Journal of Knowledge Information Technology and Systems(JKITS), Vol. 9, No. 2, pp. 295~302, April 2014

Journal of Knowledge Information Technology and Systems

ISSN 1975-7700

http://www.kkits.or.kr

A Comparative Study on Infinite NHPP Reliability Cost Model Based on Intensity Function of Log Power Form

Tae-Jin Yang[*1], **Hee-Cheul Kim**[2]

[1] *Industry-Academic Cooperation Foundation, Namseoul University*
[2] *Division of Industrial & Management Engineering, Namseoul University*

ABSTRACT

With the increasing demand to deliver high-quality software, more accurate software reliability models and software cost models are required to estimate the optimal software release time and the cost of testing efforts. In this study, reliability software cost model considering intensity function based on life distribution from the process of software product testing was studied. The comparison problem of log power intensity function that is widely used in the field of reliability, reliability growth model, was presented. The software failure pattern was used infinite failure non-homogeneous Poisson process model, the parameters estimation using maximum likelihood estimation was conducted. For analysis of software cost model considering intensity function, in this research, software developers to identify software development cost some extent be able to help. In this study, software cost model, using the infinite non-homogeneous Poisson process(NHPP) failure, more software efficiently optimum delivery times can be predicted. The proposed model is the total number of defects in software release period after the operational software and software to be discovered during maintenance and the defects are not found in all software users should assume that. It's a real step error detection and error correction cost of removing all of the remaining operational steps to eliminate the error is lower than the cost increases, the operating time can be seen as costs increase. Therefore, the optimal discharge time of the software can be realistically predict beforehand.

KEYWORDS: Cost model, Log power intensity function, NHPP, Laplace trend test, Infinite failure.

ARTICLE INFO: Received 17 March 2014, Accepted 11 April 2014.

[*]Corresponding author is with the Department of Industrial & Management Engineering, Namseoul University,91 Daehak-ro Seonghwan-eup Sebuk-gu Cheonan-si Chungcheongnam-do, 331-707, KOREA.
E-mail address: kim1458@nsu.ac.kr

1. 서 론

소프트웨어 신뢰성은 일정한 환경조건에서 일정 기간동안 고장이 나지 않고 운영 할 수 있는 확률로 시스템 신뢰도에 영향을 주는 중요한 요소가 되며 디자인 속성 측면에서는 하드웨어 신뢰성과는 다른 특성을 지닌다.

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 [2]은 고양된 비동질적인 포아송 과정 모형(Enhanced NHPP) 모형을 제시하였고 Goel 과 Okumoto [3]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 [4]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 [5]는 소프트웨어 신뢰도에서 변환점 문제를 제시하였고 Shyur [6]는 변환점을 이용한 일반화한 신뢰도 성장 모형을 제안하였다. Pham와 Zhang[7]는 테스팅 커버리지(Coverage)를 측정하여 소프트웨어 안정도를 평가 할 수 있는 소프트웨어 안정도 모형을 제시했다. 비교적 최근에, Huang [8]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[9]. 대부분의 연구가 유한고장 모형에 치중되었지만 본 연구에서는 무한고장 모형을 적용한 비용문제를 제시하고자 한다.

따라서 본 논문에서는 무한고장 NHPP에 의존한 로그 파우어 형태를 이용하고 이에 대한 비용 모형을 비교 제시하였다.

2. 관련 연구

2.1 소프트웨어 신뢰성

$N(t)$을 시간 t까지 검출된 소프트웨어의 누적고장수라고 하고, $m(t)$를 이에 대한 기대값를 나타내는 평균값 함수(MVF, Mean Value Function)로 가정하고 $\lambda(t)$을 강도함수(Intensity function) (즉, t에서의 순간 결함 검출율)이면 비동질 포아송 과정(NHPP, Non-homogeneous Poisson Processes)은 누적 고장수인 $N(t)$는 모수 $m(t)$을 가진 포아송

확률밀도함수 (Probability density function)으로 알려져 있다. 즉,

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

따라서 NHPP 모형에서 평균값 함수 $m(t)$ (Mean value function)와 강도 함수 $\lambda(t)$는 다음과 같은 관계로 표현할 수 있다[2][10].

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

이처럼 시간관련 모형(Time domain models)들은 NHPP에 의해서 확률 고장과정으로 설명이 가능하며, 유한 고장 모형과 무한 고장 범주로 분류한다[1].

유한 고장(Finite failure) NHPP 모형들은 충분한 테스트 시간이 주어지면 결함들(Faults)의 기대값이 유한값$\left(\lim_{t\to\infty} m(t)=\theta<\infty\right)$을 가지고 반면에 무한 고장(Infinite failure) NHPP 모형들은 무한값을 가진다고 가정 된다. 무한고장 NHPP 모형들은 실제 상황에서는 수리 시점에서도 고장이 발생할 상황을 반영하기 위하여 기록 멈춤 통계량(Record breaking statistics)을 사용하는 RVS(Record Value Statistics)모형을 사용할 수 있다고 하였고 이 RVS 모형과 NHPP 모형에 관해서 평균값 함수는 다음과 같이 된다고 하였다.

$$m(t)=\ -\ln(1-F(t)) \quad (3)$$

따라서 (1)식과 (3)식을 연관시키고 $f(t)$을 확률밀도함수, $F(t)$을 분포함수라고 하면 NHPP의 강도함수는 $F(t)$의 위험함수($h(t)$)가 된다.

$$\lambda(t)=m'(t)=f(t)/(1-F(t))=h(t) \quad (4)$$

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

$$x_n=\sum_{i=1}^{n} t_i\ \ (i=1,2,\cdots,n\,;0\le x_1\le x_2\le\cdots\le x_n) \quad (5)$$

이라고 하면 n번째까지 고장시점이 관찰된 고장 절단 모형일 경우에 데이터 집합 D_{x_n}은 $\{x_1,x_2\cdots x_n\}$으로 구성되며, 이 시간 절단 모형에서 Θ을 모수공간이라고 표시하면 유한고장 NHPP 모형의 우도함수는 다음과 같이 알려져 있다[11][12].

$$L_{NHPP}(\Theta|\ \underline{x}\ )=\left(\prod_{i=1}^{n}\lambda(x_i)\right)\exp[-m(x_n)] \quad (6)$$

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

2.2 소프트웨어 개발 비용모형

소프트웨어 비용 모형은 다음과 같이 정의 된다[13][14].

$$\begin{aligned}E&=E_1+E_2+E_3+E_4\\&=E_1+C_2\times t+C_3\times m(t)+C_4\times[m(t+t')-m(t)]\end{aligned} \quad (7)$$

단, E: 소프트웨어 개발 예상 총비용

E_1: 소프트웨어 설계 및 초기 소프트웨어 개발의 비용(분석 데이터, 소프트웨어 개발 전문가의 수, CPU시간 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 테스 팅 과정에서 하나의

결함을 제거하는 비용, $m(t)$는 t시점에서 탐색되어 질 수 있는 결함의 기대 수)

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'는 소프트웨어 시스템을 출시한 후 운영 및 소프트웨어를 유지할 수 있는 시간)

테스트 단계에서 소프트웨어의 결함수가 점점 오류가 제거되므로 운용 단계에서 상대적으로 결함을 검출되는 시간이 길고 오류를 제거하는 비용이 높기 때문에 현실적으로 C_4는 C_2와 C_3보다 높은 비용을 나타낸다.

그러므로 최적의 최적 소프트웨어 방출시간(t)는 다음과 같이 유도 할 수 있다.

$$\frac{\partial E}{\partial t} = E' = (E_1 + E_2 + E_{3+} E_4)' = 0 \quad (8)$$

3. 제안한 로그 파우어 강도함수를 가진 무한고장 NHPP 신뢰성장모형을 이용한 소프트웨어 개발 비용모형

기존의 연구는 강도함수의 패턴이 증가하거나 감소하는 패턴을 가진 연구가 대부분 이었다. 그러나 현실적으로는 증가하다가 감소하가나 혹은 감소하다가 증가 할 수 있다. 본 연구에서는 감소하다가 증가하는 형태를 로그 파우어 강도함수를 이용한 소프트웨어 개발 비용모형에 대하여 그 특성을 알아보고자 한다.

소프트웨어 신뢰성 분야에서 많이 사용되는 로그 파우어 강도함수는 각 각 다음과 같이 알려져 있다[15].

$$\lambda_{Log\ power} = \frac{a\, b \ln^{b-1}(1+t)}{1+t} \quad (9)$$

단, a와 b는 각 각 척도모수와 형상모수를 의미한다.

(2)식과 (9)식을 이용하면 평균값 함수는 다음과 같이 유도 할 수 있다.

$$m_{Log\ power}(t) = a \ln^{b}(1+t) \quad (10)$$

유사하게 로그 파우어 모형에 대해서는 (5)식에 (9), (10)식을 대입한 우도함수를 이용한 모수 추정은 다음과 같은 식을 만족한다.

$$\hat{a} = \frac{n}{\ln^{\hat{b}}(1+x_n)} \quad (11)$$

$$\frac{n}{\hat{b}} = \ln\left(\sum_{i=1}^{n} \ln(1+x_i)\right) + \hat{a} \ln^{\hat{b}}(1+x_n) \ln(\ln(1+x_n)) \quad (12)$$

4. 소프트웨어 고장 자료 및 비용 분석

이 절에서는 소프트웨어 고장 시간 자료[16](Failure interval time data)를 이용하여 본 논문에서 제시하는 소프트웨어 신뢰모형들의 개발비용을 분석하고자 한다. 이 자료의 고장 시간은 18.735 시간단위에 30번의 고장이 발생된 자료이며 <표 1>에 나열 되어 있다.

또한 제시하는 신뢰 모형들을 분석하기 위하여 우선 자료에 대한 추세 검정이 선행 되어야 한다[19]. 추세 분석에는 일반적으로 라플라스 추세 검정(Laplace trend test)을 사용한다. 이 검정을 실시한 결과 <그림 1>에서 라플라스 추세 검정의 결과는 라플라스 요인(Factor)이 -2와 2사이에 존재함으로서 즉, 극단값(Extreme value)이 존재하지 않으므로 이 자료를 이용하여 신뢰 성장모형을 제시하는 것이 효율적임을 시사하고 있다[17][18].

표 1. 고장 시간 자료
Table 1. Failure time data

Failure number	Failure time (hours)	$Failure\ time \times 10^{-1}$ (hours)	Failure interval (hours)
1	0.479	0.0479	0.0479
2	0.745	0.0745	0.0266
3	1.022	0.1022	0.0756
4	1.576	0.1576	0.082
5	2.61	0.261	0.179
6	3.559	0.3559	0.1769
7	4.252	0.4252	0.2483
8	4.849	0.4849	0.2366
9	4.966	0.4966	0.26
10	5.136	0.5136	0.2536
11	5.253	0.5253	0.2717
12	6.527	0.6527	0.381
13	6.996	0.6996	0.3186
14	8.17	0.817	0.4984
15	8.863	0.8863	0.3879
16	10.771	1.0771	0.6892
17	10.906	1.0906	0.4014
18	11.183	1.1183	0.7169
19	11.779	1.1779	0.461
20	12.536	1.2536	0.7926
21	12.973	1.2973	0.5047
22	15.203	1.5203	1.0156
23	15.64	1.564	0.5484
24	15.98	1.598	1.0496
25	16.385	1.6385	0.5889
26	16.96	1.696	1.1071
27	17.237	1.7237	0.6166
28	17.6	1.76	1.1434
29	18.122	1.8122	0.6688
30	18.735	1.8735	1.2047

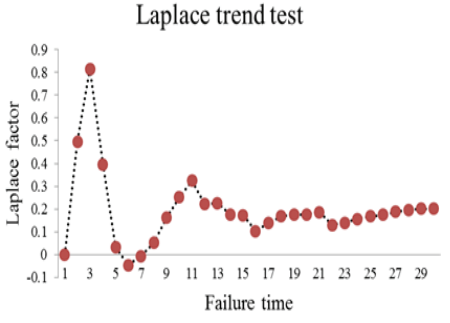

그림 1. 라플라스 추세 검정
Figure 1. Laplace trend test

모수 추정은 최우추정법을 이용하고 모수 추정을 용이하게 하기 위하여 원래의 고장시간 데이터를 변수변환($Failure\ time \times 10^{-1}$)하여 적용하였다. 비선형 방정식의 계산방법은 수치 해석적 기본 방법인 이분법(Bisection method)을 사용하였다. 그리고 이러한 계산은 초기 값을 0.0001와 3을, 허용 한계(Tolerance for width of interval)는 10^{-5} 을 주고 수렴성을 확인 하면서 충분한 반복 횟수인 100번을 C-언어를 이용하여 모수 추정을 수행하였다. 그 결과 $\hat{a}_{MLE} = 20.3058$, $\hat{b}_{MLE} = 1.4775$으로 추정 되었다.

본 연구에서는 (7)식에서 다음과 같이 가정하여 비용 곡선을 분석하고자 한다.

(가정) A

$E_1 = 10\$,\ c_2 = 1\$,\ c_3 = 0.1\$,\ c_4 = 3\$,\ t' = 10$

(가정) B

$E_1 = 10\$,\ c_2 = 1\$,\ c_3 = 0.1\$,\ c_4 = 10\$,\ t' = 10$

(가정) C

$E_1 = 10\$,\ c_2 = 1\$,\ c_3 = 0.1\$,\ c_4 = 3\$,\ t' = 100$

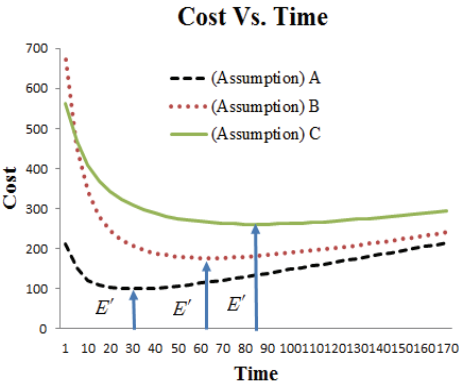

그림 2. 가정된 비용 곡선
Figure 2. The cost curve under the condition of (Assumption)

<그림 2>에서 보는 바와 같이 제안된 모델의 비용 성장 곡선은 처음에는 감소하다가 점차 증가하는 추세를 보이고 있다. 소프트웨어 시스템의 잔여 결함의 수는 결함의 제거하는 과정에서 점점 줄어

들게 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 후반부의 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[13].

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].

(가정) A에서 다른 가정은 동일하고 소프트웨어 출시 된 이후에 소프트웨어 운영 단계에서 사용자가 관찰되는 결함수정 비용 C_4를 $c_4 = 3\$$에서 $c_4 = 10\$$으로 증가한 경우는 <그림 2>에 $(Assumption)B$ 곡선을 의미한다.

곡선 $(Assumption)A$와 $(Assumption)B$을 비교 하였을 때 소프트웨어 출시 된 이후에 소프트웨어 운영 단계에서 사용자가 관찰되는 결함수정 비용 (C_4)이 증가하였기 때문에 소프트웨어 최적 방출시간이 연기됨을 알 수 있다. 따라서 이 경우에는 소프트웨어 방출시기 이후에 결함을 감소시킬 수 있도록 운영단계보다 테스팅 단계에서 가능한 결함들을 제거해야 한다,

(가정) A에서 다른 가정은 동일하고 소프트웨어 시스템을 출시 한 후 운영 및 소프트웨어를 유지할 수 있는 시간 t'를 $t' = 10$ 에서 $t' = 100$으로 증가한 경우는 <그림 2>에서 $(Assumption)C$ 곡선을 의미한다.

곡선 $(Assumption)A$와 $(Assumption)C$을 비교 하였을 때 소프트웨어 시스템을 출시 한 후 운영 및 소프트웨어를 유지할 수 있는 시간이 증가하였기 때문에 역시 소프트웨어 최적 방출시간이 연기됨을 알 수 있다. 따라서 이 경우에도 소프트웨어 방출시기 이후에 결함을 감소시킬 수 있도록 운영단계보다 테스팅 단계에서 가능한 결함들을 제거해야 한다.

5. 결 론

소프트웨어 신뢰도 성장 모델은 최적의 소프트웨어 방출 시간과 테스트 작업의 비용을 예상 할 수 있다 [15].

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

본 연구에서 사용된 무한고장 NHPP모형을 이용한 소프트웨어 비용 모델은 보다 효율적으로 최적의 소프트웨어 방출 시간을 예측 할 수 있다. 제안된 모델의 결함의 총수는 소프트웨어 방출이후 소프트웨어 운용기간 및 소프트웨어를 유지하는 동안 발견된 수이고 이 결함들은 소프트웨어 사용자가 전부 다 발견되지 않을 것이라는 가정을 한다. 그것은 실제 오류를 감지 및 제거 단계에서 오류수정 비용은 운용 단계에서 남아있는 모든 오류를 제거하는 비용보다 낮지만 운영 시간이 증가함에 따라 비용이 증가함을 알 수 있다. 따라서 최적의 소프트웨어 방출 시간은 현실적으로 미리 예측해 볼 수 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 방출 시기 문제를 비교 분석하는 연구도 가치 있는 일이라 판단된다. 따라서 최적방출시간은 운용단계에서 결함비용을 줄이고 유지기간을 증가시키는 시점을 만족해야 한다. 이 연구를 통하여 소프트웨어 개발자들은 최적방출시간과 경제적 개발 비용을 파악 하는데 도움을 줄 수 있으리라 사료 된다.

References

[1] L. Kuo, and T. Y. Yang, *Bayesian computation of software reliability,* Journal of the American Statistical Association, Vol. 91, pp. 763-773, 1996.

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

[3] Goel A L, and Okumoto K, *Time-dependent fault detection rate model for software and other performance measures*, IEEE Transaction Reliability, 28, pp. 206-11, 1978.

[4] Yamada S, and Ohba H, *S-shaped software reliability modeling for software error detection*, IEEE Transaction Reliability, 32, pp. 475-484, 1983.

[5] Zhao M, *Change-point problems in software and hardware reliability*, Communications in Statistics-Theory Methods, 22(3), pp. 757-768, 1993.

[6] Shyur H-J, *A stochastic software reliability model with imperfect debugging and change-point*, Journal of Systems and Software 66, pp. 135-141, 2003.

[7] Pham H, and Zhang X, *NHPP software reliability and cost models with testing coverage,* European Journal of Operational Research, 145, pp. 445-454, 2003.

[8] Huang C-Y, *Performance analysis of software reliability growth models with testing-effort and change-point,* Journal of Systems and Software 76, pp. 181-194, 2005.

[9] 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.

[10] Hee-Cheul KIM, *The comparative study of NHPP half-logistic distribution software reliability model using the perspective of learning effects,* JNIT: Journal of Next Generation Information Technology, Vol. 4, No. 8, pp. 132-139, 2013.

[11] Hyun-Dai SHIN, and Hee-Cheul KIM, *The comparative study of software optimal release time based on NHPP software reliability model using exponential and log shaped type for the perspective of learning effects*, IJACT: International Journal of Advancements in Computing Technology, Vol. 5, No. 12, pp. 120-129, 2013.

[12] Hee-Cheul KIM, *The comparative study of NHPP delayed s-shaped and extreme value distribution software reliability model using the perspective of learning effects,* IJACT: International Journal of Advancements in Computing Technology, Vol. 5, No. 9, pp. 1210-1218, 2013.

[13] Kyung-Soo Kim, and Hee-Cheul Kim, *The comparative software cost model of considering logarithmic fault detection rate based on failure observation time,* The Journal of Digital Policy & Management, 11(11), pp. 335-342, Nov. 2013.

[14] Ye Zhang, and Kaigui Wu, *Software cost model considering reliability and time of software in Use,* JCIT: Journal of

Convergence Information Technology, Vol. 7, No. 13, pp. 135-142, 2012.

[15] Vincent Almering, Michiel van Genuchten, and Ger Cloudt, Peter J.M. Sonnemans, *Software reliability growth models in practice,* IEEE SOFTWARE, pp. 82-88, 2007.

[16] Y. HAYAKAWA and G. TELFAR, *Mixed poisson-type processes with application in software reliability,* Mathematical and Computer Modelling, 31, pp. 151-156, 2000.

[17] Kyung-Soo Kim, and Hee-Cheul Kim, *The comparative study of software optimal release time of finite NHPP model considering property of nonlinear intensity function,* The Journal of Digital Policy & Management, 11(1): pp. 159-166, Sep. 2013.

[18] 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.

로그 파우어 강도함수에 근거한 소프트웨어 무한고장 NHPP 신뢰성 비용 모형에 관한 비교 연구

양태진[1], 김희철[2]

[1] *남서울대학교 산학협력단*

[2] *남서울대학교 산업경영공학과*

요 약

본 연구에서는 소프트웨어 제품 테스팅 과정에서 고장 수명분포의 강도함수를 고려한 소프트웨어 신뢰성 비용모형에 대하여 연구하였다. 소프트웨어 신뢰성 분야에서 많이 사용되는 로그 파우어 강도함수를 가진 소프트웨어 신뢰성장 모형에 대한 비교문제를 제시하였다. 소프트웨어 고장형태는 무한고장 비동질적인 포아송과정을 이용하고 모수추정법은 최우추정법을 이용하였다. 따라서 본 논문에서는 강도함수를 고려한 소프트웨어 비용모형 분석을 위하여 소프트웨어 고장시간자료를 적용하여 비교 분석하였다. 이 연구를 통하여 소프트웨어 개발자들은 소프트웨어 개발 비용을 파악 하는데 도움을 줄 수 있을 것으로 사료된다.

Tae Jin Yang received the M.S. degree and finished the coursework in the Department of Electronic Engineering from Hanyang University in 1992 and 1995, respectively. He has been a professor in the Industry-Academic Cooperation Foundation at Namseoul University since 2014. His current research interests include Fuzzy & Neural-Network application and software reliability. He is a membership of the KKITS.

E-mail address: solomon645@nsu.ac.kr

Hee Cheul Kim received the M.S. degree and the Ph.D. degree in the Department of Statistic from Dongguk University in 1992 and 1998, respectively. He has been a professor in the Department of Industrial & Management Engineering at Namseoul University since 2005. His current research interests include software reliability engineering, computer statistics, information systems. He is a membership of the KKITS.

E-mail address: kim1458@nsu.ac.kr