http://dx.doi.org/10.14400/JDPM.2013.11.11.335

로그형 관측고장시간에 근거한 결함 발생률을 고려한 소프트웨어 비용 모형에 관한 비교 연구

김경수*, 김희철**
백석문화대학교 인터넷 정보학부*, 남서울대학교 산업경영공학과**

The Comparative Software Cost Model of Considering Logarithmic Fault Detection Rate Based on Failure Observation Time

Kyung-Soo Kim*, Hee-Cheul Kim**
Dept. of Internet information, BaekSeok Culture University*
Dept. of Industrial & Management Engineering, Namseoul University**

요 약 본 연구에서는 소프트웨어 제품 테스팅 과정에서 관측고장시간에 근거한 로그형 결함 발생률을 고려한 소프트웨어 신뢰성 비용 모형에 대하여 연구 하였다. 신뢰성 분야에서 많이 사용되는 Goel-Okumoto모형을 이용한 새로운 로그 형 결함 확률을 반영한 문제를 제시하였다. 수명분포는 유한고장 비동질적인 포아송과정을 이용하고 모수 추정법은 최우 추정법을 이용 하였다. 따라서 본 논문에서는 로그형 결함 발생률을 고려한 소프트웨어 비용모형 분석을 위하여 소프트웨어 고장 시간간격 자료를 적용하여 비교 분석하였다. 이 연구를 통하여 소프트웨어 개발자들은 방출최적시기를 파악 하는데 어느 정도 도움을 줄 수 있을 것으로 사료 된다.

주제어 : 로그형 결함 탐색율, 소프트웨어 방출정책, 비동질적 포아송 과정, 비용모형, Goel-Okumoto 모형

Abstract In this study, reliability software cost model considering logarithmic fault detection rate based on observations from the process of software product testing was studied. Adding new fault probability using the Goel-Okumoto model that is widely used in the field of reliability problems presented. When correcting or modifying the software, finite failure non-homogeneous Poisson process model. For analysis of software cost model considering the time-dependent fault detection rate, the parameters estimation using maximum likelihood estimation of inter-failure time data was made. In this research, Software developers to identify the best time to release some extent be able to help is considered.

Key Words : Logarithmic Fault Detection Rate, Software Release Policy, NHPP, Cost Model, Goel-Okumoto Model

Received 2 September 2013, Revised 25 September 2013
Accepted 20 November 2013
Corresponding Author: Hee-Cheul Kim(Namseoul University)
Email: kim1458@nsu.ac.kr

ISSN: 1738-1916

1. 서론

소프트웨어 고장으로 인한 컴퓨터 시스템의 고장은 우리 사회에 엄청난 손실을 유발 할 수 도 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 불가결 하다. 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)에 의존한 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 모형(enhanced NHPP) 모형을 제시하였고 Goel 과 Okumoto [2]은 지수적 소프트웨어 신뢰성 모형(exponential software reliability growth model)을 제안 하였다. 이 모형은 결함의 누적수가 S 형태나 지수적 형태(S-shaped or exponential-shaped)를 가진 평균값 함수(mean value function)를 이용하였다. 이러한 모형에 의존한 일반화 모형은 Yamada 와 Ohba [3]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 [4]는 소프트웨어 신뢰도에서 변환점 문제를 제시하였고 Shyur [5]는 변환점을 이용한 일반화한 신뢰도 성장 모형을 제안하였다. Pham와 Zhang[6]는 테스팅 커버리지(coverage)를 측정하여 소프트웨어 안정도를 평가 할 수 있는 소프트웨어 안정도 모형을 제시했다. 비교적 최근에, Huang [7]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 하였고[8] 또한, 대수 선형 위험함수를 이용한 학습과정 특성을 연구하기도 하였다[9].

또한 대부분 연구에서는 테스팅하는 과정에서 새로운 결함이 나타나지 않는다는 가정 즉, 결함 탐색비율을 일정한 상수로 간주하여 분석하는 연구가 대부분이었다[10].

따라서 본 연구에서는 이 분야에서 기본 모형인 지수분포를 이용하여 개발된 Goel-Okumoto모형을 이용하여 테스팅 하는 과정에서 결함 탐색비율을 관측시간에 의존한 로그형 형태를 이용하고 이에 대한 비용 모형을 비교 제시하였다.

2. 관련연구

2.1 유한 고장 NHPP 모형

신뢰도에서 관측시간 $(0, t]$ 사이에 발견된 고장 수 $N(t)$을 모형 화 하는데 비동질적 포아송 과정이 널리 사용하여 왔다. 이 과정(process)에서 강도함수(intensity function) 혹은 고장 발생률(rate of occurrence of failure; ROCOF) $\lambda(t) = dE[N(t)]/dt$ 은 t 에 대한 단조(monotonic)함수로 흔히 가정한다[1]. 이 범주에서 지금까지 알려진 모형들은 Goel-Okumoto 모형, Weibull 모형 그리고 Cox-Lewis 모형 등이 있는데 이모형들에 대한 강도함수는 각 각 시간에 의존한 함수, 멱(power) 함수, 대수 선형(log-linear) 함수를 가정하였다[11].

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

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

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

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

이처럼 시간관련 모형(time domain models)들은 NHPP에 의해서 확률 고장 과정으로 설명이 가능하다.

이러한 NHPP 모형들은 유한 고장 모형과 무한 고장 범주로 분류한다[11]. 유한 고장(finite failure) NHPP 모형들은 충분한 테스트 시간이 주어지면 결함들(faults)의 기대값이 유한 값$\left(\lim_{t\to\infty} m(t)=\theta<\infty \right)$을 가지고 반면에 무한 고장(infinite failure) NHPP 모형들은 무한 값을 가진다고 가정된다. 유한 고장 NHPP 모형에서 충분한 테스트 시간이 주어졌을 때 탐색되어 질 수 있는 결함의 기대값을 θ라고 표현하고 $F(t)$을 분포함수라고 표현하면 유한 고장 NHPP모형의 평균값 함수는 다음과 같이 표현할 수 있다[1][11][12].

$$m(t)=\theta F(t) \tag{3}$$

(3)식으로 부터 강도함수(failure intensity)$\lambda(t)$는 다음과 같이 유도된다.

$$\lambda(t)=\theta F'(t) \tag{4}$$

또한, 시간 $(0,t]$까지 조사하기 위한 시간 절단(time truncated)모형은 n번째 까지 고장시점 자료를

$$x_k=\sum_{i=1}^{k} t_k \ (k=1,2,\cdots,n\,;0\le x_1\le x_2\le\cdots\le x_n) \tag{5}$$

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 Θ를 모수공간이라고 표시하면 우도함수는 다음과 같이 알려져 있다[11][12].

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

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

2.2 소프트웨어 개발 비용모형

소프트웨어 비용 모형은 다음과 같이 정의 된다[10].

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

단, E: 소프트웨어 개발주기의 예상 총비용

E_1: 소프트웨어 설계 및 초기 소프트웨어 개발의 비용(분석 데이터, 소프트웨어 개발 전문가의 수 , CPU 시간 등)

E_2: 단위 시간당 소프트웨어 테스팅 비용(상수) 즉, $E_2=C_2\times t$ (단, C_2는 단위 시간당 비용이고 t는 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: 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 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 최적 소프트웨어 방출시간 (t)는 다음과 같이 유도 할 수 있다.

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

3. 제안한 로그형 관측고장시간에 근거한 결함발생률을 고려한 NHPP Goel-Okumoto 모형

이 분야에서 기본 모형인 Goel-Okumoto모형은 유한 고장 상황에서 고장의 원인이 되는 결함의 기대 값을 θ라고 표현하고 결함 탐색율을 β라고 하면 NHPP와 결함 탐색율 β는 고정상수로 간주하여 정의되었지만[8] 본 연구에서는 결함 탐색율 β를 관측시간에 의존한 함수 $\beta(t)$로 간주하여 다음과 같은 로그형 패턴을 제시하고자 한다[10].

$$\beta(t_i)=\beta\ln(t_i) \tag{9}$$

단, $t_i = x_i - x_{i-1},\ i=1,\cdots,\ n(t_i$ 는 고장 간격시간)

Goel-Okumoto 모형은 결함 당 고장발생 시간의 분포(수명 분포)를 지수분포를 가정하였고 결함들(faults)의 기대 값이 θ이고 소프트웨어 결함(fault)당 고장 발생률β는 일정한 형태를 가지며 평균값 함수와 강도함수는 각각 다음과 같이 알려져 있다[2].

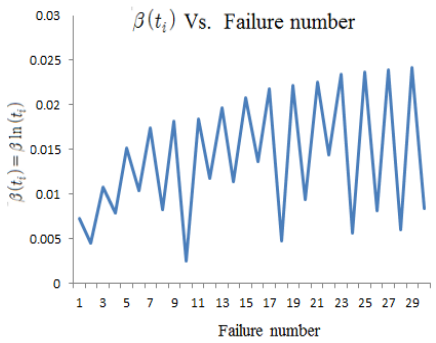
$$m(x|\theta,\beta)=\theta(1-e^{-\beta x})\ (\theta>0,\ \beta>0) \tag{10}$$

$$\lambda(x|\theta,\beta)=\theta\beta e^{-\beta x} \tag{11}$$

따라서 (그림 1)에서 보여 주듯이 결함 탐색율 β를 관측시간에 의존한 함수 $\beta(t)$로 간주하여 로그형 패턴을 반영한 평균값 함수와 강도함수는 각 각 다음과 같이 표현 할 수 있다.

$$m(x|\theta,\beta)=\theta(1-e^{-\beta\ln(t)x})\ (\theta>0,\ \beta>0) \tag{12}$$

$$\lambda(x|\theta,\beta)=\theta\beta\ln(t)e^{-\beta\ln(t)x} \tag{13}$$



[Fig. 1] Logarithmic Fault Detection Rate

최종고장시점 x_n 으로 대치하고 (6)식에 (12)식과 (13)식을 이용하면 우도함수는 다음과 같이 표현 할 수 있다.

$$L_{NHPP}(\Theta|\ D_{x_n}) = \left(\prod_{i=1}^{n}\theta\beta\ln(t_i)e^{-\beta\ln(t_i)x_i}\right)\exp(-\theta(1-e^{-\beta\ln(t_n)x_n})) \tag{14}$$

따라서 고장절단모형에서의 모수 θ 와 β 에 관한 편미분식은 다음과 같이 유도 할 수 있다[11][12].

$$\frac{\partial\ln L(\Theta|\ \underline{x})}{\partial\theta}=\frac{n}{\theta}-1+e^{-\beta\ln(t_n)x_n}=0 \tag{15}$$

$$\frac{\partial\ln L(\Theta|\ \underline{x})}{\partial\beta}= \tag{16}$$

$$\frac{n}{\beta}-\sum_{i=1}^{n}\ln(t_i)x_i-\theta\ln(t_n)\ x_n\ e^{-\beta x_n}=0$$

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

각 모수에 대한 최우추정량 $\hat{\theta}_{MLE}$ 와 $\hat{\beta}_{MLE}$ 은 다음 식을 만족한다[12].

$$\frac{n}{\theta}=1-e^{-\beta\ln(t_n)x_n} \tag{17}$$

$$\frac{n}{\beta}=\sum_{i=1}^{n}\ln(t_i)x_i+\theta\ln(t_n)\ x_n\ e^{-\beta x_n} \tag{18}$$

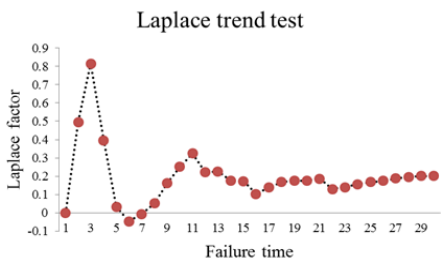
4. 소프트웨어 고장 자료 분석 및 방출 시기 분석

이 장에서 소프트웨어 고장 간격 시간 자료[13](failure interval time data)를 가지고 제시하는 신뢰모형들을 분석하고자 한다. 이 자료의 고장 시간은 18.735시간단위에 30번의 고장이 발생된 자료이며 <표 1>에 나열 되어 있다. 또한 제시하는 신뢰 모형들을 분석하기 위하여 우선 자료에 대한 추세 검정이 선행 되어야 한다[14]. 추세 분석에는 일반적으로 라플라스 추세 검정(Laplace trend test)을 사용한다. 이 검정을 실시한 결과(그림 2)에서 라플라스 추세 검정의 결과는 라플라스 요인(factor)이 -2와 2사이에 존재함으로서 신뢰성장(reliability growth) 속성을 나타내고 있다. 따라서 이 자료를 이용하여 신뢰 성장모형을 제시하는 것이 효율적임을 시사하고 있다[14]. 결함 탐색율 $\beta(t_i)=\beta\ln(t_i)$ 패턴을 적용한 Goel-Okumoto 모형에 대한 모수 추정은 최우추정법을 이용하고 비선형 방정식의 계산방법은 수치 해석적 기본 방법인 이분법(bisection method)을 사용하였다. 그리고 모수추정을 용이하게 하기 위하여 원 자료를 변수변환($failure\ time\times10$)하여 사용하였다. 이러한 계산은 초기 값을 0.001와 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 $\hat{\theta}_{MLE}=32.2137$, $\hat{b}_{MLE}=0.0046$으로 추

정 되었다.

〈Table 1〉 Failure time data

Failure number	Failure time (hours)	Failure time (hours)×10	Failure interval (hours)
1	0.479	4.79	4.79
2	0.745	7.45	2.66
3	1.022	10.22	2.77
4	1.576	15.76	5.54
5	2.61	26.1	10.34
6	3.559	35.59	9.49
7	4.252	42.52	6.93
8	4.849	48.49	5.97
9	4.966	49.66	1.17
10	5.136	51.36	1.7
11	5.253	52.53	1.17
12	6.527	65.27	12.74
13	6.996	69.96	4.69
14	8.17	81.7	11.74
15	8.863	88.63	6.93
16	10.771	107.71	19.08
17	10.906	109.06	1.35
18	11.183	111.83	2.77
19	11.779	117.79	5.96
20	12.536	125.36	7.57
21	12.973	129.73	4.37
22	15.203	152.03	22.3
23	15.64	156.4	4.37
24	15.98	159.8	3.4
25	16.385	163.85	4.05
26	16.96	169.6	5.75
27	17.237	172.37	2.77
28	17.6	176	3.63
29	18.122	181.22	5.22
30	18.735	187.35	6.13

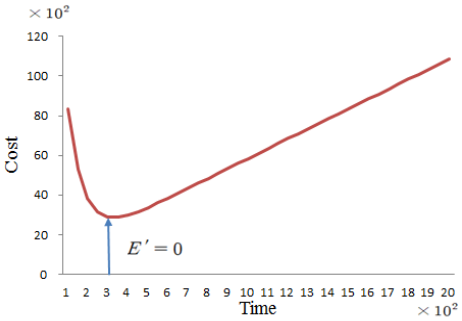


[Fig. 2] Laplace trend test

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

(단, $t = x_{30} = 187.35,\ \beta(t_{30}) = \beta \ln(t_{30}) = 0.00834$)

(가정 1) $c_2 = 5\$,\ c_3 = 25\$,\ c_4 = 800\$,\ E_1 = 50\$$
$t' = 500$



[Fig. 3] The curve under the condition of (Assumption 1)

(그림 3)에서 보는 바와 같이 제안 된 모델의 비용 성장 곡선은 처음에는 감소하다가 점차 증가하는 추세를 보이고 있다. 소프트웨어 시스템의 잔여 결함의 수는 결함의 제거하는 과정에서 점점 줄어들게 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.

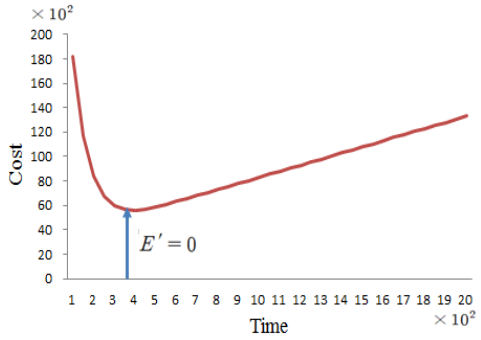
그러나 일정시간 이후의 단계에서 소프트웨어에 남아 있는 결함의 수는 적기 때문에 이 테스트 단계에서 결함을 검출 시간이 상대적으로 길고 오류를 제거하는 비용은 운용 단계에서 상대적으로 높기 때문에 비용 곡선은 시간이 흐름에 따라 지속적으로 증가하게 된다[10].

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.

(가정 2) $c_2 = 5\$,\ c_3 = 25\$,\ c_4 = 1600\$,\ E_1 = 50\$$
$t' = 500$

(가정 1)에서 다른 가정은 동일하고 소프트웨어 출시된 이후에 소프트웨어 운영 단계에서 사용자가 관찰되는

결함수정 비용 C_4를 800$에서 1600$으로 증가한 경우는 (그림 4) 에 나타내었다.

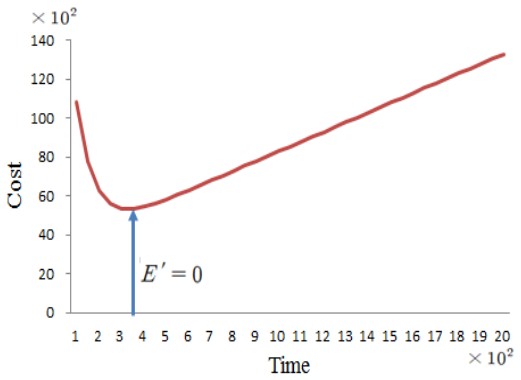


[Fig. 4] The curve under the condition of (Assumption 2)

(그림 3) 과 (그림 4)를 비교 하였을 때 소프트웨어 출시 된 이후에 소프트웨어 운영 단계에서 사용자가 관찰되는 결함수정 비용이 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,

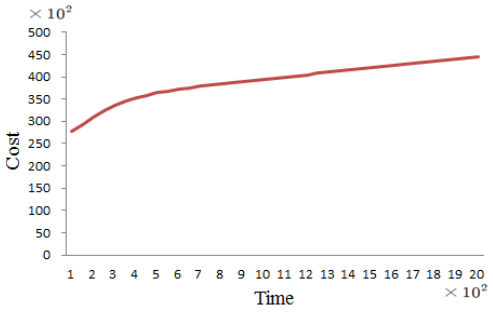
(가정 3) $c_2 = 5\$,\ c_3 = 25\$,\ c_4 = 1600\$,\ E_1 = 50\$$
$t' = 1000$

(가정 1)에서 다른 가정은 동일하고 소프트웨어 시스템을 출시 한 후 운영 및 소프트웨어를 유지할 수 있는 시간 t' 를 500에서 1000으로 증가한 경우는 (그림 5) 에 나타내었다. (그림 3)과 (그림 5)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.



[Fig. 5] The curve under the condition of (Assumption3)

(가정 4) $c_2 = 5\$,\ c_3 = 500\$,\ c_4 = 1600\$,$
$E_1 = 50\$,\ t' = 500$



[Fig. 6] The curve under the condition of (Assumption 4)

(가정 1)에서 다른 가정은 동일하고 테스팅 과정에서 하나의 결함을 제거한 비용 C_3을 25에서 500으로 증가한 경우는 (그림 6) 에 나타내었다. (그림 3) 과 (그림 6) 을 비교 하였을 때 소프트웨어 비용이 지속적으로 증가함을 알 수 있다. 그 이유는 테스팅 과정에서 하나의 결함을 제거한 비용을 증가 시켰기 때문이다. 소프트웨어 생명주기에 있어서 소프트웨어 개발 총비용은 결함 탐색 시간과 테스팅 단계에서 결함 수정비용에 의해서 결정되기 때문에 시간의 증가함에 따라 점점 증가하게 된다. 이런 경우는 소프트웨어 사용자의 피드백에 따라 오류를 관찰하고 수정 할 수 있는 노력을 해야 한다. 그러나 이런 경우는 실제 상황에서는 드문 경우이다.

따라서 기본 모형에 비교하여 소프트웨어 출시 된 이

후에 소프트웨어 운영 단계에서 사용자가 관찰되는 결함 수정 비용 증가한 경우와 운영 및 소프트웨어를 유지할 수 있는 시간을 증가한 경우도 소프트웨어 최적 방출시간이 연기되므로 소프트웨어 방출시기 이후에 결함을 감소시킬 수 있도록 운영단계보다 테스팅 단계에서 가능한 결함들을 제거해야 한다. 그리고 테스팅 과정에서 하나의 결함을 제거한 비용을 증가한 경우는 소프트웨어 사용자의 피드백에 따라 오류를 관찰하고 수정 할 수 있는 노력을 해야 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 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 오류 수정 비용은 운용 단계에서 남아있는 모든 오류를 제거하는 비용보다 낮다는 것을 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 판단되고 이 연구를 통하여 소프트웨어 개발자들은 방출최적시기를 파악 하는데 어느 정도 도움을 줄 수 있으리라 사료 된다.

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

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

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

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

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

[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, pp. 1410 - .1421, 2008.

[9] Kim Hee Cheul and Shin Hyun Cheul, The Comparative Study for NHPP Software Reliability Model based on the Property of Learning Effect of Log Linear Shaped Hazard Function, Convergence Security Journal, Vol. 12, No. 3, pp. 19-26, 2012.

[10] Ye Zhang and Kaigui Wu, Software Cost Model Considering Reliability and Time of Software in Use, Journal of Convergence Information Technology (JCIT), Volume 7, Number 13, pp. 135-142, 2012.

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

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

[13] Y. HAYAKAWA and G. TELFAR., Mixed Poisson Type Processes with Application in Software Reliability, Mathematical and Computer Modeling, 31, pp. 151-156, 2000.
[14] 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.
[15] Changjie Ma, Guochang Gu, Jing Zhao, A Novel Software Reliability Assessment Approach based on Neural Network in Network Environment, IJACT: International Journal of Advancements in Computing Technology, Vol. 4, No. 1, pp. 136 - 144, 2012.

김 경 수(Kim, Kyung Soo)

- 2001년 8월 : 순천향대학교 전산학과(공학 박사)
- 2005년 3월 ~ 2007년 2월 : VCU DBLab Visiting Scholar
- 1998년 3월 ~ 현재 : 백석문화대학교 인터넷정보학부 교수
- 관심분야 : 소프트웨어신뢰성 공학, 정보보안, 웹 프로그래밍
- E-Mail : kkskim@bscu.ac.kr

김 희 철(Kim, Hee Cheul)

- 1992년 2월 : 동국대학교 통계학과 (이학 석사)
- 1998년 8월 : 동국대학교 통계학과 (이학박사)
- 2005년 3월 ~ 현재 : 남서울대학교 산업경영공학과 교수
- 관심분야 : 소프트웨어신뢰성 공학, 전산 통계
- E-Mail : kim1458@nsu.ac.kr