디지털산업정보학회 논문지 제7권 제2호-2011년 6월

NHPP 극값 분포 소프트웨어 신뢰모형에 대한 학습효과 기법 비교 연구*

김 희 철**

The Comparative Study of NHPP Extreme Value Distribution Software Reliability Model from the Perspective of Learning Effects

Kim, Hee Cheul

〈Abstract〉

In this study, software products developed in the course of testing, software managers in the process of testing software test and test tools for effective learning effects perspective has been studied using the NHPP software. The finite failure non-homogeneous Poisson process models presented and the life distribution applied extreme distribution which used to find the minimum (or the maximum) of a number of samples of various distributions. Software error detection techniques known in advance, but influencing factors for considering the errors found automatically and learning factors, by prior experience, to find precisely the error factor setting up the testing manager are presented comparing the problem. As a result, the learning factor is greater than automatic error that is generally efficient model could be confirmed. This paper, a numerical example of applying using time between failures and parameter estimation using maximum likelihood estimation method, after the efficiency of the data through trend analysis model selection were efficient using the mean square error.

Key Words : Learning Effects, Non-Homogeneous Poission Process, Extreme Value Distribution Model

Ⅰ. 서론

소프트웨어 고장으로 인한 컴퓨터 시스템의 고장은 우리 사회에 엄청난 손실을 유발할 수도 있다. 따라서 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 불가결하다. 결국 소프트웨어 제품의 결함내용을 예측하기 위한 모형 개발이 필요하다. 지금까지 많은 소프트웨어 신뢰성 모형이 제안되었다. 이 중에서 비동질적 포아송 과정(NHPP)에 의존한 모형은 에러 탐색 과정 측면에서는 우수한 모형이고, 이 모형은 결함

* 이 논문은 2011학년도 남서울대학교 학술연구비 지원에 의하여 연구되었음.

** 남서울대학교 산업경영공학과 교수 (교신저자)

이 발생하면 즉시 제거되고 디버깅 과정에서 새로운 결함이 발생되지 않는다는 가정을 하고 있다.

Gokhale과Trivedi[1]은 고양된 비동질적인 포아송 과정 모형(Enhanced NHPP) 모형을 제시하였고 Goel과 Okumoto[2]은 지수적 소프트웨어 신뢰성 모형(Exponential software reliability growth model)을 제안하였다. 이 모형은 결함의 누적수가 S-형태나 지수적 형태(S-shaped or exponential-shaped)를 가진 평균값 함수(Mean value function)을 이용하였다. 이러한 모형에 의존한 일반화 모형은 Yamada와 Ohba[3]에 의해 지연된 S 형태 신뢰 성장모형(Delayed S-shaped reliability growth model)과 변곡된 S형태 신뢰 성장모형(Inflection S-shaped reliability growth model)이 제안되었다. Zhao[4]는 소프트웨어 신뢰도에서 변환점 문제를 제시하였고 Shyur[5]는 변환점을 이용한 일반화한 신뢰도 성장모형을 제안하였다. Pham와 Zhang[6]는 테스팅 커버리지(Coverage)를 측정하여 소프트웨어 안정도를 평가 할 수 있는 소프트웨어 안정도 모형을 제시했다. 비교적 최근 Huang[7]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. 그리고 최근에는 S-형태 모형은 소프트웨어 관리자들이 소프트웨어 및 검사 도구에 익숙해지는 학습 과정을 설명할 수 있다고 하였다[8].

따라서 본 연구는 고장시점을 측정할 때 두터운 꼬리 영역을 가지는 수명분포를 명시적으로 고려한다는 점 이외에, 분포의 비대칭성도 효과적으로 고려할 수 있다는 장점을 가진 극값 분포를 유한 고장 NHPP 소프트웨어 수명분포로 적용하고 이 모형에 대하여 자동적으로 발견되는 에러를 고려한 영향요인(Influential factor)과 사전에 설정되는 학습요인(Learning factor)으로 구성된 학습효과의 특성에 대한 문제를 비교 제시하였다.

Ⅱ. 관련연구

2.1 무한고장과 유한 고장 NHPP 모형

NHPP 모형에서 평균값 함수 $m(t)$ (Mean value function)와 강도 함수 $\lambda(t)$는 다음과 같은 관계로 표현할 수 있다[9].

$$m(t)=\int_0^t \lambda(s)ds\ ,\ \frac{dm(t)}{dt}=\lambda(t) \qquad (1)$$

$N(t)$는 모수 $m(t)$을 가진 포아송 확률밀도함수 (Probability density function) 로 알려져 있다. 즉,

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

이처럼 시간관련 모형(Time domain models)들은 NHPP에 의해서 확률 고장과정으로 설명이 가능하다. 이러한 NHPP 모형들은 유한 고장 모형과 무한 고장 범주로 분류한다[10].

유한 고장(Finite failure) NHPP 모형들은 충분한 테스트 시간이 주어지면 결함들(Faults)의 기대 값이 유한 값$\left(\lim_{t\to\infty} m(t)=\theta<\infty\right)$을 가지고 반면에 무한 고장 (Infinite failure) NHPP 모형들은 무한 값을 가진다고 가정 된다. 무한 고장 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[10].

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

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.

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

반면에 유한 고장 NHPP 모형에서는 탐색되어 질 수 있는 결함의 기대 값을 θ라고 표현하면 유한 고장 NHPP모형의 평균값 함수와 강도함수는 다음과 같이 표현할 수 있다[1].

$$m(t) = \theta F(t) \ , \ \lambda(t) = \theta F'(t) \quad (5)$$

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

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

이라고 하면 데이터 집합 D_t는 $\{n, x_1 x_2 \cdots x_n\,;t\}$와 같이 구성된다. 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 모형에서의 Θ을 모수공간이라고 표시하면 유한고장 우도함수는 다음과 같이 알려져 있다[9-10].

$$L_{NHPP}(\Theta \mid D_{x_n}) = \left(\prod_{i=1}^{n} \lambda(x_i)\right) \exp(-m(x_n)) \quad (7)$$

NHPP 모형에서 테스트 시점 x_n에서 소프트웨어 고장이 일어난다고 하는 가정 하에서 신뢰구간 $(x_n, x_n + t]$ (단, t는 임무시간(Mission time))사이에서 소프트웨어의 고장이 일어나지 않을 확률인 신뢰도(Reliability) $\hat{R}(t \mid x_n)$는 다음과 같이 됨이 알려져 있다[9].

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

2.2 학습효과를 고려한 강도함수와 누적함수

소프트웨어 테스팅 작업에 있어서 학습효과는 테스팅 관리자에 의해 동일한 혹은 조작 가능한 작업이 될 수 있으므로 이러한 효과들을 어떠한 방법으로 반영하는가는 소프트웨어 신뢰성에 중요한 과정이 된다.

소프트웨어 에러들을 발견하기 위하여 자동 에러 탐색요인(Autonomous errors-detected factor)γ 과 학습요인(Learning factor) η이 포함된 영향요인들(Influential factors)이 고려될 수 있다. 따라서 $f(t)$을 t시점에서 에러를 발견될 확률을 나타내는 강도함수이고 $F(t)$을 $(0, t]$ 시점까지의 누적함수라고 가정하면 영향요인들을 고려한 모형은 다음과 같이 표현된다[8].

$$f(t) = (\gamma + \eta F(t))\,(1 - F(t)) \quad (9)$$

단, $\gamma > 0$, $\eta > 0$.

(9)식에서 자동 에러 탐색요인 γ는 사전에 알지 못하지만 테스팅 과정에서 테스팅 관리자가 자동적으로 에러를 발견하는 요인이지만 학습요인 η 은 과거에 발견된 에러패턴을 바탕으로 세밀하게 에러를 발견하기 위하여 테스팅 관리자가 설정해놓은 요인을 의미한다.

한편 (9)식을 다음과 같이 위험함수 형태로 변경 할 수 있다.

$$h(t) = (\gamma + \eta F(t)) \quad (10)$$

단, $h(t) = \dfrac{F'(t)}{1-F(t)} = \dfrac{f(t)}{1-F(t)}$

(10)식에서 누적함수와 강도함수는 다음과 같이 수정 가능하다.

$$F(t)=\frac{h(t)-\gamma}{\eta},\ f(t)=F'(t)=\frac{h'(t)}{\eta} \quad (11)$$

Ⅲ. 제안한 NHPP 극값 신뢰모형

이 절에서 극값 분포(Extreme value distribution)[10] 모형에 대하여 제안 하고자 한다. 이 분포는 고장시점을 측정할 때 두터운 꼬리영역을 가지는 수명분포를 명시적으로 고려한다는 점 이외에, 분포의 비대칭성도 효과적으로 고려할 수 있다는 장점을 가진 분포로 알려져 있다. 이 극값 분포의 확률 밀도 함수와 분포 함수는 각각 다음과 같다[11].

$$f(t\,|\,\beta_0,\ \beta_1)=\exp\left(\beta_0+\beta_1 t-\frac{e^{\beta_0+\beta_1 t}-e^{\beta_0}}{\beta_1}\right) \quad (12)$$

$$F(t\,|\,\beta_0,\ \beta_1)=1-\exp\left(-\frac{e^{\beta_0+\beta_1 t}-e^{\beta_0}}{\beta_1}\right) \quad (13)$$

단, 모수 $-\infty<\beta_0<\infty,\ \beta_1>0,\ t\geq 0$

위험함수는(10)식과 (12), (13)식을 관련하면 다음과 같이 유도된다.

$$h(t)=f(t)/(1-F(t))\ =\exp(\beta_0+\beta_1 t) \quad (14)$$

따라서 (14)식과 (11)를 이용하면 다음과 같이 누적 함수와 강도 함수는 다음과 같이 수정 가능하다.

$$F(t)=\frac{\exp(\beta_0+\beta_1 t)-\gamma}{\eta} \quad (15)$$

$$f(t)=\frac{\beta_1\exp(\beta_0+\beta_1 t)}{\eta} \quad (16)$$

유한 고장 NHPP모형의 평균값 함수와 강도함수는 (15)식과 (16)식을 이용하면 각각 다음과 같이 표현 할 수 있다.

$$m(t)=\theta F(t)=\theta\left(\frac{\exp(\beta_0+\beta_1 t)-\gamma}{\eta}\right) \quad (17)$$

$$\lambda(t)=\theta F'(t)=\frac{\theta\beta_1\exp(\beta_0+\beta_1 t)}{\eta} \quad (18)$$

이 경우의 우도함수는 (7)식에 (17)식과 (18)식을 대입하면 다음과 같다.

$$L_{NHPP}(\Theta\,|\,D_{x_n})= \prod_{i=1}^{n}\left(\frac{\theta\beta_1 e^{\beta_0+\beta_1 x_i}}{\eta}\right)\exp\left[-\theta\left(\frac{e^{\beta_0+\beta_1 x_n}-\gamma}{\eta}\right)\right] \quad (19)$$

모수 추정방법은 최우추정법(MLE)을 사용하였고 최우추정법을 이용하기 위한 로그 우도 함수는 (19)식과 관련하여 다음과 같이 유도된다.

$$\ln L_{NHPP}(\Theta\,|\,D_{x_n})= n\ln\theta+n\ln\beta_1-n\ln\eta+n\beta_0+\beta_1\sum_{i=1}^{n}x_i-\theta\left(\frac{e^{\beta_0+\beta_1 x_n}-\gamma}{\eta}\right) \quad (20)$$

본 논문에서는 극값 분포의 특성을 유지하면서도 보다 간결하게 하기 위하여 (20)식에서 형상모수 $\beta_0=0$으로 가정한 모형을 사용하고자 한다. 즉, θ와 β_1에 대하여 편미분하여 다음과 같은 식을 만족하는 $\hat{\theta}_{MLE}$ 와 $\hat{\beta}_{1\,MLE}$을 수치 해석적 방법으로 추정할 수 있다[12].

$$\frac{\partial\ln L_{NHPP}(\Theta\,|\,D_{x_n})}{\partial\theta}=\frac{n}{\theta}-\left(\frac{e^{\beta_0+\beta_1 x_n}-\gamma}{\eta}\right)=0 \quad (21)$$

$$\frac{\partial\ln L_{NHPP}(\Theta\,|\,D_{x_n})}{\partial\beta_1}=\frac{n}{\beta_1}+\sum_{i=1}^{n}x_i-\frac{\theta\ x_n\ e^{\beta_0+\beta_1 x_n}}{\eta}=0 \quad (22)$$

Ⅳ. 효율적 모형을 위한 모형 비교

최근에 모형에 대한 효율성을 조사하기 위한 기준으로서 MSE(평균자승오차, Mean square error)를 사용한다[8].

평균자승오차는 실제 관찰 값과 예측 값에 대한 차이를 측정 하는 도구로서 다음과 같이 정의된다.

$$MSE = \frac{\sum_{i=1}^{n}\left(m(x_i)-\hat{m}(x_i)\right)^2}{n-k} \tag{23}$$

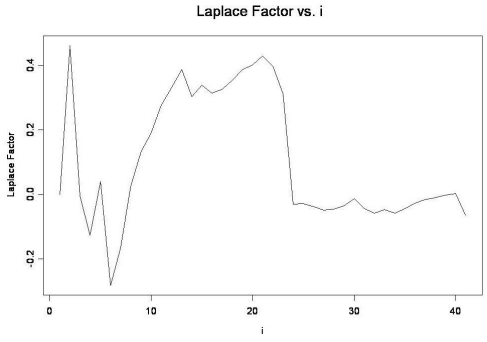
단, $m(x_i)$은 시간$(0,\ x_i]$까지 나타난 에러들의 누적함수를 의미하고 $\hat{m}(x_i)$는 x_i시점까지 평균값 함수로부터 추정된 에러의 누적개수를 의미한다. 그리고 n은 관찰 값의 수이고, k는 모수의 수를 의미한다.

Ⅴ. 수치적인 예

이 장에서 Musa[12]가 군사관련 소프트웨어에 대한 고장간격 자료(S27)를 이용하여 학습요인에 관한 특성을 분석하고자 한다.

이 자료는 1197.945 시간단위에 41번의 고장이 발생된 자료이며 <표 1>에 나열되어 있고 제시 하는 신뢰 모형들을 분석하기 위하여 우선 자료에 대한 추세검정이 선행되어야 한다[12].

추세분석에는 일반적으로 라플라스 추세검정(Laplace trend test)을 이용한다. 이 검정을 실시한 결과 <그림 1>



<그림 1> 라플라스 추세 검정

<표 1> 고장 간격자료

Failure number	Failure Time(hours)	Failure Interval (hours)	$failure\ time \times 10^{-2}$
1	5.649	5.649	5.649×10^{-2}
2	8.92	3.271	8.92×10^{-2}
3	20.29	11.37	20.29×10^{-2}
4	29.955	9.665	29.955×10^{-2}
5	34.715	4.76	34.715×10^{-2}
6	75.95	41.235	75.95×10^{-2}
7	78.171	2.221	78.171×10^{-2}
8	78.625	0.454	78.625×10^{-2}
9	83.022	4.397	83.022×10^{-2}
10	89.114	6.092	89.114×10^{-2}
11	89.804	0.69	89.804×10^{-2}
12	92.86	3.056	92.86×10^{-2}
13	93.66	0.8	93.66×10^{-2}
14	110.655	16.995	110.655×10^{-2}
15	111.988	1.333	111.988×10^{-2}
16	122.545	10.557	122.545×10^{-2}
17	127.045	4.5	127.045×10^{-2}
18	128.712	1.667	128.712×10^{-2}
19	128.99	0.278	128.99×10^{-2}
20	131.768	2.778	131.768×10^{-2}
21	131.829	0.061	131.829×10^{-2}
22	141.712	9.883	141.712×10^{-2}
23	164.212	22.5	164.212×10^{-2}
24	342.85	178.638	342.85×10^{-2}
25	356.144	13.294	356.144×10^{-2}
26	399.144	43	399.144×10^{-2}
27	446.494	47.35	446.494×10^{-2}
28	476.644	30.15	476.644×10^{-2}
29	497.144	20.5	497.144×10^{-2}
30	497.661	0.517	497.661×10^{-2}
31	591.161	93.5	591.161×10^{-2}
32	665.644	74.483	665.644×10^{-2}
33	686.444	20.8	686.444×10^{-2}
34	765.944	79.5	765.944×10^{-2}
35	772.977	7.033	772.977×10^{-2}
36	774.944	1.967	774.944×10^{-2}
37	791.561	16.617	791.561×10^{-2}
38	815.978	24.417	815.978×10^{-2}
39	837.145	21.167	837.145×10^{-2}
40	861.945	24.8	861.945×10^{-2}
41	1197.945	336	1197.945×10^{-2}

에서 보여 주듯이 라플라스 요인(Factor)이 -2와 2사이에 존재함으로서 신뢰성장(Reliability growth) 속성을 나타내고 있다. 따라서 이 자료를 이용하여 신뢰도를 추정하는 것이 가능하다[2].

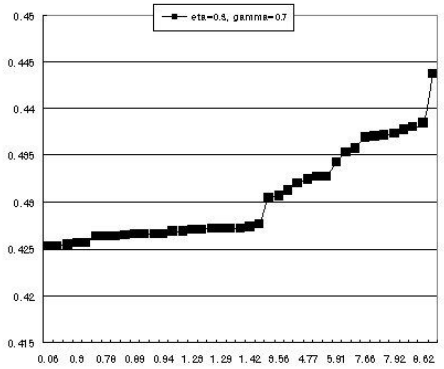
학습효과를 고려한 소프트웨어 신뢰성 모형의 모수 추정은 모수추정을 용이하게 하기 위하여 원 데이터를 변수변환($failure\ time \times 10^{-2}$)하여 최우추정법을 이용하였다.

<표 2> 각 모형의 모수 추정값

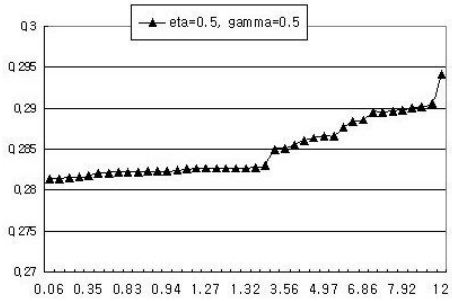
Model		MLE		모형 비교
η	γ	$\hat{\theta}_{MLE}$	$\hat{\beta}_{1_{MLE}}$	MSE
0.1	0.9	30.66	2.983×10^{-3}	11402.20
0.2	0.8	33.98	3.380×10^{-3}	3365.555
0.3	0.7	33.80	3.563×10^{-3}	1686.203
0.4	0.6	36.80	3.670×10^{-3}	981.3356
0.5	0.5	37.55	3.746×10^{-3}	811.0953
0.6	0.4	38.05	3.792×10^{-3}	650.1410
0.7	0.3	38.42	3.838×10^{-3}	550.4616
0.8	0.2	38.71	3.863×10^{-3}	508.1629
0.9	0.1	38.83	3.893×10^{-3}	489.6116

학습요인 η과 자동 에러 탐색요인 γ은 0.1부터 0.9까지 고정 시킨 비선형 방정식의 계산방법은 수치 해석적 방법인 이분법(Bisection method)을 사용하였다. 이러한 계산은 초기값을 0과 1을, 허용 한계(Tolerance for width of interval)는 10^{-10} 을 주고 수렴성을 모니터링하면서 충분한 반복 횟수인 100번을 C-언어를 이용하여 모수 추정을 수행하였다. 각 모형에 대한 모수의 추정 값들의 결과와 MSE(평균자승오차) 값들이 <표 2>에 요약되었다.

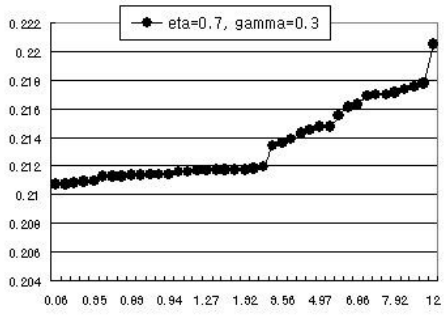
이 표에서 η이 증가하고 γ이 감소하는 경우에 MSE 값들이 작게 나타나 점점 효율적인 모형으로 변해가고 있음을 확인할 수 있다. 즉 학습요인 η 이 자동 에러 탐색요인 γ보다 클수록 효율적인 모형으로 나타나고 있다. 그리고 <그림 2>, <그림 3> 그리고 <그림 4>에서도 강도함수는 비감소패턴을 보이고 있으며 학습요인 η이 증가할수록 강도함수는 낮게 나타남을 확인할 수 있다. <그림 5> 에서는 임무시간에 대한 신뢰도 그림에서도 η이 증가할수록 신뢰도가 상승으로 나타나고 있다.



<그림 2> η=0.3, γ=0.7인 경우의 강도함수



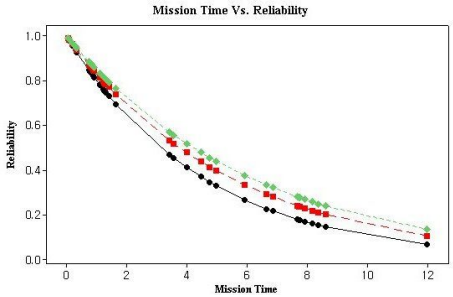
<그림 3> η=0.5, γ=0.5인 경우의 강도함수



<그림 4> η=0.7, γ=0.3인 경우의 강도함수

Ⅵ. 결론

대용량 소프트웨어가 수정과 변경하는 과정에서 결점의 발생을 거의 피할 수 없는 상황이 현실이다. 따라서 소프트웨어 관리자들이 소프트웨어 및 검사 도구에 효율적인 학습 과정을 이용한 NHPP 소프트웨어 모형에 대하여 연구하였다. 사전에 알지 못하지만 자동적으로 발견되는 에러를 고려한 영향요인과 사전경험에 의하여 세밀하게 에러를 발견하기 위하여 테스팅 관리자가 설정해 놓은 요인인 학습효과의 특성에 대한 문제를 비교 제시하였다.



<그림 5> 임무시간에 대한 신뢰도

본 연구에서는 고장시점을 측정할 때 두터운 꼬리영역을 가지는 수명분포를 명시적으로 고려한다는 점 이외에, 분포의 비대칭성도 효과적으로 고려할 수 있다는 장점을 가진 극값분포를 유한 고장 NHPP 소프트웨어 수명분포로 적용하여 분석한 결과 학습요인이 자동 에러 탐색요인보다 큰 경우가 대체적으로 효율적인 모형임을 확인할 수 있어서 이 분야에서 효율적 모형으로 선택될 수 있음을 보여주고 있다.

경우에 따라서는 왜도와 첨도 측면에서 효율적인 카파분포, 지수화지수분포 등 업데이트된 분포에 대한 적용 문제를 비교 분석하는 연구도 가치 있는 일이라 판단되고 이 연구를 통하여 소프트웨어 개발자들은 수명분포에 의존한 학습요인을 파악하는데 어느 정도 도움을 줄 수 있으리라 사료 된다.

참고문헌

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

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

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

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

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

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

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

[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, 2008, pp. 1410-1421.

[9] J. F. Lawless. Statistical Models and Methods for Lifetime Data. John Wiley & Sons, New York, 1981.

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

[11] 김희철, "극값 분포 특성을 가진 소프트웨어 신뢰성 보증 모형에 관한 연구," 한국 통신학회 논문지, 34권6호, 2009, pp. 623-629.

[12] K. Kanoun and 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 ■

김 희 철
Kim, Heel Cheul

2005년 3월~현재 남서울대학교 산업경영공학과 교수
1998년 8월 동국대학교 통계학과(이학박사)
1992년 2월 동국대학교 통계학과(이학석사)

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

논문접수일 : 2011년 4월 2일
수 정 일 : 2011년 5월 4일(1차), 5월 18일(2차)
게재확정일 : 2011년 5월 24일