한국정보전자통신기술학회논문지(jkiiect)'16-04, Vol.9 No.2
논문 16-09-02-171 http://dx.doi.org/10.17661/jkiiect.2016.9.2.171

Lomax 분포의 형상모수에 근거한 소프트웨어 신뢰성 비용모형에 관한 연구

양태진*

A Software Reliability Cost Model Based on the Shape Parameter of Lomax Distribution

Tae-Jin Yang*

요 약 소프트웨어 개발과정에서 소프트웨어 신뢰성은 매우 중요한 이슈이다. 소프트웨어 고장분석을 위한 무한 고장 비동질적인 포아송과정에서 고장발생률이 상수이거나, 단조 증가 또는 단조 감소하는 패턴을 가질 수 있다.본 연구에서는 소프트웨어 제품 테스팅 과정에서 고장 수명분포의 형상모수를 고려한 소프트웨어 신뢰성 비용 모형에 대하여 연구 하였다. 소프트웨어 신뢰성 분야에서 많이 사용되는 Lomax-NHPP 신뢰 성장 모형에 대한 비용 비교 문제를 제시하였다. 소프트웨어 고장모형은 무한고장 비동질적인 포아송과정을 이용하고 모수추정법은 최우추정법을 이용 하였다. 따라서 본 논문에서는 형상모수를 고려한 소프트웨어 비용모형 분석을 위하여 소프트웨어 고장시간 자료를 적용하여 비교 분석하였다. 대용량 소프트웨어가 수정과 변경하는 과정에서 결함의 발생을 거의 피할 수 없는 상황이 현실이다. 신뢰성 요구를 만족하고 총비용을 최소화하는 상황이 최적방출시간이다. 경우에 따라서는 왜도와 첨도 측면에서 효율적인 카파분포, 지수화지수분포 등 업데이트된 분포에 대한 방출 시기 문제를 비교 분석하는 연구도 가치 있는 일이라 판단된다. 이 연구를 통하여 소프트웨어 개발자들은 최적방출시간과 경제적 개발 비용을 파악 하는데 도움을 줄 수 있으리라 사료 된다.

Abstract Software reliability in the software development process is an important issue. Software process improvement helps in finishing with reliable software product. Infinite failure NHPP software reliability models presented in the literature exhibit either constant, monotonic increasing or monotonic decreasing failure occurrence rates per fault. In this study, reliability software cost model considering shape parameter based on life distribution from the process of software product testing was studied. The cost comparison problem of the Lomax distribution reliability growth model that is widely used in the field of reliability presented. The software failure model was used the infinite failure non-homogeneous Poisson process model. The parameters estimation using maximum likelihood estimation was conducted. For analysis of software cost model considering shape parameter. In the process of change and large software fix this situation can scarcely avoid the occurrence of defects is reality. The conditions that meet the reliability requirements and to minimize the total cost of the optimal release time. Studies comparing emissions when analyzing the problem to help kurtosis So why Kappa efficient distribution, exponential distribution, etc. updated in terms of the case is considered as also worthwhile. In this research, software developers to identify software development cost some extent be able to help is considered.

Key Words : Software Development Cost Model, Non-Homogeneous Poisson Process, Lomax Distribution, Shape Parameter, Reliability

*Corresponding Author : Industry-Academic Cooperation Foundation, Namseoul University(solomon645@nsu.ac.kr)
Received April 14, 2016 Revised April 20, 2016 Accepted April 26, 2016

1. 서론

소프트웨어 신뢰성은 일정한 환경조건에서 일정기간동안 고장이 나지 않고 운영 할 수 있는 확률이다.

따라서 이러한 소프트웨어 신뢰성은 시스템 신뢰도에 영향을 주는 중요한 요소가 되고 디자인 속성 측면에서는 하드웨어 신뢰성과는 다른 면을 가지고 있다.

소프트웨어의 다양한 기능은 소프트웨어 신뢰성 문제들에 관한 주요한 요인이 된다. 소프트웨어의 신뢰도의 일반적인 정의는 일정한 기간 동안 주어진 환경 하에서 컴퓨터 프로그램을 고장 없이 사용할 수 있는 확률을 의미한다. 결국 소프트웨어 개발 과정에서 소프트웨어 신뢰성은 중요한 문제이다. 이 문제는 사용자의 요구조건과 테스팅 비용을 만족시켜야 한다.

소프트웨어 테스팅(디버깅)면에서 비용을 줄이기 위해서는 소프트웨어의 신뢰성의 변동과 테스팅 비용을 사전에 알고 있어야 효율적이다. 따라서 신뢰도, 비용 및 방출 시간의 고려사항을 가진 소프트웨어 개발 과정은 필수 불가결 하다.

지금까지 많은 소프트웨어 신뢰성 모형이 제안 되었다. 이 중에서 비동질적 포아송 과정(Non-homogeneous Poisson process; NHPP)에 의존한 모형[1]은 에러 탐색 과정측면에서는 우수한 모형이고 이러한 모형은 결함이 발생하면 즉시 제거되고 디버깅 과정에서 새로운 결함이 발생되지 않는다는 가정을 하고 있다.

이 분야에서 Gokhale과Trivedi [1]은 고양된 비동질적인 포아송 과정(Enhanced NHPP) 모형을 제시하였고 Goel 과 Okumoto [2]은 결함의 누적수가 S 형태나 지수적 형태(S-shaped or exponential-shaped)를 가진 평균값 함수(Mean value function)를 이용한 지수적 소프트웨어 신뢰성 모형(Exponential software reliability growth model)을 제안 하였다. 이모형에 의존한 일반화 모형은 Yamada 와 Ohba [3]에 의해 지연된 S-형태 신뢰 성장모형(Delayed S-shaped reliability growth model)과 변곡된 S-형태 신뢰성장모형(Inflection S-shaped reliability growth model)이 제안되었다. Zhao [4]는 소프트웨어 신뢰도에서 변환점 문제를 제시하였고 Shyur [5]는 변환점을 이용한 일반화한 신뢰도 성장 모형을 제안하였다. Pham와 Zhang[6]는 테스팅 커버리지(Coverage)를 측정하여 소프트웨어 안정도를 평가 할 수 있는 소프트웨어 안정도 모형을 제시했다. 비교적 최근에, Huang [7]은 일반화 로지스틱 테스팅 노력 함수(Generalized logistic testing-effort function)와 변환점 모수(Change-point parameter)를 통합하여 효율적인 소프트웨어 신뢰성 예측 기술을 제시하기도 하였다. 그리고 최근에는 S-형태 모형은 소프트웨어 관리자들이 소프트웨어 및 검사 도구에 익숙해지는 학습 과정을 설명할 수 있다고 하였다[8]

본 논문에서는 소프트웨어 신뢰성 분야에서 많이 사용되는 Loxmax-NHPP 신뢰 성장 모형에 대한 소프트웨어 개발 비용 모형을 비교 제시하였다.

2. 관련연구

2.1 NHPP 소프트웨어 신뢰성

$N(t)$을 시간 t까지 검출된 소프트웨어의 누적 고장수라고 하고, $m(t)$를 이에 대한 기대값를 나타내는 평균값 함수(Mean Value Function)로 가정하고 $\lambda(t)$을 강도함수(Intensity function) (즉, t에서의 순간 결함 검출율)이면 비동질 포아송 과정(NHPP)은 누적 고장수인 $N(t)$는 모수 $m(t)$을 가진 포아송 확률밀도함수 (Probability density function)로 알려져 있다. 즉,

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

따라서, NHPP모형에서 평균값함수 $m(t)$와 강도함수 $\lambda(t)$는 다음과 같은 관계로 표현할 수 있다[1, 9].

$$m(t)=\int_0^t \lambda(s)ds\ ,\ \frac{d\,m(t)}{dt}=\lambda(t) \qquad (2)$$

이처럼 시간관련모형(Time domain models)들은 NHPP에 의해서 확률고장과정으로 설명이 가능하다[1]. 이러한 NHPP모형들은 유한고장모형과 무한고장 범주로 분류한다[1]. 유한고장 NHPP모형에서는 시간 $(0,t]$ 까지 탐색되어 질 수 있는 결함의 기대값을 θ라고 표현하면 유한고장 NHPP 모형의 평균값 함수와 강도함수는 다음과 같이 표현할 수 있다[1, 9].

$$m(t)=\theta F(t)\ ,\ \lambda(t)=\theta F'(t) \qquad (3)$$

반면에 무한고장 NHPP모형들은 수리시점에서도 고장이 발생할 상황을 반영하기 위하여 기록멈춤 통계량(Record breaking statistics)을 사용하는 RVS(Record Value Statistics)모형을 사용할 수 있다고 하였고 이 RVS모형과 NHPP모형에 관해서 평균값함수는 다음과 같이 된다고 하였다[1].

$$m(t)=\ -\ln(1-F(t)) \qquad (4)$$

따라서 (2)식과 (4)식을 연관시키고 $f(t)$을 확률밀도함수, $F(t)$을 분포함수라고 하면 NHPP의 강도함수는 다음과 같이 위험함수($h(t)$)가 된다.

$$\lambda(t)\ =\ m'(t)\ =\ f(t)/(1-F(t))=h(t) \qquad (5)$$

시간 $(0,t]$ 까지 조사하기 위한 시간절단(Time truncated)모형은 n번째까지 고장시점 자료를

$$x_n=\sum_{i=1}^{n} t_i\ \ (i=1,2,\cdots,n\,;0\le x_1\le x_2\le\cdots\le x_n) \qquad (6)$$

이라고 하면 n번째까지 고장시점이 관찰된 고장절단 모형일 경우에 데이터 집합 D_{x_n}은 $\{x_1,x_2\cdots x_n\}$으로 구성되며, 이 고장절단모형에서 Θ을 모수공간이라고 표시하면 NHPP모형의 우도함수는 다음과 같이 알려져 있다[1, 9].

$$L_{NHPP}(\Theta|\ \underline{x}\ )=\left(\prod_{i=1}^{n}\lambda(x_i)\right)\exp[-m(x_n)] \qquad (7)$$

단, $\underline{x}=(x_1,x_2,x_3,\cdots,x_n]$

2.2 소프트웨어 개발 비용

소프트웨어 비용 모형은 다음과 같이 정의 된다[11, 12].

$$\begin{aligned}E&=E_1+E_2+E_3+E_4 \qquad (8)\\&=E_1+C_2\times t+C_3\times m(t)+C_4\times[m(t+t')-m(t)]\end{aligned}$$

단, E: 소프트웨어 개발 예상 총비용

E_1: 소프트웨어 설계 및 초기 소프트웨어 개발의 비용(분석 데이터, 소프트웨어 개발 전문가의 수 , CPU 운용시간 등)

E_2: 단위 시간당 소프트웨어 테스팅 비용(상수) 즉, $E_2=C_2\times t$ (단, C_2는 단위 시간당 테스팅 비용이고 t는 테스팅 시점)

E_3: 기본 결함을 감지하고 결함을 제거하는 등의 활동으로 하나의 결함을 제거하는 비용 즉, $E_3=C_3\times m(t)$ (단, C_3는 테스팅 과정에서 하나의 결함을 제거하는 비용, $m(t)$는 t시점에서 탐색되어 질 수 있는 결함의 기대수)

E_4: 운영 소프트웨어 시스템에서 남아있는 모든 결함을 제거하는 비용(상수)

즉, $E_4=C_4\times[m(t+t')-m(t)]$ (단, C_4는 소프트웨어 출시 된 이후에 소프트웨어 운영 단계에서 사용자가 관찰되는 결함수정 비용, t' 는소프트웨어 시스템을 출시 한 후 운영 및 소프트웨어를 유지할 수 있는 시간)

현실적으로는 C_4 는 C_2 와 C_3 보다 높은 비용을 나타낸다. 그러므로 최적의 소프트웨어 방출시간 (t)는 다음과 같이 유도 할 수 있다.

$$\frac{\partial E}{\partial t}=E'=(E_1+E_2+E_{3+}E_4)'=0 \qquad (7)$$

3. 제안한 Lomax-NHPP 신뢰성모형을 이용한 소프트웨어 개발비용 모형

파레토 유형 II 분포족(Pareto Type II distribution family)에 속해있는 로맥스분포는 두꺼운 꼬리 확률 분포(heavy-tail probability) 속성을 따르기 때문에 경영, 경제, 수리보험 분야에서 모형화는데 흔히 사용되는 분포이다. 이러한 다양한 분야에 사용되는 로맥스분포의 확률밀도함수와 분포함수는 다음과 같이 알려져 있다 [13, 14].

$$f(x)=\frac{\lambda k}{(1+\lambda x)^{k+1}}, \qquad x>0 \tag{8}$$

$$F(t)=1-(1+\lambda x)^{-k} \tag{9}$$

단, $\lambda(>0)$는 척도모수(Scale parameter)이고 $k(>0)$는 형상모수(Shape parameter)을 의미한다.

(8)과 (9)식을 이용하여 로맥스분포에 근거한 무한고장 NHPP로 접근하면 강도함수와 평균값함수는 다음과 같이 표현 할 수 있다[11].

$$\lambda(t)=h(t)=f(t)/(1-F(t))=\frac{\lambda k}{1+\lambda x} \tag{10}$$

$$m(t)=-\ln(1-F(t))=k\ln(1+\lambda x) \tag{11}$$

이경우의 로맥스모형 로그우도함수는 (7)식에 (10)식과 (11)식을 이용하면 다음과 같이 유도할 수 있다.

$$\ln L_{NHPP}(\lambda,k,|\underline{x})= n\ln\lambda+n\ln k-\sum_{i=1}^{n}\ln(1+\lambda x_i)-k\ln(1+\lambda x_n) \tag{12}$$

단, $\underline{x}=(x_1,x_2,x_3,\cdots,x_n]$

본 연구에서는 형상모수 k를 고정된 상수 $(0.5,1,1.5)$로 간주하여 모수 추정을 하고자 한다.

따라서 최우추정값 $\hat{\lambda}_{MLE}$ 는 다음 식을 만족한다.

$$\frac{\partial \ln L_{NHPP}(\lambda,k,|\underline{x})}{\partial\lambda}= \frac{n}{\lambda}-\sum_{i=1}^{n}\frac{x_i}{1+\lambda x_i}-k\left(\frac{x_n}{1+\lambda x_n}\right)=0 \tag{13}$$

4. 소프트웨어 고장시간 분석

표 1. 소프트웨어의 고장시간자료
Table 1. Failure time data of software

Failure number	Failure time (hours)	Failure number	Failure time (hours)
1	0.479	16	10.771
2	0.745	17	10.906
3	1.022	18	11.183
4	1.576	19	11.779
5	2.61	20	12.536
6	3.559	21	12.973
7	4.252	22	15.203
8	4.849	23	15.64
9	4.966	24	15.98
10	5.136	25	16.385
11	5.253	26	16.96
12	6.527	27	17.237
13	6.996	28	17.6
14	8.17	29	18.122
15	8.863	30	18.735

이 절에서는 소프트웨어 고장 시간자료[15] (Failure time data)를 이용하여 본 논문에서 제시하는 소프트웨어 신뢰모형들을 이용한 소프트웨어 고장시간을 분석하고자 한다. 이 자료의 고장시간은 18.735 시간단위에 30번의 고장이 발생된 자료이며 [표 1]에 나열 되어 있다.

또 한 제시하는 신뢰모형들을 분석하기 위하여 우선 자료에 대한 추세 검정이 선행 되어야 한다[10, 16]. 추세분석에는 일반적으로 라플라스 추세검정(Laplace trend test)을 사용한다. 이 검정을 실시한 결과 [그림 1]에서 라플라스 추세 검정의 결과는 라플라스요인(Factor)이 -2와 2사이에 존재함으로서 즉, 극단값(Extreme value)이 존재

하지 않으므로 이 자료를 이용하여 신뢰성장모형을 제시하는 것이 효율적임을 시사하고 있다.

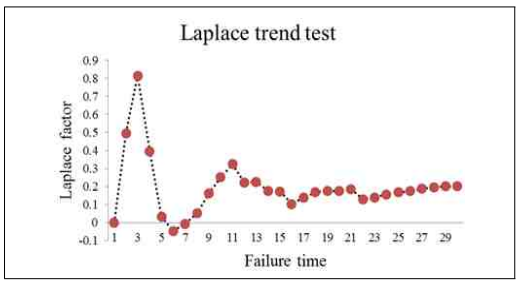

그림 1. 라플라스 추세검정
Fig. 1. Laplace trend test

모수추정은 최우추정법을 이용하고 모수추정을 용이하게 하기 위하여 원래의 고장시간 데이터를 변수변환($Failure\ time \times 10^{-1}$)하여 적용하였다. 비선형 방정식의 계산방법은 수치 해석적 기본방법인 이분법(Bisection method)을 사용하였다. 이러한 계산은 초기 값을 0.01과 3을, 허용한계(Tolerance for width of interval)는 10^{-5} 을 주고 수렴성을 확인하면서 충분한 반복횟수인 100번을 C-언어를 이용하여 모수추정을 수행하였다. 각 모형에 대한 모수추정 값은 [표 2]에 요약되었다.

표 2. 각 모형에 대한 모수추정값
Table 2. Parameter estimation of each model

Parameter	MLE
$k=30$	$\hat{\lambda}_{MLE}=0.8214$
$k=40$	$\hat{\lambda}_{MLE}=0.5606$
$k=50$	$\hat{\lambda}_{MLE}=0.4218$

Note.
MLE : Maximum likelihood estimation;

본 연구에서는 (8)식에서 다음과 같이 가정하여 비용곡선을 분석하고자 한다.

$(Assumption)$
$E_1=10\$, \quad c_2=0.7\$, \quad c_3=0.5\$, \quad c_4=1.5\$, \quad t'=1.5$

$(Assumption)$의 비용곡선은 [그림 2]에 요약되었다. 이 그림에서 모든 비용곡선은 처음에는 감소하다가 점차 증가하는 추세를 보이고 있다. 소프트웨어 시스템의 잔여 결함의 수는 결함의 제거하는 과정에서 점점 줄어들게 되고 즉, 남아 있는 결함이 관측될 확률은 낮아지게 된다. 따라서 테스트의 초기 단계에 있는 소프트웨어는 여전히 많은 오류가 있기 때문에 쉽게 감지 및 제거 단계에서 오류를 제거하는 비용은 운용 단계에서 오류를 제거하는 것보다 훨씬 낮기 때문에, 소프트웨어의 총비용은 감소한다. 그러나 일정시간 이후의 단계에서 소프트웨어에 남아있는 결함의 수는 적기 때문에 이 테스트 단계에서 결함을 검출 시간이 상대적으로 길고 오류를 제거하는 비용은 운용 단계에서 상대적으로 높기 때문에 비용 곡선은 시간이 흐름에 따라 지속적으로 증가하게 된다[11, 12].

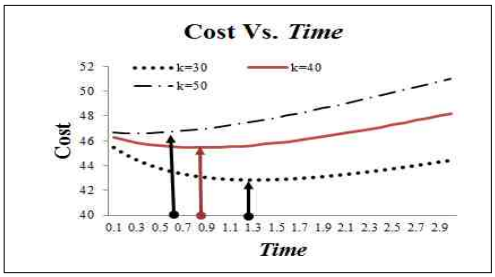

그림 2. 소프트웨어 개발 비용곡선
Fig. 2. Software development cost curve

결국 비용 곡선의 추세를 이용하여 최적의 소프트웨어 방출시간을 추정 할 수 있다. 이러한 상황은 가장 현실적인 상황이며 대부분의 경우 실제 소프트웨어 개발의 과정에서 이러한 패턴을 가지게 된다[14]

이 비용곡선에서 형상모수가 $k=50$인 경우가 $k=30$ 와 $k=40$인 경우보다 소프트웨어 방출시간은 제일 빠르다. 그러나 비용측면에서는 $k=30$인 경우가 상대적으로 다른 비교 모형에 비해 경제적임을 알 수 있다.

5. 결론

소프트웨어 신뢰도 성장 모형은 최적의 소프트웨어 방출 시간과 테스트 작업의 비용을 예상할 수 있다.

따라서 보다 효율적인 모형은 테스트 비용을 줄이고 소프트웨어를 방출 이익을 증가 시 킬 수 있도록 해야 한다.

본 연구에서 사용된 Lomax-NHPP모형에서 형상모수가 작은 경우가 소프트웨어 방출시간은 늦지만 비용측면에서는 효율적으로 나타나고 있다.

대용량 소프트웨어가 수정과 변경하는 과정에서 결함의 발생을 거의 피할 수 없는 상황이 현실이다. 신뢰성 요구를 만족하고 총비용을 최소화하는 상황이 최적방출시간이다. 경우에 따라서는 왜도와 첨도 측면에서 효율적인 카파분포, 지수화지수분포 등 업데이트된 분포에 대한 방출시기 문제를 비교 분석하는 연구도 가치 있는 일이라 판단되고 이 연구를 통하여 소프트웨어 개발자들은 최적방출시간과 경제적 개발 비용을 파악하는데 도움을 줄 수 있으리라 사료 된다.

REFERENCES

[1] Gokhale, S. S. and Trivedi, K. S. A, “time/structure based software reliability model”, Annals of Software Engineering. 8, pp. 85-121. 1999.

[2]Goel A L, Okumoto K, “Time-dependent fault detection rate model for software and other performance measures”, IEEE Trans. Reliab. 28, pp.206-11, 1978.

[3]Yamada S, Ohba H, “S-shaped software reliability modeling for software error detection", IEEE Trans. Reliab, 32, pp.475-484, 1983.

[4] Zhao M, “Change-point problems in software and hardware reliability”, Commun. Stat. Theory Methods, 22(3), pp.757-768, 1993.

[5] Shyur H-J, “A stochastic software reliability model with imperfect debugging and change-point”, J. Syst. Software 66, pp.135-141, 2003.

[6] Pham H, Zhang X., “NHPP software reliability and cost models with testing coverage", Eur. J. Oper. Res, 145, pp.445-454, 2003.

[7] Huang C-Y, “Performance analysis of software reliability growth models with testing-effort and change-point”, J. Syst. Software 76, pp. 181-194, 2005.

[8] Kuei-Chen, C., Yeu-Shiang, H., and Tzai-Zang, L., “A study of software reliability growth from the perspective of learning effects”, Reliability Engineering and System Safety 93, pp. 1410 - 1421, 2008.

[9]Hee-Cheul KIM, "The Assessing Comparative Study for Statistical Process Control of Software Reliability Model Based on Rayleigh and Burr Type", Journal of korea society of digital industry and information management, Volume 10, No.2, pp. 1-11. 2014.

[10]Tae-Hyun Yoo, “The Infinite NHPP Software Reliability Model based on Monotonic Intensity Function", Indian Journal of Science and Technology, Volume 8, No. 14, pp. 1-7, 2015.

[11]Zhang Y, Wu K. Software cost model considering reliability and time of software in use. Journal of Convergence Information Technology. 7(13), pp. 135 - 42, 2012.

[12] Kim H-C. The Property of Learning effect based on Delayed Software S-Shaped Reliability Model using Finite NHPP Software Cost Model, Indian Journal of Science and Technology 8(34), pp. 1-7, 2015.

[13]http://www.math.wm.edu/~leemis/chart/UDR/

PDFs/Lomax.pdf

[14]Satya Prasad R, Sridevi G, Kumari KS. Assessing Pareto Type II Software Reliability using SPC. International Journal of Computer Applications (0975 - 8887),62(3), pp.17-21, 2013.

[15]Y. HAYAKAWA and G. TELFAR, "Mixed Poisson-Type Processes with Application in Software Reliability", Mathematical and Computer Modelling, 31, pp. 151-156, 2000.

[16] K. Kanoun and J. C. Laprie, "Handbook of Software Reliability Engineering", M.R.Lyu, Editor, chapter Trend Analysis. McGraw-Hill New York, NY, pp. 401-437, 1996.

[17]Tae-Jin Yang, Jea-gun Park, "A Study on Software Reliability Model based on Mixture Weibull NHPP Property", International Journal of Applied Engineering Research Vol.10, No.90, pp. 548-552, 2015.

저자약력

양 태 진 (Tae-Jin Yang) **[정회원]**

- 1992년 2월 : 한양대학교 전자공학과 (공학석사)
- 1995년 2월 : 한양대학교 전자공학과 박사(수료)
- 1993년 3월 ~ 2013년 12월 : 서울호서전문학교 정보통신과 교수, HRD센터 교수부장
- 2014년 3월 ~ 현재 : 남서울대학교 산학협력단 조교수

<관심분야>

소프트웨어신뢰성 공학, 인공지능(Artificial Intelligence), Fuzzy Application & Neural-Network