디지털산업정보학회 논문지 제10권 제4호-2014년 12월

로그형 평균값함수를 고려한 소프트웨어 신뢰성모형에 대한 비교연구

신 현 철* · 김 희 철**

A Comparative Study of Software Reliability Model Considering Log Type Mean Value Function

Shin, Hyun Cheul · Kim, Hee Cheul

〈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 paper, proposes the reliability model with log type mean value function (Musa-Okumoto and log power model), which made out efficiency application for software reliability. Algorithm to estimate the parameters used to maximum likelihood estimator and bisection method, model selection based on mean square error (MSE) and coefficient of determination(R^2), for the sake of efficient model, was employed. Analysis of failure using real data set for the sake of proposing log type mean value function was employed. This analysis of failure data compared with log type mean value function. In order to insurance for the reliability of data, Laplace trend test was employed. In this study, the log type model is also efficient in terms of reliability because it (the coefficient of determination is 70% or more) in the field of the conventional model can be used as an alternative could be confirmed.

From this paper, software developers have to consider the growth model by prior knowledge of the software to identify failure modes which can be able to help.

Key Words : Software Reliability, Non-Homogeneous Poisson Process, Log Type Mean Value Function

Ⅰ. 서론

현대사회에 있어서 인간이 의존하고 있는 시스템의 대부분은 소프트웨어 시스템(System)을 포함하고 있으며 이러한 시스템의 기능이 확대되어 소프트웨어가 대규모화, 복잡화, 다양화되고 수요량도 급속히 증대하고 있다. 그러나 시스템이 고장(Failure)이 발생하면 고장의 원인이 되는 결함 (Fault)을 찾아 제거

* 백석문화대학교 인터넷 정보학부 교수
** 남서울대학교 산업경영공학과 교수(교신저자)

하는 디버깅과정을 거치면서 소프트웨어에 잔존하는 결함의 수는 점차 감소하며 고장이 발생하는 시간간격이 점차 증가함으로 소프트웨어 신뢰성은 성장한다.

소프트웨어 테스팅단계에서 소프트웨어 고장수(Number of failure)와 고장간격시간에 의해 소프트웨어 고장현상을 수리적으로 모형화하면 소프트웨어에 대한 평가를 쉽게 할 수 있으며 신뢰도모형에 의해 소프트웨어 고장수, 소프트웨어 고장발생간격시간, 소프트웨어 신뢰도 및 고장률 등의 신뢰성 평가 측도들이 추정되어 미래의 고장시간을 예측할 수도 있다.

소프트웨어 고장시간은 수명자료가 된다. 그러므로 비음(Nonnegative)의 값을 가지기 때문에 수명분포가 한정되어 있다. 시간이 지날수록 비증가 패턴을 가지는 지수(Exponential)분포나, 혹은 와이블(Weibull), 로그정규(Lognormal), 감마분포(Gamma distribution)등을 사용하여 모형을 적용하는 것이 일반적 현상이다. 이러한 분포들은 일반화 감마분포(Generalized gamma distribution)의 특수한 경우로 알려져 있다[1]. 지금까지 많은 소프트웨어 신뢰성모형이 제안되었다. 이 중에서 비동질적 포아송과정(Non-Homogenous Poisson Process; NHPP)에 의존한 모형은 에러 탐색과정 측면에서는 우수한 모형[2]이고, 결함이 발생하면 즉시 제거되고 디버깅과정에서 새로운 결함이 발생되지 않는다는 가정을 하고 있다.

Gokhale과 Trivedi[3]은 고양된 비동질적인 포아송과정모형(Enhanced NHPP) 모형을 제시하였고 Goel과 Okumoto[4]은 지수적 소프트웨어 신뢰성모형(Exponential software reliability growth model)을 제안 하였다. 이 모형은 결함의 누적수가 S-형태나 지수적 형태(S-shaped or exponential-shaped)를 가진 평균값함수(Mean value function)을 이용하였다. 이러한 모형에 의존한 일반화모형은 Yamada와 Ohba[5]에 의해 지연된 S형태 신뢰성장모형(Delayed S-shaped reliability growth model)과 변곡된 S 형태 신뢰성장모형(Inflection S-shaped reliability growth model)이 제안되었다. Zhao[6]는 소프트웨어 신뢰도에서 변환점 문제를 제시하였고 Shyur[7]는 변환점을 이용한 일반화한 신뢰도 성장모형을 제안하였다. Pham와 Zhang[8]는 테스팅 커버리지(Coverage)를 측정하여 소프트웨어 안정도를 평가 할 수 있는 소프트웨어 안정도 모형을 제시했다. 비교적 최근 Huang[9]은 일반화 로지스틱 테스팅 노력함수(Generalized logistic testing-effort function)와 변환점 모수(Change-point parameter)을 통합하여 효율적인 소프트웨어 신뢰성을 예측하는 기술을 제시하기도 하였다. 또한 소프트웨어 공정관리측면에 대한 연구인 Rayleigh형과 Burr형 NHPP 소프트웨어 신뢰모형에 관한 통계적 공정관리 접근방법 비교도 이루어졌다[10].

본 연구에서는 NHPP 모형에서 비감소 패턴을 가지는 로그형 평균값함수를 가지는 모형을 비교 분석하고자 한다.

Ⅱ. 관련연구

2.1 소프트웨어 신뢰성

$N(t)$을 시간 t까지 검출된 소프트웨어의 누적고장수라고 하고, $m(t)$를 이에 대한 기대값을 나타내는 평균값함수(Mean Value Function)로 가정하고 $\lambda(t)$을 강도함수(Intensity function) (즉, t에서의 순간 결함 검출율)이면 비동질 포아송과정을 적용 할 수 있다. 따라서 누적 고장수인 $N(t)$는 모수 $m(t)$을 가진 포아송 확률밀도함수 (Probability density function)

로 알려져 있다[11-12]. 즉,

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

그리고 강도함수와 평균값함수는 다음과 같은 관계가 있다[12].

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

이처럼 시간관련모형(Time domain models)들은 NHPP에 의해서 확률고장과정으로 설명이 가능하다. 이러한 NHPP 모형들은 유한고장과 무한고장 범주로 분류한다[13].

무한고장 NHPP 모형들은 실제상황에서는 수리시점에서도 고장이 발생할 상황을 반영하기 위하여 기록멈춤 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 다음과 같이 위험함수($h(t)$)가 된다.

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

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

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

이라고 하면 n번째까지 고장시점이 관찰된 고장절단 모형일 경우에 데이터 집합 D_{x_n}은 $\{x_1,x_2\cdots x_n\}$으로 구성되며, 이 시간절단 모형에서 Θ을 모수공간이라고 표시하면 유한고장 NHPP모형의 우도함수는 다음과 같이 알려져 있다[12-13].

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

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

NHPP 모형에서 테스트 시점 x_n에서 소프트웨어 고장이 일어난다고 하는 가정 하에서 신뢰구간 $(x_n,x_n+\tau]$ (단, τ는 임무시간(Mission time))사이에서 소프트웨어의 고장이 일어나지 않을 확률인 신뢰도(Reliability) $\hat{R}(\tau\mid x_n)$는 다음과 같이 됨이 알려져 있다[12].

$$\hat{R}(\tau\mid x_n)=e^{-\int_{x_n}^{x_n+\tau}\lambda(x)dx}=\exp[-\{m(\tau+x_n)-m(x_n)\}] \quad (7)$$

2.2 효율적 모형을 위한 모형의 비교

최근 모형에 대한 효율성을 조사하기 위한 기준으로서 평균제곱오차(Mean square error; MSE)와 결정계수(Coefficient of determination; R^2)를 사용한다[14-15].

2.2.1 평균제곱오차

평균제곱오차는 실제 관찰 값과 예측 값에 대한 차이를 측정하는 도구로서 다음과 같이 정의된다.

$$MSE=\frac{\sum_{i=1}^{n}\left(m(x_i)-\hat{m}(x_i)\right)^2}{n-k} \quad (8)$$

단, $m(x_i)$은 시간$(0,\ x_i]$까지 나타난 고장들의 누적개수를 의미하고 $\hat{m}(x_i)$는 x_i시점까지 평균값함수로부터 추정된 고장의 누적개수를 의미한다. 그리고 n은 관찰값의 수이고 k는 모수의 수를 의미한다. 즉, 작은 평균제곱오차 값을 가진 모형은 효율적인 모형이 된다.

2.2.2 결정계수

결정계수는 예측값의 차이에 대한 설명력을 나타내는 도구로서 다음과 같이 정의된다. 따라서 보다 큰 결정계수를 가진 모형은 효율적인 모형으로 간주된다.

$$R^2=1-\frac{\sum_{i=1}^{n}\left(m(x_i)-\hat{m}(x_i)\right)^2}{\sum_{i=1}^{n}\left(m(x_i)-\sum_{j=1}^{n}m(x_j)/n\right)^2} \quad (9)$$

Ⅲ. 제안된 로그형 평균값함수를 가진 무한고장 NHPP 모형

3.1 Musa-Okumoto모형

Musa-Okumoto 모형[16-17]의 평균값 함수는 다음과 같이 로그형 함수로 알려져 있다.

$$m_{M-O}(t)=a\ln(1+bt) \quad (10)$$

단, $t\in[0,\infty]$ 이고 $a>0$는 척도모수(Scale parameter)이고 $b>0$는 형상모수(Shape parameter)를 의미하고 (4)식을 이용한 강도함수는 다음과 같이 유도된다.

$$\lambda_{M-O}(t)=m'_{M-O}(t)=\frac{ab}{(1+bt)} \quad (11)$$

(6)식을 이용하면 우도함수는 다음과 같이 유도된다.

$$L_{NHPP}(\Theta|\underline{x}) = \left(\prod_{i=1}^{n}\frac{ab}{(1+bx_i)}\right)\exp[-a\ln(1+bx_n)] \quad (12)$$

단, Θ는 모수공간이고 $\underline{x}=(x_1,x_2,x_3,\cdots,x_n]$.
모수추정방법은 최우추정법을 사용하였고 이러한 최우추정법을 이용하기 위한 로그우도함수는 (12)식과 관련하여 다음과 같이 유도된다.

$$\ln L_{NHPP}(\Theta|\underline{x}) = n\ln a+n\ln b-\sum_{i=1}^{n}\ln(1+bx_i)-a\ln(1+bx_n) \quad (13)$$

(13)식에서 a와 b에 대하여 편미분 하여 다음과 같은 식을 만족하는 $\hat{a}_{MLE}$ 와 $\hat{b}_{MLE}$을 수치 해석적 방법으로 계산할 수 있다.

$$\frac{\partial\ln L_{NHPP}(\Theta|\underline{x})}{\partial a}=\frac{n}{a}-\ln(1+bx_n)=0 \quad (14)$$

즉, $\hat{a}=\frac{n}{\ln(1+\hat{b}x_n)}$

$$\frac{\partial\ln L_{NHPP}(\Theta|\underline{x})}{\partial b}=\frac{n}{\hat{b}}-\sum_{i=1}^{n}\frac{x_i}{1+\hat{b}x_i}-\frac{\hat{a}\hat{b}}{1+\hat{b}x_n}=0 \quad (15)$$

단, $\underline{x}=(x_1,x_2,x_3,\cdots,x_n]$
또한, (6)식을 이용한 신뢰도는 다음과 같이 유도 된다.

$$\hat{R}(\tau|x_n) = \exp\left[-m(\tau+x_n)+m(x_n)\right] \quad (16)$$

단, τ는 임무시간(Mission time)

$m(\tau+x_n) = a\ln(1+b(x_n+\tau))$

$m(x_n) = a\ln(1+bx_n)$

3.2 Log power모형

소프트웨어 신뢰성 분야에서 많이 사용되는 로그 파우어 강도함수는 다음과 같이 알려져 있다[18-19].

$$\lambda_{L-P}(t) = \frac{a\,b\ln^{b-1}(1+t)}{1+t} \quad (17)$$

단, a와 b는 각 각 척도모수와 형상모수를 의미한다. (2)식과 (17)식을 이용하면 평균값함수는 다음과 같이 로그형으로 유도 된다.

$$m_{L-P}(t) = a\ln^{b}(1+t) \quad (18)$$

유사하게 로그 파우어모형에 대해서도 (6)식에 (17), (18)식을 대입한 우도함수를 이용한 모수추정은 다음과 같은 식을 만족한다.

$$\hat{a} = \frac{n}{\ln^{\hat{b}}(1+x_n)} \quad (19)$$

$$\frac{n}{\hat{b}} = \ln\left(\sum_{i=1}^{n}\ln(1+x_i)\right) + \hat{a}\ln^{\hat{b}}(1+x_n)\ln(\ln(1+x_n)) \quad (20)$$

Ⅳ. 소프트웨어 고장자료 분석

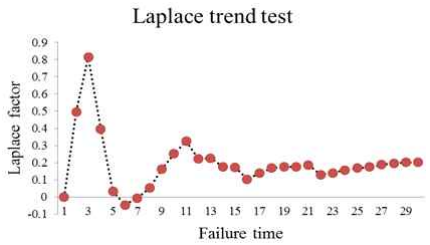
이 장에서 소프트웨어 고장시간 자료[20] (Failure time data)을 가지고 제시하는 신뢰모형들을 분석하고자 한다. 이 자료의 고장시간은 18. 735시간단위에 30번의 고장이 발생된 자료이며 <표 1>에 나열 되어 있다.

또한 제시하는 신뢰모형들을 분석하기 위하여 우선 적용자료에 대한 추세검정이 선행 되어야 한다[21].

<표 2> 유스케이스와 시스템 인터페이스 간의 매핑

Failure number	Failure time (hours)	Failure time $\times 10^{-1}$
1	0.479	0.0479
2	0.745	‘0.0745
3	1.022	0.1022
4	1.576	0.1576
5	2.61	0.261
6	3.559	0.3559
7	4.252	0.4252
8	4.849	0.4849
9	4.966	0.4966
10	5.136	0.5136
11	5.253	0.5253
12	6.527	0.6527
13	6.996	0.6996
14	8.17	0.817
15	8.863	0.8863
16	10.771	1.0771
17	10.906	1.0906
18	11.183	1.1183
19	11.779	1.1779
20	12.536	1.2536
21	12.973	1.2973
22	15.203	1.5203
23	15.64	1.564
24	15.98	1.598
25	16.385	1.6385
26	16.96	1.696
27	17.237	1.7237
28	17.6	1.76
29	18.122	1.8122
30	18.735	1.8735

추세분석에는 일반적으로 라플라스 추세검정(Laplace trend test)을 사용한다. 이 검정을 실시한 결과 <그림 1>에서 라플라스 추세검정의 결과는 라플라스 요인(Factor)이 -2와 2사이에 분포함으로서(즉, 극단값이 존재하지 않음) 신뢰성장(Reliability growth) 속성을 나타내고 있다. 따라서 이 자료를 이용하여 신뢰성장모형을 제시하는 것이 효율적임을 시사하고 있다[21].



<그림 1> 라플라스 추세 검정

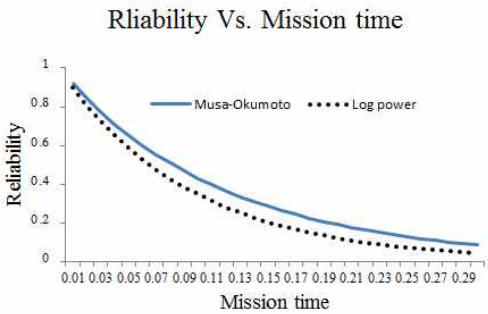
<표 2> 모수 추정치 및 MSE, R^2

Model	MLE	Model Comparison	
		MSE	R^2
Musa-Okumoto	$\hat{a}_{MLE}=21.3144$, $\hat{b}_{MLE}=1.6471$	12.0373	0.850
Log power	$\hat{a}_{MLE}=20.3058$, $\hat{b}_{MLE}=1.4775$	23.9879	0.701

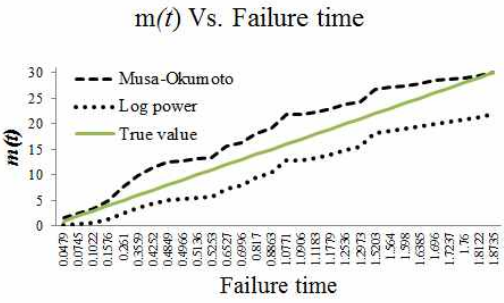
Note. MLE : Maximum likelihood estimation;
MSE : Mean square error;
R^2 :Coefficient of determination

모수추정은 최우추정법을 이용하고 모수의 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. 이러한 계산은 초기 값을 0.01과 3을, 허용 한계(Tolerance for width of interval)는 10^{-5} 을 주고 수렴성을 확인하면서 충분한 반복 횟수인 100번을 $C-$언어를 이용하여 모수추정을 수행하였다. 최우추정법의 결과와 모형에 대한 효율성을 조사하기 위한 기준으로서 MSE(평균제곱오차)와 R^2(결정계수)가 <표 2>에 요약되었다. 이 표에서 Musa-Okumo 모형이 Log-power 모형보다 실제값과 예측값에 대한 차이를 측정하는 평균제곱오차가 제일 낮고 예측값의 차이에 대한 설명력을 의미하는 설명력도 제일 높게 나타나기 때문에 Musa-Okumo 모형이 Log-power 모형보다 효율적인 모형으로 간주할 수 있다.

또한 <그림 2> 에서는 임무시간에 대한 신뢰도 그림에서도 Musa-Okumo 모형과 Log-power 모형을 비교 했을 때 임무시간이 증가 할수록 Log-power 모형보다는 Musa-Okumo 모형이 신뢰도가 상승하는 경향이 있다.



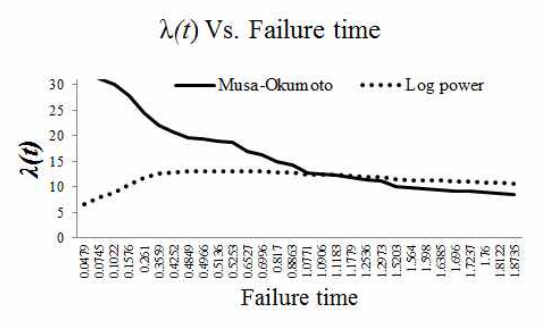
<그림 2> 각 모형에 대한 신뢰도



<그림 3> 평균값함수 패턴

<그림 3>은 Musa-Okumo 모형과 Log-power 모형에

대한 평균값 함수의 값을 나타내었다. 이 그림에서 Musa-Okumo 모형은 참값과의 차이에서 과대평가 추정이 이루어졌고 Log-power 모형은 과소평가 추정 되었다.



<그림 4> 강도함수 패턴

<그림 4>는 Musa-Okumo모형과 Log-power모형에 대한 강도(위험)함수의 값을 나타내었다. 이 그림에서 Musa-Okumo모형은 감소패턴을 Log-power모형은 증가하다가 근소하게 감소하는 패턴을 보이고 있다. Musa-Okumo모형과 Log-power모형의 효율성 비교에 있어서 Log-power모형 보다는 Musa-Okumo모형이 상대적으로 효율적이다. 그 근거는 첫째 Musa-Okumo모형이 Log-power모형보다 평균제곱오차는 낮게 나타나고 결정계수는 높게 나타났기 때문에 Musa-Okumo모형이 Log-power모형보다 효율적인 모형으로 간주할 수 있다(<표 2>). 둘째 신뢰도 측면에서 임무시간에 대한 신뢰도 그림에서도 Log-power 모형과 Musa-Okumo모형을 비교 했을 때 임무시간이 증가 할수록 Log-power모형 보다는 Musa-Okumo 모형이 신뢰도가 높게 나타나고 있다(<그림 2>).

셋째 Musa-Okumo모형과 Log-power모형에 대한 강도(위험)함수의 패턴에 있어서도 Musa-Okumo모형은 감소패턴을 Log-power모형은 증가하다가 감소하는 패턴을 보이고 있다(<그림 4>).

따라서 소프트웨어 개발 시 소프트웨어 고장현상을 사전정보로 활용하면 소프트웨어 고장품질을 향상 시킬 수 있다.

Ⅴ. 결론

대용량 소프트웨어가 수정과 변경하는 과정에서 결점의 발생을 거의 피 할 수 없는 상황이 현실이다. 소프트웨어 신뢰성은 개발의 최종단계에 있는 테스트 공정이나 실제 사용단계에 있어서 소프트웨어 내에 존재하는 고장 수나 고장 발생시간에 의해서 효과적으로 평가할 수 있는 상황으로 그 평가기술이 중요하게 된다. 따라서 소프트웨어 개발의 테스트공정이나 실제사용단계에 있어서 고장 발생환경이나 고장 발생현상을 수리적으로 모형화가 가능하면 평가를 할 수 있다.

테스트시간이나 혹은 실행시간, 발생된 고장 수와 고장 발생시간 과의 관계를 효율적으로 관리함으로서 소프트웨어 신뢰도를 성장 시 킬 수 있다. 이러한 과정을 소프트웨어 성장과정이라고 볼 수 있다.

따라서 본 연구에서는 소프트웨어 관리자들이 소프트웨어 고장원인 및 검사 도구에 활용 할 수 있는 평균값함수가 로그형을 가지는 Musa-Okumo모형과 Log-power모형에 대하여 비교 연구 하였다.

그 결과 Log-power모형 보다는 Musa-Okumo모형이 상대적으로 효율적인 모형으로 나타났다.

따라서 본 연구에 제안된 평균값함수가 로그형을 가진 신뢰성모형도 신뢰성 측면에서 효율적이기 때문에(결정계수가 70% 이상) 이 분야에서 기존의 모형의 하나의 대안으로 사용할 수 있음을 확인 할 수 있었다.

이 연구를 통하여 소프트웨어 개발자들은 다양한

평균값함수를 고려함으로서 소프트웨어 고장형태에 대한 사전지식을 파악하는데 어느 정도 도움을 줄 수 있으리라 사료 된다.

참고문헌

[1] Musa, J. D, Iannino, A. and Okumoto, K. "Software Reliability: Measurement, Prediction, Application," McGraw Hill, New York, 1987, pp. 289-291.

[2] L. Kuo, and T. Y. Yang, Bayesian Computation of Software Reliability, Journal of the American Statistical Association, Vol. 91, 1996, pp. 763-773.

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

[4] Goel AL, Okumoto K, "Time-dependent fault detection rate model for software and other performance measures," IEEE Trans Reliab. 28, pp. 206-11, 1978.

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

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

[7] Shyur H-J. "A stochastic software reliability model with imperfect debugging and change-point," J.syst. Software 66, 2003, pp. 135-141.

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

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

[10] 김희철, "Rayleigh형과 Burr형 NHPP 소프트웨어 신뢰모형에 관한 통계적 공정관리 접근방법 비교연구," 디지털산업정보학회논문지, 제10권, 제2호, 2014, pp. 1-11.

[11] 김희철, "NHPP 극값 분포 소프트웨어 신뢰모형에 대한 학습효과 기법 연구," 디지털산업정보학회 논문지, 제 7권, 제 2호, 2011, pp. 1-8.

[12] Hee-Cheul KIM, The Comparative Study of NHPP Half-Logistic Distribution Software Reliability Model using the Perspective of Learning Effects, Journal of Next Generation Information Technology, Vol. 4, No. 8, 2013, pp. 132-139.

[13] Kuo, L. and Yang, T. Y, "Bayesian Computation of Software Reliability," Journal of the American Statistical Association, Vol. 91, 1996, pp. 763-773.

[14] 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, 2008, pp. 1410-1421.

[15] Hee-Cheul KIM, The Comparative Study of NHPP Half-Logistic Distribution Software Reliability Model using the Perspective of Learning Effects, Journal of Next Generation Information Technology, Vol. 4, No. 8, pp. 2013, pp. 132-139.

[16] J. D. Musa, and K. Okumoto, "A logarithmic

Poisson execution time model for software reliability measurement," Proc. 7th Inf. Conf. on Software Engineering, 23@238, 1984.

[17] K. Venkata Subba Reddy and Dr. B. Raveendra Babu, "A Log Based Approach for Software Reliability Modeling," International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 2, February 2014, pp. 49-51.

[18] Vincent Almering, Michiel van Genuchten, and Ger Cloudt, Peter J. M. Sonnemans, Software Reliability Growth Models in Practice, IEEE SOFTWARE, 2007, pp. 82-887.

[19] Tae-Jin Yang and Hee-Cheul Kim, "The Comparative Software Infinite NHPP Reliability Cost Model Based on Intensity Function of Log Power Form," Journal of The Korea Knowledge Information Technology Society(JKKITS), Vol. 9, No. 1, February 2014, pp. 22-29.

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

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

■ 저자소개 ■

신 현 철
Shin, Hyun Cheul

1994년 ~현재 백석문화대학교 인터넷정보학부 교수
2002년 2월 원광대학교 컴퓨터공학과 졸업 (공학박사)
1990년 2월 광운대학교 전자계산학과 졸업 (공학석사)

관심분야 : 소프트웨어 신뢰성공학, 정보보호
E-mail : hcshin@bscu.ac.kr

김 희 철
Kim, Hee Cheul

2005년 3월~현재 남서울대학교 산업경영공학과 교수
1998년 2월 동국대학교 통계학과(이학박사)
1992년 2월 동국대학교 통계학과(이학석사)

관심분야 : 소프트웨어 신뢰성공학, 웹 프로그래밍, 전산통계
E-mail : kim1458@nsu.ac.kr

논문접수일: 2014년 11월 12일
수 정 일: 2014년 11월 28일(1차)
2014년 12월 5일(2차)
게재확정일: 2014년 12월 8일