한국정보전자통신기술학회논문지(jkiiect)'16-08, Vol.9 No.4
논문 16-09-04-351 http://dx.doi.org/10.17661/jkiiect.2016.9.4.351

어랑분포를 적용한 유한 및 무한 고장 소프트웨어 신뢰모형에 관한 성능 비교 평가에 관한 연구

양태진*

A Performance Comparative Evaluation for Finite and Infinite Failure Software Reliability Model using the Erlang Distribution

Tae-Jin Yang*

요 약 과학기술이 급속하게 발전함에 따라 더 강력한 소프트웨어 기능의 급속한 발전과 함께 소프트웨어의 복잡성이 크게 증가함으로써 소프트웨어 테스트 및 신뢰성 평가의 어려움이 증가하고 있다. 소프트웨어 고장분석을 위한 비동질적인 포아송 과정에서 결함당 고장발생률이 상수이거나, 단조 증가 또는, 단조 감소하는 패턴을 가질 수 있다. 본 논문에서는 결함의 기대값을 가정하는 유한고장 소프트웨어 NHPP 모형과 수리시점에서도 고장이 발생할 상황을 반영하는 무한고장 NHPP 모형들을 상호 비교 제시하였다. 소프트웨어 신뢰성 분야에서 많이 사용되는 어랑분포에 근거한 유한고장과 무한고장 소프트웨어 신뢰성 모형에 대한 신뢰도 성능을 비교 분석하였다. 그 결과 유한고장 모형이 무한고장 모형보다 효율적으로 좋게 나타났으며, 이 과정에서 모수추정법은 최우추정법을 이용하였다. 본 연구결과를 통하여 소프트웨어 개발자들에게 소프트웨어 고장현상을 파악하는데 도움을 줄 수 있을 것으로 판단된다.

Abstract Science and technology is developing rapidly as more powerful software with the rapid development of software testing and reliability assessment by the difficulty increases with the complexity of the software features of the larger increasesNHPP software reliability models for failure analysis can have, in the literature, exhibit either constant, monotonic increasing or monotonic decreasing failure occurrence rates per fault. In this paper, finite failure NHPP models that assuming the expected value of the defect and infinite failures NHPP models that repairing software failure point in time reflects the situation, were presented for comparing property. Commonly used in the field of software reliability based on Erlang distribution software reliability model finite failures and infinite failures were presented for performance comparative evaluation problem. As a result, finite failure model is better than infinite failure model effectively. The parameters estimation using maximum likelihood estimation in the course of this study was conducted. As the results of this research, software developers to identify software failure property be able to help is concluded.

Key Words : Software failure reliability model, NHPP, Decreasing intensity function, Finite Failure, Infinite Failure, Erlang Distribution

1. 서론

과학기술이 급속하게 발전함에 따라 더 강력한 소프트웨어 기능의 급속한 발전과 함께 소프트웨어의 복잡성이 크게 증가함으로써 소프트웨어 테스트 및 신뢰성 평가의 어려움이 증가하고 있다.

*Corresponding Author : Academic Cooperation Foundation, Namseoul University(solomon645@nsu.ac.kr)
Received July 29, 2016 Revised August 19, 2016 Accepted August 20, 2016

소프트웨어 시스템의 품질을 보장하기 위해 소프트웨어 테스트 및 신뢰성 평가기술이 절실히 필요하다.

소프트웨어 테스트의 목적은 소프트웨어 시스템 요구 사항을 충족시키는지를 테스트하는 것이다. 이 문제는 사용자의 요구조건과 테스팅 비용을 만족시켜야 한다. 소프트웨어 테스팅(디버깅)면에서 비용을 줄이기 위해서는 소프트웨어의 신뢰성의 변동과 테스팅 비용을 사전에 알고 있어야 효율적이다. 결국 소프트웨어 제품의 결함내용을 예측하기 위한 모형 개발이 필요하다. 지금까지 많은 소프트웨어 신뢰성 모형이 제안 되었다. 이 중에서 비동질적 포아송 과정(Non-homogeneous Poisson process; NHPP)을 적용한 모형[1]은 에러 탐색 과정측면에서는 우수한 모형이고 이 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)을 제시하였고 Goel 과 Okumoto[2]은 지수적인 소프트웨어 신뢰성 모형(exponential software reliability growth model)을 제안 하였다. S-형태 모형은 소프트웨어 관리자들이 소프트웨어 및 검사 도구에 익숙해지는 학습 과정을 설명 할 수 있다고 하였고[3] 또한, 대수 선형 위험함수를 이용한 학습과정 특성을 연구하기도 하였다[4].

따라서 본 연구에서는 다양한 현상을 나타낼 수 있는어랑분포를 적용한 소프트웨어 유한고장 및 무한고장 NHPP 모형을 이용하여 신뢰도 성능을 비교하고자 한다.

2. 유한 및 무한 고장 소프트웨어 NHPP 신뢰모형

$N(t)$을 시간 t까지 탐지된 소프트웨어의 고장의 누적수이고 $m(t)$를 이에 대한 기대값를 나타내는 평균값 함수(Mean Value Function)를 나타내고 $\lambda(t)$을 강도함수(Intensity function)로 표시하면 비동질 포아송 과정(NHPP)을 적용시키면 누적 고장수 $N(t)$는 모수 $m(t)$을 가진 포아송 확률 밀도함수 (Probability density function)를 따른다고 알려져 있다[1, 5]. 즉,

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

이런 상황에서, NHPP모형은 평균값함수 $m(t)$와 강도함수 $\lambda(t)$는 다음과 같은 관련성을 가진다고 알려져 있다[6].

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

한편, NHPP모형들은 유한고장모형과 무한고장모형으로 구분된다[1]. 유한고장 NHPP모형에서는 시간 $(0,t]$까지 탐지되어 질 수 있는 결함의 기대값을 θ로나타내면 유한고장 NHPP모형의 평균값함수와 강도함수는 다음과 같은 관계로 유도 할 수 있다[1, 6].

$$m(t)=\theta F(t),\ \lambda(t)=\theta F'(t) \tag{3}$$

또한, 유한고장모형은 수리시간 동안에는 새로운 결함이 발생하지 않는다는 가정을 하지만 무한고장 NHPP모형들은 수리시간 동안에도 결함이 발생할 상황을 반영하기 위한 무한고장 NHPP모형에 대한 평균값함수와 강도함수는 다음과 같이 알려져 있다[1, 5].

$$m(t)= -\ln(1-F(t)) \tag{4}$$

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

관측시간 $(0,t]$까지 관측하기 위한 시간절단(Time truncated)모형은 n번째까지 고장시점 자료 x_n은 다음과 같은 관계로 표시된다.

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

이 고장절단모형에서 Θ을 모수공간이라고 가정하면 NHPP모형의 우도함수는 다음과 같이 알려져 있다[6].

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

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

NHPP 모형에서 테스트 시점 x_n에서 소프트웨어 고장이 발생하는 상황에서 신뢰구간 $(x_n, x_n+\tau]$(단, τ는 임무시간(Mission time))사이에서는 소프트웨어의 고장이 일어나지 않을 확률인 신뢰도(reliability) $\hat{R}(t \mid x_n)$는 다음과 같이 표현될 수 있다[5].

$$\hat{R}(t \mid x_n) = e^{-\int_{x_n}^{x_{n+t}} \lambda(\tau) d\tau} = \exp[-\{m(t+x_n) - m(x_n)\}] \quad (8)$$

3. 제안한 어랑분포를 이용한 유한고장과 무한고장 NHPP 소프트웨어 신뢰성 모형

어랑분포(Erlang distribution)는 다양한 현상을

나타낼 수 있는 분포로서 형상모수(a)와 척도모수(b)에 따른 확률밀도함수와 누적분포는 다음과 같이 알려져 있다[7].

$$f(t) = \frac{b^a}{\Gamma(a)} t^{a-1} e^{-bt} \quad (9)$$

$$F(t) = \left(1 - e^{-bt} \sum_{i=0}^{a-1} \frac{(bt)^i}{i!}\right) \quad (10)$$

단, $a, b > 0,\ a = 1,2,3\cdots,\ t \in [0,\infty]$

본 연구에서는 분포의 특징을 결정하는 형상모수 a가 2인 경우를 고려하고자 한다. 즉 확률밀도함수와 누적분포함수는 다음과 같이 표현할 수 있다.

$$f(t) = b^2 t\, e^{-bt} \quad (11)$$

$$F(t) = 1 - e^{-bt}\,[1 + (bt)] \quad (12)$$

(3)식과 (13), (14)식에 근거하면 유한고장 NHPP 강도함수와 평균값함수는 다음과 같이 표현할 수 있다.

$$\lambda(t) = \theta F'(t) = \theta f(t) = \theta b^2 t\, e^{-bt} \quad (13)$$

$$m(t) = \theta F(t) = \theta\left(1 - e^{-bt}\,[1 + (bt)]\right) \quad (14)$$

따라서 우도함수는 (7)식에 (13)식과 (14)식을 이용하면 다음과 같이 유도된다.

$$L_{NHPP}(\Theta|\ \underline{x}\ ) = \left(\prod_{i=1}^{n} \theta b^2 x_i\, e^{-bx_i}\right) \exp\left[-\theta\left(1 - e^{-bx_n}\,[1 + bx_n]\right)\right] \quad (15)$$

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

모수추정을 위한 로그우도함수는 (15)식과 관련하여 다음과 같이 유도된다.

$$\ln L_{NHPP}(\Theta \mid \underline{x}) = n\ln\theta + 2\,n\ln b - b\sum_{i=1}^{n} x_i - \theta + \theta e^{-bx_n}\,[1 + (bx_n)] \quad (16)$$

(16)식에서 θ와 b에 대하여 편미분을 수행하여 최우추정값 $\hat{\theta}_{MLE}$와 $\hat{b}_{MLE}$을 수치 해석적 방법으로 추정 할 수 있다.

$$\frac{\partial \ln L_{NHPP}(\Theta \mid \underline{x})}{\partial \theta} = \frac{n}{\theta} - 1 + e^{-b x_n}(1 + b x_n) = 0 \quad (17)$$

$$\frac{\partial \ln L_{NHPP}(\Theta \mid \underline{x})}{\partial b} = \frac{2n}{b} - \theta b (x_n)^2 e^{-b x_n} - \sum_{i=1}^{n} x_i = 0 \quad (18)$$

또한, 임무시간을 부여하면 신뢰도는 (8)식을 이용하여 다음과 같이 추정된다[12].

$$\hat{R}(\tau \mid x_n) = \exp\left[-m(\tau + x_n) + m(x_n)\right] \quad (19)$$

단 ,

$$m(x_n + \tau) = \theta\left(1 - e^{-b(\tau + x_n)}\left[1 + b(\tau + x_n)\right]\right),$$

$$m(x_n) = \theta\left(1 - e^{-b x_n}\left[1 + b x_n\right]\right)$$

반면에 무한고장 NHPP 강도함수와 평균값함수는 (4)식과 (5)식을 이용하면 다음과 같이 유도할 수 있다.

$$\lambda(t) = f(t)/(1 - F(t)) = \frac{b^2 t}{1 + bt} \quad (20)$$

$$m(t) = -\ln(1 - F(t)) = b t - \ln(1 + b t) \quad (21)$$

이 상황에서 우도함수는 유한고장모형과 유사한 방법을 이용하면 다음과 같이 유도된다.

$$L_{NHPP}(\Theta \mid \underline{x}) = \left(\prod_{i=1}^{n} \frac{b^2 x_i}{1 + b x_i}\right) \exp\left[-b x_n + \ln(1 + b x_n\right] \quad (22)$$

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

최우추정법을 이용하기 위한 로그우도함수는 (22)식과 관련하여 다음과 같은 식이 된다.

$$\ln L_{NHPP}(\Theta \mid \underline{x}) = \sum_{i=1}^{n} \ln(b^2 x_i) - \sum_{i=1}^{n} \ln(1 + b x_i) - b x_n + \ln(1 + b x_n) \quad (23)$$

(23)식에서 b에 대하여 편미분하면 최우추정값 $\hat{b}_{MLE}$을 수치 해석적 방법으로 추정할 수 있다.

$$\frac{\partial \ln L_{NHPP}(\Theta \mid \underline{x})}{\partial b} = \frac{2n}{b} - \sum_{i=1}^{n} \frac{x_i}{1 + b x_i} - x_n + \frac{x_n}{1 + b x_n} = 0 \quad (24)$$

또한, 임무시간이 부여되면 신뢰도는 다음과 같은 방법으로 추정된다[6].

$$\hat{R}(\tau \mid x_n) = \exp\left[-m(\tau + x_n) + m(x_n)\right] \quad (25)$$

단, $m(\tau + x_n) = b\left[\tau + x_n - \ln(1 + b(\tau + x_n))\right]$,

$m(x_n) = b t - \ln(1 + b x_n)$

4. 소프트웨어 고장시간 분석

이 절에서는 소프트웨어 고장 시간자료 [8] (Failure time data)를 적용하여 본 논문에서 제안하는 유한과정 및 무한고장 NHPP 소프트웨어 신뢰모형에 대하여 신뢰도 성능을 비교 분석하고자 한다. 이 자료의 고장시간은 13.853 시간단위에 30번의 고장이 발생된 자료이며 표 1에 나열 되어 있다.

또 한 자료에 대한 신뢰성을 확보하기 위하여 자료에 대한 추세검정이 선행 되어야 한다[9, 10]. 따라서 라플라스 추세검정(Laplace trend test)을 실시하였다. 그 결과는 그림 1에 요약되었다. 이 그림에서 라플라스요인(Factor)이 -2와 2사이에 존재함으로서 즉, 극단값(Extreme value)이 존재하지 않으므로 이 자료를 이용하여 신뢰성장모형을 제시하는 것이 효율적임을 나타내고 있다[6, 10].

표 1. 소프트웨어의 고장시간자료
Table 1. software failure time data

Failure number	Failure time (hours)	Failure numbe	Failure time(hours)
1	30.02	16	151.78
2	31.46	17	177.50
3	53.93	18	180.29
4	55.290	19	182.21
5	58.720	20	186.34
6	71.920	21	256.81
7	77.070	22	273.88
8	80.900	23	277.87
9	101.90	24	453.93
10	114.87	25	535.00
11	115.34	26	537.27
12	121.57	27	552.9
13	124.97	28	673.68
14	134.07	29	704.49
15	136.25	30	738.68

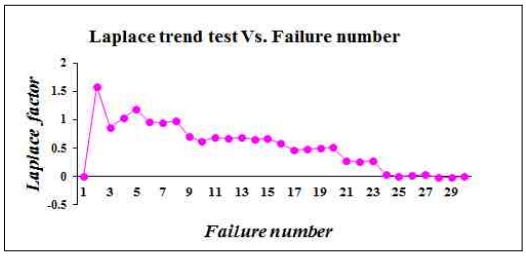

그림 1. 라플라스 추세검정
Fig. 1. Laplace trend test

모수추정은 최우추정법을 이용하였고, 비선형 방정식의 계산방법은 수치 해석적 기본 방법인 이분법(Bisection method)을 사용하였다. 이러한 계산은 초기 값을 0.001과 5을, 허용한계(Tolerance for width of interval)는 10^{-5} 을 주고 수렴성을 확인 하면서 충분한 반복 횟수인 100번을 C-언어를 이용하여 모수 추정을 수행하였다. 최우추정법의 결과와 모형에 대한 효율성을 조사하기 위한 기준으로서 MSE(평균제곱오차)와 R^2(결정계수)가 표 2에 요약되었다.

표 2. 모수 추정값 및 MSE , R^2
Table 2. Parameter estimation of the each model and MSE , R^2

Model	MLE		Model comparison	
	$\hat{\theta}$	$\hat{b}$	MSE	R^2
Finite	30.5978	0.0079	14.0893	0.9323
Infinite	-	0.0478	60.9796	0.7808

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

표 2에서 평균제곱오차는 실제 관찰 값과 예측 값에 대한 차이를 측정하는 측도로서 다음과 같이 정의 된다[3, 6].

$$MSE = \frac{\sum_{i=1}^{n}\left(m(x_i) - \hat{m}(x_i)\right)^2}{n-k} \tag{21}$$

단, $m(x_i)$은 시간$(0, x_i]$까지 나타난 에러들의 누적함수를 의미하고 $\hat{m}(x_i)$는 x_i 시점까지 평균값 함수로부터 추정된 에러의 누적개수를 의미한다. 그리고 n은 관찰 값의 수이고 k는 모수의 수를 의미한다. 즉 비교에 있어서 평균제곱오차 값이 작으면 상대적으로 효율적인 모형이 된다.

R^2(결정계수)는 관찰 값의 차이에 대한 설명력을 나타내는 도구로서 다음과 같이 정의 된다[5, 6].

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left(m(x_i) - \hat{m}(x_i)\right)^2}{\sum_{i=1}^{n}\left(m(x_i) - \sum_{j=1}^{n} m(x_j)/n\right)^2} \tag{22}$$

즉 비교에 있어서 결정계수 값이 크면 상대적으로 효율적인 모형이 된다.

표 2에서 유한고장모형이 무한고장모형보다 실제 값과 예측 값에 대한 차이를 측정하는 평균

제곱오차가 낮게 나타나고 있다. 즉 유한고장모형이 효율적모형으로 판단 할 수 있다. 각 고장수에 따른 MSE값은 그림 2에서 보여주고 있다. 즉, 이 그림에서 전 고장수 범위에서 무한고장 모형이 유한고장 모형보다 평균제곱오차가 높게 나타남을 보여주고 있다.

예측 값의 차이에 대한 설명력을 의미하는 결정계수도 무한고장 모형이 유한고장 모형보다 평균제곱오차가 높게 나타나기 때문에 유한고장 모형이 효율적인 모형으로 간주할 수 있다.

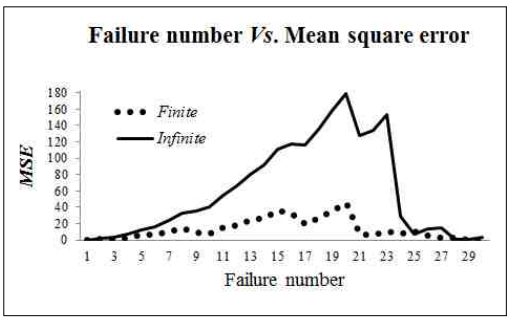

그림 2. 평균제곱오차
Fig. 2. Mean square error

그림 3은 강도함수에 대한 패턴이 요약 되었다. 이 그림에서 유한고장 모형은 고장시간이 지남에 따라 처음에는 증가하다가 일정시간이 지나면 감소형태를 보이고 있다. 반면에 무한고장모형은 완만한 증가 추세를 보이고 있다.

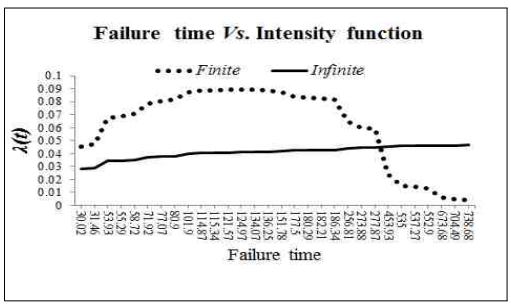

그림 3. 강도함수의 형태
Fig. 3. Pattern of intensity function

그림 4는 평균값 함수에 대한 패턴이 요약 되었다.이 그림에서 모든 모형은 참값과의 차이에서 과소평가 추정이 이루어졌으나 무한고장 모형이 유한고장 모형보다 참값과의 차이의 폭이 작게 추정되었다.

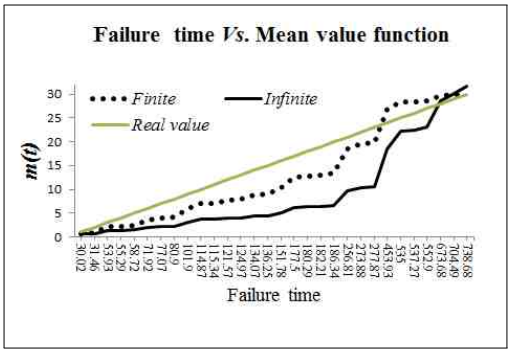

그림 4. 평균값 함수의 형태
Fig. 4. Pattern of mean value function

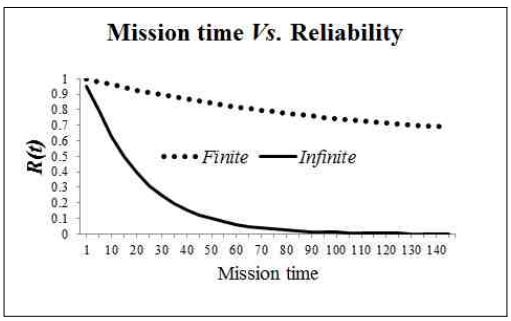

그림 5. 신뢰도의 형태
Fig. 5. Pattern of reliability

그림 5에서 보여 주듯이 임무시간에 대한 신뢰도 그림에서도 유한고장모형이 무한고장모형과 비교 했을 때 임무시간이 증가 할수록 유한고장모형이 무한고장모형보다 각 시점에서 신뢰도의 상승으로 나타나고 있다.

5. 결론

소프트웨어 신뢰성은 개발의 최종단계에 있는 테스트 공정이나 실제 사용단계에 있어서 소프트웨어 내에 존재하는 고장 수나 고장 발생시간에 의해서 효과적으로 평가할 수 있는 상황으로 그 평가 기술이 중요하게 된다. 따라서 소프트웨어 개발의 테스트공정이나 실제 사용 단계에 있어서 고장 발생 환경이나 고장 발생현상을 수리적으로 모형화가 가능하면 평가를 할 수 있다.

따라서 본 연구에서는 소프트웨어 관리자들이 소프트웨어 고장원인 및 검사 도구에 활용 할 수 있는 소프트웨어 신뢰성 분야에서 많이 사용되는 어랑분포에 근거한 유한고장과 무한고장 소프트웨어 신뢰성모형에 대한 신뢰도 성능을 비교분석하였다. 그 결과 유한고장모형이 무한고장모형보다 실제 값과 예측 값에 대한 차이를 측정하는 평균제곱오차가 낮고 예측 값의 차이에 대한 설명력을 의미하는 결정계수도 제일 높게 나타나기 때문에 유한고장모형이 무한고장모형보다 효율적인 모형으로 간주할 수 있다. 또한, 미래 임무시간에 대한 신뢰도에 있어서도 유한고장모형이 무한고장모형보다 임무시간이 증가할수록 신뢰도의 상승으로 나타나고 있다. 따라서 이 분야에서 기존의 모형의 하나의 대안으로 유한고장모형이 무한고장모형보다는 효율적으로 사용할 수 있음을 확인 할 수 있었다. 이 연구를 통하여 소프트웨어 개발자들은 다양한 고장모형을 고려함으로서 소프트웨어 고장형태에 대한 사전지식을 파악하는데 도움을 줄 수 있으리라 판단된다.

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] 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.

[4] 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.

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

[6] Kim H-C, A Performance Analysis of Software Reliability Model using Lomax and Gompertz Distribution Property., Indian Journal of Science and Technology, Vol. 9, No. 20, pp. 1-6, 2016.

[7]V. K. Rohatgi, ″Statistical inference″,John Wiley & Sons, Inci, New York, pp.398-416, 1984.

[8] K,H Rao, R. S, Prasad and. R.L.Kantham “Software Reliability Measuring using Modified Maximum Likelihood Estimation and SPC”, International Journal of Computer Applications (0975 - 8887), Vol. 21, No.7, pp. 1-5., May 2011.

[9] 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, Voi.8, No.34, pp.1-7, 2015.

[10] 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.
[11] Tae-Jin Yang, "The Comparative Study of NHPP Software Reliability Mode Based on Log and Exponential Power Intensity Function", The Journal of Korea Institute of Information, Electronics, and Communication Technology, Vol 8, No6, pp445-452, 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 정보통신학과 교수, HRD센터 교수부장
- 2014년 3월 ~ 현재 : 남서울대학교 산학협력단 교수

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