i}{\phi \tau_i + 1} - \frac{m \tau_e}{(\phi \tau_e + 1)\ln(\phi \tau_e + 1)} = 0$근 이고 $\hat{\phi} = \hat{\lambda}_0 \hat{\theta}$ 이다.

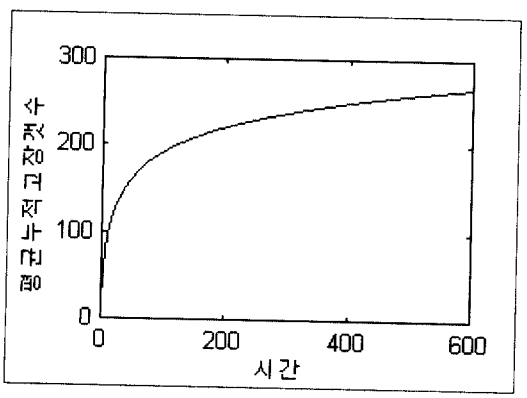
$\hat{\phi}$를 수치해석적인 방법으로 구하면 $\hat{\phi} = 0.952$ / (CPU 시간) 이고 따라서 $\hat{\theta} = 0.0238$ /고장, $\hat{\lambda}_0 = 40$ 고장 / (CPU 시간) 이다. 참고로 위에서 추정된 모수들을 이용해서 $\hat{m}(t)$와 신뢰도 평가 척도들인 $\hat{R}(x \mid t)$, $\widehat{MTBF}(t)$을 다음과 같이 구할 수 있다.

$$\hat{m}(t) = \frac{1}{0.0238} \ln[(40)(0.0238)t + 1] \tag{5.5}$$

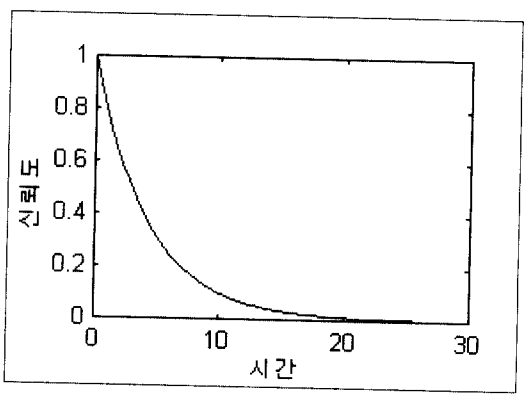
$$\hat{R}(x \mid t) = \left(\frac{40 \times 0.0238t + 1}{40 \times 0.0238(t + x) + 1} \right)^{\frac{1}{0.0238}} \tag{5.6}$$

$$\widehat{MTBF}(t) = \left(\frac{40}{40 \times 0.0238t + 1} \right)^{-1} \tag{5.7}$$

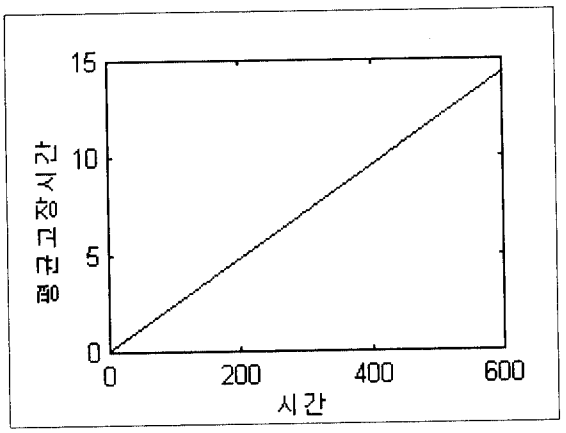
비용 모수들 $c_1 = 2(\$)$, $c_2 = 6(\$)$ 그리고 $c_3 = 10(\$)$ 라고 가정하고 시스템 수명 시간은 10000 이고 $x = 0.3$에 대해서 $R_0 = 0.95$의 신뢰도를 가질려면 T_R과 T_C는 각각 175.32와 15.76이다. 따라서 최적 방출시간 T_{op}는 175.32 시간이다.



< 그림 1 > 평균누적고장갯수



< 그림 2 > 신뢰도



< 그림 3 > 평균고장시간

6. 결론

이 논문에서 토론된 최적 소프트웨어 방출 정책을 사용해서 소프트웨어 개발자들은 소프트웨어 시스템이 방출되는 최적 시간을 결정할 수가 있다. 이것은 그들에게 소프트웨어 개발 과정을 디자인하는데 도움을 줄 수가 있고 충분하지만 과도하지 않는 시간이 시험 동안 제공되어지고 제작품은 제때에 혹은 예정보다 빨리 소비자에게 방출되어질 수가 있다.

소프트웨어 방출시간에 대한 거의 모든 출판된 연구들은 유한인 평균값 함수를 가진 NHPP 모델들에 근거한다. 소프트웨어가 클 때 이것은 수정과 변경에서 결점의 발생은 거의 피할 수 없을 때 유한이 아닌 NHPP 모델은 더욱 실제적이 되고 방출시간은 그러한 모델을 사용해서 결정되어져야 한다. Musa-Okumoto 모형에 근거한 최적 방출시간이 여기에서 사용되었다.

참고문헌

[1] S.R. Dalal, C.L. Mallows(1988), "When should one stop testing software," *Journal of American Statistical Association*, Vol. 83, pp. 872-879.

[2] P.K. Kapur, R.B. Garg(1989), "Cost-reliability optimum release policies for a software system under penalty cost," *Journal of Systems Science*, Vol. 20, pp. 2547-2562.

[3] H.S. Koch, P. Kubat(1983), "Optimal release time of computer software," *IEEE Transactions on Software Engineering*, Vol. SE-9, pp. 323-327.

[4] John D. Musa, Anthony Iannino, Kazuhiro Okumoto(1987), Software Reliability: Measurement, Prediction, Application; McGraw-Hill, ISBN 0-07-044093-X.

[5] John D. Musa(1979), "Software Reliability Data," report available from Data and Analysis Center for Software, Rome Air Development Center, Rome, NY.

[6] John D. Musa, K. Okumoto(1984), "A Logarithmic Poisson Execution Time Model for Software Reliability Measurement," *Proceedings the 7th International Conference on Software Engineering*, pp 230-238.

[7] S. Yamada, S. Osaki(1985), "Cost-reliability optimal release policies for software systems," *IEEE Transactions on Reliability*, Vol. R-34, pp. 422-424.

[8] S. Yamada, S. Osaki(1987), "Optimal software release policies with simultaneous cost and reliability requirements," *European Journal of Operations Research*, Vol. 31, pp. 46-51.

[9] M. Xie(1991), "On the determination of optimum software release time," *Proceeding International Symposium on Software Reliability Engineering*, Vol. May 17-19, Texas, USA, pp. 37-42.

[10] M. Xie(1991), Software Reliability Modelling, World Scientific Publisher, Singapore.

[11] M. Zhio, M, Xie(1993), "On the log-power software reliability model," *Proceeding International Symposium on Software Reliability Engineering*, Oct. 7-10, North Carolina, USA, pp. 14-22.