한국정보전자통신기술학회논문지(jkiiect)'16-12, Vol.9 No.6
논문 16-09-06-566 http://dx.doi.org/10.17661/jkiiect.2016.9.6.566

어랑분포의 형상모수 변화에 따른 소프트웨어 개발 비용모형에 관한 비교 연구

양태진*

The Comparative Software Development Cost Model Considering the Change in the Shape Parameter of the Erlang Distribution

Tae-Jin Yang*

요 약 소프트웨어 개발과정에서 소프트웨어 신뢰성은 매우 중요한 문제 중에 하나이다. 소프트웨어 고장현상을 분석하기 위하여 비동질적인 포아송과정에서 고장 발생 추이를 의미하는 위험함수가 고장시간에 독립적으로 일정하거나, 종속적인 경우, 즉 비-증가 또는, 비-감소하는 속성을 가질 수 있다. 본 연구에서는 소프트웨어 제품 테스팅 과정에서 고장 수명분포로서 어랑분포의 다양한 형상모수를 고려한 소프트웨어 개발 비용 분석에 대하여 연구되었다. 소프트웨어 고장현상을 분석하기 위하여 모수추정은 최우추정법이 사용되었다. 따라서 본 논문에서는 어랑분포의 형상모수를 고려한 소프트웨어 개발비용모형 분석을 위하여 소프트웨어 고장간격 시간자료를 이용하여 비교 및 평가하였다. 그 결과 형상모수에 따른 비용곡선을 비교 하였을 때 형상모형이 작을수록 비용이 많고 소프트웨어 최적 방출시간이 지연됨을 알 수 있었다. 이 연구를 통하여 소프트웨어 개발자들에게 소프트웨어 형상모수에 따른 개발 비용을 탐색하는데, 기본적으로 도움을 줄 수 있는 사전정보의 역할을 할 수 있을 것으로 판단된다.

Abstract Software Reliability implemented in software development is one of the most important issues. In finite failure NHPP software reliability models for software failure analysis, the hazard function that means a failure rate may have constant independently for failure time, non-increasing or non-decreasing pattern. In this study, software development cost analysis considering the variable shape parameter of Erlang distribution as the failure life distribution in the software product testing process was studied. The software failure model was applied finite failure Non-Homogeneous Poisson Procedure and the parameters approximation using maximum likelihood estimation was accompanied. Thus, this paper was presented comparative analysis by applying a software failure time data to the software, considering the shape parameter of Erlang distribution for development cost model analysis. When compared to the cost curve in accordance with the shape parameter, the model of smaller shape can be seen that the optimal software release time delay and more cost. Through this study, it is thought that it can serve as a preliminary information which can basically help the software developers to search for development cost according to software shape parameters.

Key Words : Software development Cost model, NHPP, Erlang Distribution, Finite Failure Model

1. 서론

소프트웨어에 대한 신뢰성은 시스템 신뢰도에 영향을 미치는 중요한 원인이 되고 있다. 따라서 소프트웨어 결함 때문에 발생할 수 있는 컴퓨터 시스템의 고장은 소프트웨어 운용자들에게 큰 피

Funding for this paper was provided by Namseoul University year 2016
*Corresponding Author : Academic Cooperation Foundation, Namseoul University(solomon645@nsu.ac.kr)
Received November 26, 2016 Revised December 19, 2016 Accepted December 19, 2016

해를 줄 수 있다. 결국, 소프트웨어를 개발하는 동안 소프트웨어 신뢰성 분석은 중요한 기본 문제가 된다. 결국, 이 문제는 소프트웨어 사용자의 요구사항과 테스팅 비용을 만족시켜야 한다. 소프트웨어 테스팅 과정에서 비용을 효율적으로 관리하기 위해서는 소프트웨어의 신뢰성의 추세와 테스팅 비용을 사전에 미리 예측 할 수 있다면 효율적인 개발과정이 될 수 있다. 따라서 신뢰도, 비용 및 소프트웨어 방출시점에 대한 추정시간을 고려한 소프트웨어 개발과정은 중요한 문제가 된다. 따라서, 소프트웨어 제품의 결함내용을 사전에 미리 예측하기 위한 신뢰성 모형 개발이 필요하게 된다. 이러한 문제를 해결하기 위하여 지금까지 많은 소프트웨어 신뢰성 관련모형이 제시되어 왔다. 이러한 신뢰성 모형중에서 비동질적 포아송 과정(Non-Homogeneous Poisson Process ; NHPP)을 이용한 모형[1]은 "에러 탐색시행에서는 효율적인 모형이고, 결함이 발생하면 즉시 제거되고 디버깅(debugging) 과정에서 새로운 결함이 생성되지 않는다."는 가정을 하고 있는 모형이라고 알려져 있다.

Goel과 Okumoto[2]은 결함의 누적수가 S-형태나 지수적 형태(S-shaped or exponential-shaped)를 가진 평균값함수(Mean value function)을 이용한 지수적 소프트웨어 신뢰성모형(Exponential software reliability growth model)을 제시하였다.

또한, Huang[3]은 일반화된 로지스틱 테스팅 노력함수(Generalized logistic testing effort function)와 변환점 모수(Change point parameter)을 이용하여 효율적인 소프트웨어 신뢰성을 분석하는 방법을 제시하기도 하였다. 그리고, 고장 탐색측면에서 S-형태 모형은 소프트웨어 운영자들이 소프트웨어 고장 검사도구에 이용될 수 있는 학습과정을 제시 하였다[4].

더불어, 공정관리 측면에서 김희철[5]은 레일리형과 버르형 NHPP 소프트웨어 신뢰성 모형에 관한 공정관리 접근방법을 이용하여 비교한 연구 결과에서 레일리 분포 모형 보다는 버르 분포모형이 상대적으로 효율적 모형임을 확인하였다.

본 연구에서는 소프트웨어 고장 간격시간 자료를 바탕으로 NHPP 모형에서 어랑분포를 수명분포로 적용하고, 이 분포의 다양한 형상모수를 고려한 소프트웨어 개발 비용모형에 대한 비교 분석을 하고자 한다.

2. NHPP 소프트웨어 신뢰성

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

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

따라서, NHPP모형의 평균값함수 $m(t)$와 강도함수 $\lambda(t)$는 다음과 같은 관련성을 가진다[1, 6].

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

이처럼 시간관련(Time domain) NHPP모형들은 유한고장모형과 무한고장 모형으로 분류된다[6]. 유한고장 NHPP모형에서는 시간 $(0,t]$까지 검출될 수 있는 결함의 기대값을 θ라고 하면, 유한고장 NHPP모형의 평균값 함수와 강도함수는 다음과 같은 관계로 유도된다[1, 7].

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

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

$$x_n=\sum_{i=1}^{n} t_i \ (i=1,2,\cdots,n\,;\,0\le x_1\le x_2\le\cdots\le x_n) \quad (4)$$

이라고 하면, 고장시점이 n번째까지 검출된 시간 절단 모형일 경우에 데이터 집합 D_{x_n}은 $\{x_1, x_2 \cdots x_n\}$으로 구성된다. 이와 같은 고장 절단 모형에서 Θ을 모수공간이라고 표시하면, 비동질적 포아송 과정(Non-Homogeneous Poisson Process ; NHPP) 모형의 우도함수는 다음과 같 다[1, 7].

$$L_{NHPP}(\Theta|\ \underline{x}\ ) = \left(\prod_{i=1}^{n} \lambda(x_i)\right) \exp[-m(x_n)] \qquad (5)$$

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

3. 소프트웨어 개발 비용모형

소프트웨어 개발 예상 총비용은 일반적으로 다음과 같이 구성된다[6, 8].

$$\begin{aligned} E &= E_1 + E_2 + E_3 + E_4 \\ &= E_1 + C_2 \times t + C_3 \times m(t) + C_4 \times [m(t+t') - m(t)] \end{aligned} \qquad (6)$$

단, E: 소프트웨어 개발 예상 총비용

E_1: 소프트웨어 설계 및 초기 소프트웨어 개발의 비용(예를 들면, 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 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 기대 수)

E_4: 소프트웨어 운영 시스템에서 남아있는 모든 결함을제거하는 비용(상수) 즉,

$E_4 = C_4 \times [m(t+t') - m(t)]$ (단, C_4는 소프트웨어가 방출된 이후에 소프트웨어 운영 단계에서 사용자가 관찰되는 결함수정 비용, t' 는 소프트웨어 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 다음(7)식과 같이 유도 할 수 있다[8].

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

4. 제안된 어랑 분포 NHPP 유한고장 모형

어랑분포(Erlang distribution)는 다양한 현상을 나타낼 수 있는 분포로서 형상모수(a)와 척도모수(b)에 따른 확률밀도함수와 누적분포는 다음과 같다[5].

$$f(t) = \frac{b^a}{\Gamma(a)} t^{a-1} e^{-bt} \qquad (8)$$

$$F(t) = \left(1 - e^{-bt} \sum_{i=0}^{a-1} \frac{(bt)^i}{i!}\right) \qquad (9)$$

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

유한고장 NHPP모형에서는 시간 $(0, t]$까지 검출되어 질 수 있는 결함의 기대값을 θ로 나타내면 유한고장 NHPP 소프트웨어 신뢰성 모형의 평균값 함수와 강도함수는 다음과 같은 관계로 유도 할 수 있다[1, 6].

$$m(t) = \theta F(t),\ \lambda(t) = \theta F'(t) \qquad (10)$$

관측시간 $(0, t]$까지 검출하기 위한 시간절단(Time truncated)모형은 n번째까지 고장시점 자료 x_n은 다음과 같은 관계로 표시된다[7].

$$x_n = \sum_{i=1}^{n} t_i\ (i = 1,2,\cdots,n\,;\, 0 \le x_1 \le x_2 \le \cdots \le x_n) \qquad (11)$$

이 경우에 확률밀도함수$f(\bullet)$ 와 누적분포$F(\bullet)$을 이용하면 NHPP모형의 우도함수는 다음과 같이 알려져 있다[6].

$$L_{NHPP}(\Theta|\ \underline{x}\ ) = \left(\prod_{i=1}^{n} \theta f(x_i)\right) \exp[-\theta F(x_n)] \qquad (12)$$

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

의미한다.

따라서 (12)식을 이용하면 유한고장 NHPP모형의 로그우도함수는 다음과 같이 유도 할 수 있다.

$$\ln L_{NHPP}(\Theta|\underline{x}) = \quad (13)$$
$$n\ln\theta - n\ln\Gamma(a) + na\ln b + (a-1)\sum_{i=1}^{n}\ln x_i$$
$$-b\sum_{I=1}^{n}x_i - \theta + \theta\, e^{-b x_n}\left(\sum_{i=0}^{a-1}\frac{(b\,x_n)^i}{i!}\right)$$

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

$$\frac{\partial \ln L_{NHPP}(\Theta\,|\,\underline{x})}{\partial\theta} = \quad (14)$$
$$\frac{n}{\theta} - 1 + e^{-b x_n}\left(\sum_{i=0}^{a-1}\frac{(b\,x_n)^i}{i!}\right) = 0$$

$$\frac{\partial \ln L_{NHPP}(\Theta\,|\,\underline{x})}{\partial b} = \quad (15)$$
$$\frac{a\,n}{b} - \sum_{i=1}^{n}x_i + \frac{\partial\left[\theta\, e^{-b x_n}\left(\sum_{i=0}^{a-1}\frac{(b\,x_n)^i}{i!}\right)\right]}{\partial b} = 0$$

본 연구에서는 수명분포의 특징을 결정하는 형상모수 $a=1,2,3$인 경우를 고려하고자 한다.

즉, 형상모수 $a=1$인 경우는 다음과 같이 표현된다.

$$\frac{\partial \ln L_{NHPP}(\Theta\,|\,\underline{x})}{\partial\theta} = \frac{n}{\theta} - 1 + e^{-b x_n} = 0 \quad (16)$$

$$\frac{\partial \ln L_{NHPP}(\Theta\,|\,\underline{x})}{\partial b} = \quad (17)$$
$$\frac{n}{b} - \theta\, x_n\, e^{-b x_n} - \sum_{i=1}^{n}x_i = 0$$

형상모수 $a=2$ 인 경우는 다음과 같이 표현된다.

$$\frac{\partial \ln L_{NHPP}(\Theta\,|\,\underline{x})}{\partial\theta} = \quad (18)$$
$$\frac{n}{\theta} - 1 + e^{-b x_n}(1 + b x_n) = 0$$

$$\frac{\partial \ln L_{NHPP}(\Theta\,|\,\underline{x})}{\partial b} = \quad (19)$$
$$\frac{2\,n}{b} - \theta\, b\, x_n^2 e^{-b x_n} - \sum_{i=1}^{n}x_i = 0$$

형상모수 $a=3$인 경우는 다음과 같이 표현된다.

$$\frac{\partial \ln L_{NHPP}(\Theta\,|\,\underline{x})}{\partial\theta} = \quad (20)$$
$$\frac{n}{\theta} - 1 + e^{-b x_n}\left(1 + b x_n\,\frac{x_n^2}{2}b^2\right) = 0$$

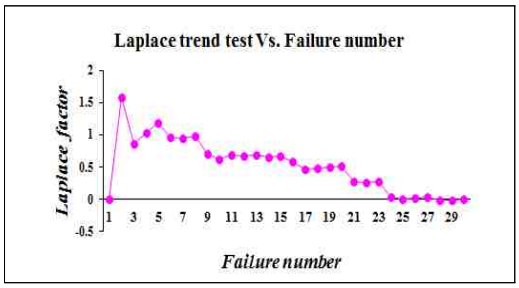
$$\frac{\partial \ln L_{NHPP}(\Theta\,|\,\underline{x})}{\partial b} = \quad (21)$$
$$\frac{3\,n}{b} - \theta\, b^2\,\frac{x_n^3}{2}e^{-b x_n} - \sum_{i=1}^{n}x_i = 0$$

5. 소프트웨어 고장시간 및 비용분석

이 절에서는 소프트웨어 고장 시간자료[10](Failure time data)를 이용하여 본 논문에서 제안하는 소프트웨어 신뢰성 비용모형들을 분석 및 비교를 하고자 한다. 이 자료의 고장시간은 18.735 시간단위에 30번의 고장이 관측된 자료이며 [표 1]에 나타내었다.

[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

또한, 자료에 대한 신뢰성을 확보하기 위하여 추세 검정이 선행 되어야 하는데[7, 11], 본 연구에서 추세분석은 라플라스 추세검정(Laplace trend test)을 사용하였다. [그림 1]에서 라플라스 추세 검정의 결과는 라플라스 요인(Factor)의 값이 "-2와 2"사이에 존재함으로서 즉, 극단값(Extreme value)이 존재하지 않기 때문에, 이 자료를 이용하여 개발 비용 모형을 제안하는 것이 효율적임을 시사하고 있다[7]. 모수에 대한 추정은 최우추정법을 이용하고 모수추정을 용이하게 하기 위하여 원래의 고장시간 데이터에 수치변환($Failure\ time \times 10^{-2}$)하여 적용하였고, 비선형 방정식의 계산방법은 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. 그 결과는 [표 2]에 요약되었다.

[표 2] 형상모수에 따른 모수 추정값
[Table 2] Parameter estimation considering the shape parameter

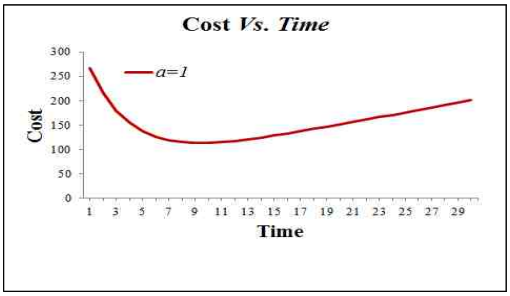
Shape parameter	*MLE*	
	$\hat{\theta}$	$\hat{b}$
$a=1$	33.4092	0.3089
$a=2$	30.5978	0.7922
$a=3$	31.2672	1.2425

Note. *MLE* : Maximum likelihood estimation;

본 연구에서는 다음 (22)식과 같이 가정하여, 비용 곡선을 분석하고자 한다.

$(Assumption)$

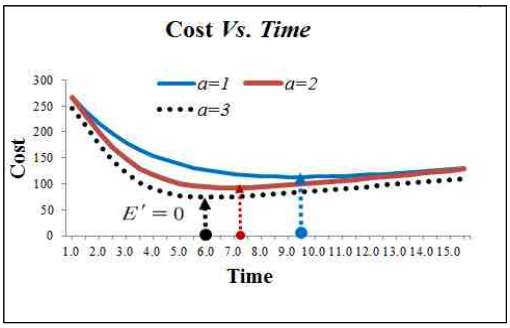
$$E_1 = 5\$,\ c_2 = 5\$,\ c_3 = 1.5\$,\ c_4 = 10\$,\ t' = 20 \qquad (22)$$



[그림 2] $(Assumption)$에 대한 비용곡선
[Fig. 2] The cost curve under the condition of $(Assumption)$

(22)식의 가정을 이용한 비용곡선의 결과는 [그림 2]에 요약되었다. 이 그림에서 보여주듯이 제안된 모형의 비용곡선은 초기단계에서는 감소하다가 일정 고장시간이 지나면 점차 증가하는 추세를 보이고 있다. 그 이유는 소프트웨어 시스템의 잔여 결함의 수는 결함이 제거되는 과정에서 점점 줄어들게 된다. 즉, 남아 있는 결함이 관측될 확률은 상대적으로 낮아지게 된다. 따라서 테스트의 초기 단계에 있는 소프트웨어는 여전히 많은 오류가 있기 때문에 쉽게 감지되지만, 제거단계에서 오류를 제거하는 비용은 운용단계에서 오류를 제거하는 비용보다 훨씬 작기 때문에 소프트웨어의 총비용은 결국 감소한다. 그러나 후반부의 단계에서 소프트웨어에 남아있는 결함의 수는 작기 때문에, 이 후반부 테스트 단계에서 오류를 검출할 시간이 상대적으로 길고 오류를 제거하는 비용은 운용단계에서 상대적으로 높기 때문에 비용곡선은 시간이 흐름에 따라 지속적으로 증가하게 된다.

결국, 비용곡선의 패턴을 이용하여 최적의 소프트웨어 방출시간을 추정 할 수 있다. 이러한 상황은 가장 현실적인 상황이며 대부분의 경우 실제 소프트웨어의 개발 과정에서 이러한 패턴을 가지게 된다[8].



[그림 3] 형상모수에 따른 비용곡선의 비교
[Fig. 3] Comparative of the cost curve under the condition shape parameter

[그림 3]에서 형상모수에 따른 비용곡선을 비교하였다. 그 결과, 수명현상을 결정하는 형상모형이 작을수록 비용이 많고 소프트웨어 최적 방출시간이 연기됨을 알 수 있다. 따라서 이 경우에는 소프트웨어 방출시기 이후에 결함을 감소시킬 수 있도록 운영단계보다 테스팅 단계에서 가능한 결함들을 제거해야 함을 알수 있다.

5. 결론

소프트웨어 신뢰도 운용 모형은 최적의 소프트웨어 방출 시간과 테스트 작업의 비용을 예측할 수 있다[12, 13].

따라서 보다 효율적인 모형은 테스트 비용을 줄이고 소프트웨어를 방출 이익을 증가 시 킬 수 있도록 해야 한다. 본 연구에서 사용된 어랑분포의 형상모수에 근거한 유한고장 NHPP모형을 이용한 소프트웨어 개발 비용 모형은 보다 효율적으로 최적의 소프트웨어 방출 시간을 예측 할 수 있다. 따라서, 제안된 모형의 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 감지하고, 제거하는 단계에서 오류 수정 비용은 운용 단계에서 남아있는 모든 오류를 제거하는 비용보다 작지만, 운영 시간이 증가함에 따라 비용이 증가함을 알 수 있다. 따라서 최적의 소프트웨어 방출시간은 현실적으로 미리 예측해 볼 수 있다. 빅데이터를 처리하는 과정에서 대용량 소프트웨어를 수정하고 변경하는 작업 과정은 결함의 발생을 피할 수 없는 상황이 현실이다. 신뢰성 요구를 만족하고 총비용을 최소화하는 상황이 최적방출시간이 된다. 본 연구를 통하여 소프트웨어 개발자들에게 최적의 방출시간과 가장 경제적인 개발 비용을 파악하는데 실제적인 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]Huang C-Y. Performance analysis of software reliability growth models with testing-effort and change-point. J Syst Software. 2005; 76:181-94.

[4]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.

[5]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.

[6]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.

[7] 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.

[8] Ye Zhang, and Kaigui Wu, "Software Cost Model Considering Reliability and Time of Software in Use",Journal of Convergence Information Technology, Vol.7, No. 13, pp. 135-142, 2012.

[9] Kersey, Jing Xiong, "Weighted Inverse Weibull and Beta-Inverse Weibull Distribution", Electronic Theses & Dissertations Paper 661, pp. 1-53, 2010.

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

[11]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.

[12] Vincent Almering, Michiel van Genuchten, and Ger Cloudt, Peter J.M. Sonnemans, "Software Reliability Growth Models in Practice", IEEE SOFTWARE, pp. 82-88, 2007.

[13] Tae-Jin Yang, "A Performance Comparative Eval-uation for Finite and Infinite Failure Software Reliability Model using the Erlang Distribution", The Journal of Korea Institute of Information, Electronics, and Communication Technology, Vol 9, No4, pp351-358, 2016.

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, 산학처장/학생처장, 교수부장
- 2014년 3월 ~ 현재 : 남서울대학교 산학협력단 교수

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