다항지수 신뢰도 함수

최규식

건양대학교 의공학과
che@konyang.ac.kr
충남 논산시 내동 26

요약

다항지수 신뢰도 함수(multinomial-exponential reliability function ; MERF)는 소프트웨어의 고장/수정 공정을 세밀하게 수행하는 중에 개발되는 관계에 있다. 후에 MERF는 좀더 매우 단순화한 지수 신뢰도 함수(exponential reliability function ; EARF)로 근사화되는 공정을 거치게 된다. 이는 MERF의 특성을 대부분 가지고 있어서 두 개의 함수가 하나의 신뢰도 함수로 단일화되도록 한다. 신뢰도 모델 MERF/EARF는 소프트웨어 고장 공정을 NHPP로, 수정공정을 다항분포로 고려한다. 이 모델은 두 개의 공정 모두가 통계적 독립인 것으로 간주한다.

본 논문에서는 모델의 이론적인 기준, 수학적 특성, 소프트웨어 신뢰도에의 응용을 검토한다. 이는 물리적인 시스템을 검사하고 유지보수하는 선도적인 모델응용이다. 본 논문에는 소프트웨어 신뢰도 분석에 응용하는 하나의 수치 예를 포함한다.

키워드
리틀우드, MERF, EARF, TBF,

1. 서론

다항지수 신뢰도 함수(MERF)는 ESREL'94에서 도입되었으며, 이의 지수 근사치(exponential approximation ; EARF)는 ESREL'97에 도입되었다. MERF는 소프트웨어 신뢰도 탐사의 결과로서 개발된 것이다. MERF와 EARF는 모두가 소프트웨어 신뢰도 모델이며, MERF는 이론적인 배경을 가진 모델로서, EARF는 이의 실제 구현이다. MERF/EARF 모델의 소개로서 소프트웨어 신뢰도의 특수한 형태를 개략적으로 설명한다.

소프트웨어 신뢰도 모델은 주어진 환경 및 주어진 시간 동안 컴퓨터 프로그램이 고장 없이 운전될 확률이며, 이 환경이란 입력변수와 연결되어 있고 프로그램 고장은 출력변수와 연관되어 있다. 입력변수는 프로그램 외부에 있는 어떠한 데이터 항목이며, 프로그램이 그 기능중의 하나를 실행할 때 프로그램이 사용하는 것이다. 프로그램과 관련된 입력변수 집합은 입력상태를 식별하여 특수 목적의 프로그램실행을 식별하기도 한다. 입력상태 집합과 그발생확률은 프로그램의 환경이나 프로그램의 운영프로필을 결정한다. 출력변수는 프로그램 외부에 있는 어떠한 데이터 항목이며, 프로그램 실행에 의해서 설정된다. 출력상태는 프로그램 실행으로 생기는 출력변수값의 완벽한 집합이다. 예를 들어 항공 예약 시스템의 입력상태는 변수, 출발지, 기착지 항공라인, 일자 및 항공편의 특수한 값일 수 있으며, 출력상태는 승객 티켓에 인쇄된 특수한 변수값일 수도 있다. 고장은 프로그램요건과 동떨어진 출력상태이다. 그러므로, 고장은 프로그램 실행을 함축한다. 고장은 프로그램이 어떤 특수한 조건에서 운영될 때 프로그램의 결함에 의해서 발생된다. 결함은 프로그램상의 흠이다. 흠이 있거나 누락되거나 또는 여분으로 있는 명령이나 지침 등이다. 그리고 모든 고장이 반복되는 것을 피하기 위해 수정, 추가, 또는 제거되어야 할 모든 명령들을 포함해야 한다. 프로그램의 흠이나 결함은 인간의 실수, 프로그램 개발기간 및 테스트기간 동안의 개발활동자들 간의 대화부족에서 발생된다.(고객/설계자/프로그래머/테스터) 여기에는 응용분야의 지식부족, 설계기법과 프로그램 언어의 지식부족, 그리고 최종적으로 문제분석이 완전하지 못한 것 등이다.

2. 적용

소프트웨어 신뢰도 정의는 아래와 같은 표현을 통하여 소프트웨어 프로그램 테스트단계에적용할 수 있다.

$$R(x|k) = \Pr[X_{k+1} \geq x|k] = 1 - F(x|k) \quad (1)$$

여기서, x는 k번째 고장이 발생한 후 고장 없이 프로그램이 실행되는 시간을 의미한다. X_{k+1}은 k번째와 k+1번째 고장 사이의 시간간격(TBF)을 나타내는 무작위변수이며, F(x|k)는 주어진 k개의 고장에서 누적고장분포함수이다. {M(t), t≥0}를 계수공정이라 하고 여기서 M(t)는 시각 t에서의 고장의 수를 표시한다. 공정누적고장강도 W(t)는 시간 간격(0,t] 동안의 고장평균치인 것으로 정의한다. 고장발생률 함수(ROCOF)이나 공정고장강도 w(t)는 W(t)의 유도치(미분치)이며 시각 t에서의 단위시간당 고장의 평균고장치를 나타낸다. 두 개또는 그 이상의 동시 고장확률을 무시할 수 있다면 ROCOF 함수는 시각 t에서의 단위시간당 고장확률을 의미한다. 계수공정 {M(t), t≥0}는 ROCOF 함수w(t)이다. ROCOF함수가 시간종속적일 때 계수공정이 비동차(비정지)인 것으로 일컫는다. 시스템을 보수하거나 수정하지 못한다면 고장률(위험률)은 확률분포주요 파라미터이다. 시스템이 시각 t까지 운전되었다고 했을 때 시각 t에서 단위시간당 첫 번째이자 단 하나의 시스템 고장이 발생할 확률인 것으로 정의한다. 고장률은 항목의 수명테스트데이타에서 얻는다.

3. 모델

3.1 가정

MERF는 다음과같은가정을필요로한다.

1. 고장과 수정공정은 독립적이다.

2. 고장공정{M(t), t≥0}은 ROCOF 함수w(t)를 가진 비동차포아송 공정(NHPP)이다. 그러므로,

- 시각 t=0에서는 프로그램 고장이 없어서 M(0)=0이다.

- 고장공정은 독립증가이다. 이는 공정의 미래상태가 현재의 상태에만 의존한다는 것을 의미한다.(마코프특성)

- 시각 t에서의 단위시간당 하나의 고장이 발생할 확률은 w(t)이다.

- 두 개 이상의 고장이 동시에 일어날 확률은 무시한다.

3. ROCOF함수 w(t)는 시각 t에서 프로그램의 결함수 N(t)에 비례한다.

-고장 X1, X2, …, Xk+1, …, Xn 사이의 시간은 고장률 h(x|k)들을 가진 독립 무작위변수이다. k고장 후 시각 x에서 매 시간당 k+1 고장이 발생할 확률은 h(x|k)이며, 이는 무작위 변수 Xk+1의 고장률 정의에 따른 것이며, 또한, w(t)이기도 하다. 이 w(t)는 프로그램 테스트 시작시부터 측정되는 시각 t에서의 매 시간당 고장확률을 나타내는 공정 ROCOF함수이다.

결국,

$$h(x|k) = w(t_k + x) \quad (2)$$

이다. 그래서 ROCOF 함수는 고장시간 간격 동안 일정하며, Xk+1 누적분포함수는 지수함수적이다. 그러므로,

$$h(x|k) = \lambda_k = \lambda_0 N(t) \quad (3)$$

이고 이 중 λ0는 결함당 고장률을 나타낸다.

- 젤린스키 - 모란다 모델은 1972년에 가설 3을 도입하였다. 이 모델은 프로그램결함이 동일하게 시스템을 이용불능도에 공헌을 한다고 하고 또 일단 고장이 발생하면 완벽하게 결함을 제거하는 것(완벽한 수정)으로 가정하였으므로 비난을 받아왔다. 그럼에도 불구하고 무사가 개발한 기본실행시간모델에서 가설3을 채택하였다.

-리틀우드와 베럴 모델은 상기 기준에 대해서 부분적으로 응답하였다. 이 베이시안 모델은 성공적인 ROCOF가 감소 무작위 변수 집합을 형성한다는 것을 고려하고, 또한 결함을 완벽하게 수정하는 것으로 생각하였다. 결국, 운영프로파일이 일정하다면 최대 고장률을 가진 결함이 맨 처음에 필요하다.그,러나, 운영파일이 일정하지 않으면 결함 가중치를 보상할 가능성이높다. 그럼에도 불구하고 무작위적으로 선정된 입력의 영향을 결정하기 위해 이미 수행된 실험에서는 결함검출공정에서 격렬한 변화를 나타내고 있으며, 그렇다고 판단하기에는 너무나 시기상조이다.

4. 결함수정시간은 중요하지 않은 것으로 간주한다.

5. 고장을일으키는 결함은 3개의 시간독립적인 수정가능성에따라 고정것으로한다.

- p : 결함을수정할 확률- 완전수정

- q : 시스템을 고장전의 결함수와 동일한 결함수로

있을 확률- 불완전수정

- r: 마지막고장을 일으킨 결함을 수정하려고 노력할 때 새로운 결함이 도입될 확률- 결함정 추가

여기에서

$$p+q+r=1 \quad (4)$$

이므로, 논문의 모델수정공정은 다중확률분포함수를 가진다. 시각 t에서 시스템 고장의 수가 시각 t에서 제거된 결함의 수와 동일하다는 젤린스키와 모란다 모델, 리틀우드 모델에서 심사숙고한 결함고정 결정론적 특성 대신 논문의 모델 수정공정의 확률특성을 강조해야만 한다.

3.2 MERF 개발

시각 t에서의 프로그램 결함수 N(t)는

$$N(t)=n_0-[k-I_k-2J_k]=n_0-k+I_k+2J_k \quad (5)$$

여기서, n0는 테스트단계를 시작할 때의 결함의 수를 나타내며, k는 시각 t까지 발생된 고장의 수를, Ik와 Jk는 각각 k번째 고장까지 수행된 불완전한 수정의 수, 결함추가의 수를 나타내는 무작위 이산변수들을 표시한다(가정 5). 그러면 시각 t까지의 완벽한 수정의 수는 k-Ik-Jk이다. N(t)가 정수이어야 하므로 그리고 결함 수정 공정이 다항분포인 것(가정 5)으로 가정하므로 k개의 고장 시각 tk에 의한 N개 결함의 확률은 모든 ik, jk = 0, 1, …, k에 대해서

$$\Pr(N=n_0-k+i_k+2j_k) = \frac{k!}{i_k!j_k!(k-i_k-j_k)!}\, q^{i_k}r^{j_k}p^{k-i_k-j_k} \quad (6)$$

이다.

k고장이 방정식(2)에 따라 지수

$$\lambda_k=N(t)\lambda_0=(n_0-k+I_k+2J_k)\lambda_0 \quad (7)$$

를 가지는 지수분포(가정3)를 가지므로 그리고, Xk+1은 조건누적분포함수이므로 고장공정은

$$F(X_{k+1}|k|i_k,j_k)=1-\exp[(n_0-k+I_k+2J_k)\lambda_0\cdot x] \quad (8)$$

이다.

Xk+1 분포함수는 ik, jk에 대한 모든 확률이론을 적용하여 구할 수 있을 것이다.

$$F(X_{k+1}|k)=\sum_{i_k=0}^{k}\sum_{j_k=0}^{k}F(X_{k+1}|k|i_k,\ j_k) \cdot \Pr(N=n_0-k+i_k+2J_k) \quad (9)$$

방정식 (6)과 (8)을 (9)에 대입하여

$$R(x|k)\equiv[p+q\cdot\exp(-\lambda_0 x)+r\cdot\exp(-2\lambda_0 x)]^k \cdot \exp[-(n_0-k)\lambda_0 x] \quad (10)$$

를 유도한다. 하이브리드함수(10)은 MERF라 부르며, 이항지수함수를 개발한 것으로 생각할 수 있다.

그림 1은 파라미터 λ0=0.1, n0=40, p=0.8, q=0.15, r=0.05, k=0, 20, 30, 35, 40에 대해서 MERF k 곡선을 설명한 것이다. 이는 프로그램이 고장 나서 수정한 후에 TBF 누적분포함수가 다른 분포함수로 바뀌는 것을 보여주고 있다.

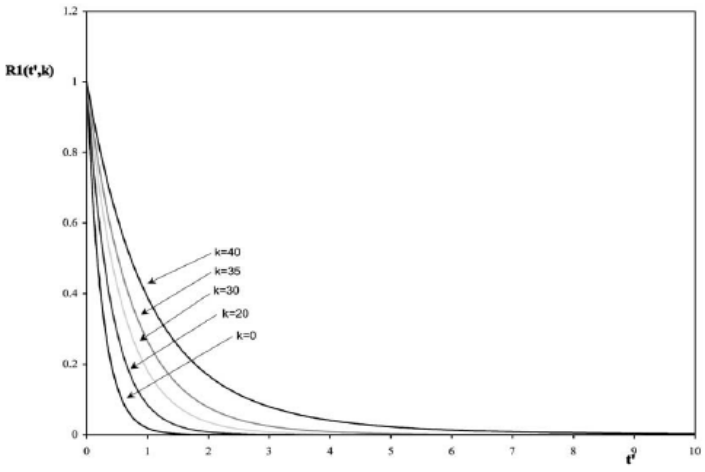

그림 1. MERF k 곡선

3.3 MERF 특성

MERF는 아래와 같이 표현되는 두 개의 중요한 특성을 가지고 있다.

정리 2.3.1 k고장으로 주어진 MERF가 감소확률을 나타낸다.

증명 : 고장간 시간간격 동안 MERF는 누적분포함수(10)을 가지므로 그 거동이 수정불가시스템과 융화할 수 있다.(요구되는 프로그램은 새로운 프로그램으로 간주될 수 있음). 그러면 주어진 k로 고장(위험)을 정의하는 것이 가능하다.

$$h(x|k)=-\frac{R'(x|k)}{R(x|k)} = \frac{k\lambda_0(qe^{-\lambda_0 t}+2re^{-2\lambda_0 t})}{p+qe^{-\lambda_0 t}+re^{-2\lambda_0 t}}+\lambda_0(n_0-k) \quad (11)$$

고장률은 k 고장 후 초기치를 가지는 단조감소 양의 함수이다.

$$h(0|k)=k\lambda_0(q+2r)+\lambda_0(n_0-k)=[n_0-(1-q-2r)k]\lambda_0 =[n_0-(p-r)k]\lambda_0 \quad (12)$$

이것이 점근선을 가로막지 않으면서 점근치 $\lambda_0(n_0-k)$로 감소한다. 따라서, 고장률 도함수는 k 고장 후 초기치 $\lambda_0^2[-(q+4r)+(q+2r)^2]$로 시작하는 단조 음의 함수이며, 시간축을 교차하지 않고 0까지 증가한다.

정리 2.3.2 MERF는 일정상수 x와 p≥r에 대해서 k값에 따라 증가한다.

증명 : 시각 x에서 (k+1)고장과 k고장 후의 신뢰도 차이는 다음과 같이 주어진다.

$$R1(x|k+1)-R2(xz|k)=[p(e^{\lambda_0 x}-1)+r(e^{-\lambda_0 x}-1]$$
$$\cdot[p+q\cdot e^{-\lambda_0 x}+r\cdot e^{-2\lambda_0 x}]\cdot e^{(n_0-k)\lambda_0 x} \quad (13)$$

두 번째 항과 세 번째 항이 언제나 양의 값이므로 어떠한 x에 있어서도 첫 번째 항이 양의값이라는 것을 증명하면 충분하다.

- p>r이면 어떠한 x에 대해서도 $p(e^{\lambda_0 x}-1)>r(1-e^{-\lambda_0 x})$이므로 첫 번째 항이 언제나 +이다.

-p<r이면 시간 간격 $[0,1/n_0\ \ln(p/r)]$동안 첫 번째 항이 -이다.

4. MERF 지수 점근

4.1 MERF점근함수

k고장 후 프로그램에 나타나는 결함의 수는 결함의 수로 표현되는 것으로 대체한다.

$$E(N)=E(n_0-k+I_k+2J_k)=n_0-k+(q+2r)k$$
$$=n_0-(p-r)k \quad (14)$$

우리는 k 고장과 k+1 고장 사이의 일정고장률을 얻는다.

$$\lambda_1=[n_0-k(p-r)]\lambda_0 \quad (15)$$

그리고 k 고장 후 프로그램 신뢰도는

$$R2(x|k)=\exp\{-[n_0-k(p-r)\lambda_0 x]\} \quad (16)$$

x≥0, λ0>0, n0≥k(p-r)

방정식(16)은 EARF로서 MERF에 근접하는 것으로 이해해도 된다. MERF를 테일러 시리즈로 전개하여 두 번째 항 이상의 항을 무시하여 그 근접치를 얻을 수도 있다. EARF 특성은 MERF 특성과 유사하다.

- k 고장과 k+1 고장 시간 간격기간 동안 EARF는 일정 고장률(15)을 나타낸다. 이는 MERF 초기 고장률(13)과 부합한다. 그래서 어떠한 x>0과 k에서도 $L1(x|k)<L2(x|k)$이다

- p>r이면 어떠한 고장에도 신뢰도 성장이 있다. 어떠한 k에 대해서

$$R2(x|k)=e^{(p-r)\lambda_0 x}R2(x|k-1) \quad (17)$$

그러므로, 공정의 고장/수정 파라미터가 알려져 있으면 EARF k 곡선 집합은 그림 1에서 보인 MERF k 곡선과 유사하게 그릴 수 있다. 그럼에도 불구하고 EARF k 곡선은 아래와 같이 정의되는 k 고장 후 등가시간 te를 채택하여 하나의 지수곡선으로 단순화할 수 있다.

$$t_e=\frac{[n_0-k(p-r)]r'}{n_0},\ t_{e>0} \quad (18)$$

결과적으로 방정식(16)과 등가인 하나의 파라미터 지수함수는 주어진 k에서 본래부터 내재하고 있는 오차 d는 체비세프(최대)노름으로 정의한다.

$$R2(x|k)=R2(t_e)=e^{-\lambda_0 n_0 t_e} \quad (19)$$

te≥0, λ0>0, n0≥0

4.2 오차 분석

그림 2는 MERF곡선, $R1(x|35)$ 및 이와 관련된 EARF 곡선, $R2(x|35)$를 보여주고 있다. λ0=0.1, n0=40, p=0.8, q=0.15, r=0.05이다. 그림 2의 우상단은 $R1(x|35)-R2(x|35)$를 보여주고 있다.

$$d=\|R1(x|k)-R2(x|k)\|_\infty \max|R1(x|k)-R2(x|k)|$$

그림 2의 오차 곡선은 시각 2.71이고 고장의 평균시간은 1.209에서 최대 절대값 오차 또는 모델 오차가 d=0.041임을 보여주고 있다.

모델오차 d는 각 파라미터 x에 대한 오차범위를 수립한다.

$$R2(x|k)-d\le R1(x|k)\le R2(x|k)+d \quad (20)$$

절대오차 d는 아래와 같은 표현으로 상대오차 e에 의해서 근사화할 수 있다.

$$d\fallingdotseq e\cdot R2(x|k) \quad (21)$$

여기서 R1(x|k) 대신 R2(x|k)를 써왔다.

체비세프 노름 가로좌표에서의 절대오차와 상대오차 e를 쉽게 산출하기 위해 각 k에 대한 컨투어 작도맵을 계산하였다.

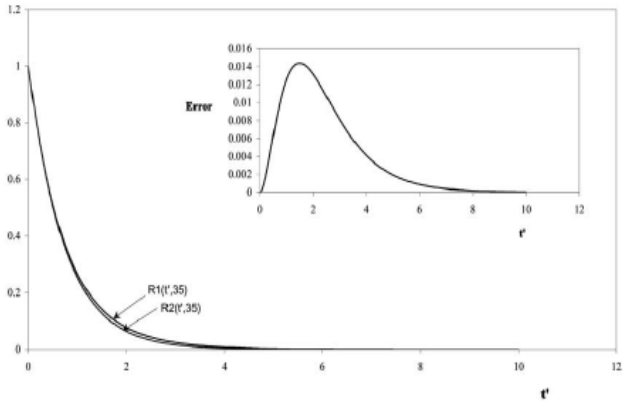

그림 2. MEFR-R1

4.3 파라미터 산출

EARF를 소프트웨어 신뢰도분석에 적용하려면 파라미터 λ0, n0, p, q, r을 산출해야 한다.

본 항에서는 두 공정이 독립적이라고 가정하여 MLE를 이용하여 수정 다항식 분포 파라미터 산출 $\hat{p}$, $\hat{q}$, $\hat{r}$을 산출하고 고장지수 근사신뢰도 함수 $\hat{\lambda}_0$, $\hat{n}_0$를 구한다.

m을 프로그램 고장의 총 테스트수라고 하자. $L_1(\theta_1)$, $L_2(\theta_2)$를 다항식이고 EARF분포 L함수라하며, Θ1과 Θ2를 이들의 파라미터벡터라고 하면

$$L_1(\theta_1) = p^{m-i_m-j_m} q^{i_m} r^{j_m}, \quad p+q+r=1 \tag{22}$$

이고, im과 jm은 관찰된 누적 불완전 및 오류도입이므로 EARF 확률분포함수로부터

$$L_2(\theta_2) = \prod_{k=1}^{m} f(x_k|\theta_2) = \lambda_0^{\,m} \prod_{k=1}^{m} [n_0 - k(p-r)]$$

$$\cdot \exp\left[-\lambda_0 \sum_{k=1}^{m} [n_0 - k(p-r)] x_k\right] \tag{23}$$

를 얻으며 여기서 x는 (k-1)번째 고장과 k번째 고장의 시간간격을 말한다.

고장과 결함추가공정이 독립적이라고 가정하였기 때문에 $L_1(\theta_1)$로부터 p, q, r을, $L_2(\theta_2)$로부터 λ0, n0을 구할 수 있다.

MLE는 $L_1(\theta_1)$을 최대로 하는 $\hat{p}$, $\hat{q}$, $\hat{r}$을 구하는 것이다.

$$\hat{p} = \frac{m-i_m-j_m}{m}, \quad \hat{q} = \frac{i_m}{m}, \quad r = \frac{j_m}{m} \tag{24}$$

그리고, 구한 $\hat{p}$, $\hat{q}$, $\hat{r}$ 을 이용하여 $\ln L_2(\theta_2)$을 최대로 하는 $\hat{\lambda}_0$, $\hat{n}_0$을 구하는 것이다.

$$\frac{m}{\hat{\lambda}_0} - \hat{n}_0 \sum_{k=1}^{m} x_k + (\hat{p}-\hat{r}) \sum_{k=1}^{m} k x_k = 0,$$

$$\sum_{k=1}^{m} \frac{1}{\hat{n}_0 - k(\hat{p}-\hat{r})} - \hat{\lambda}_0 \sum_{k=1}^{m} x_k = 0 \tag{25}$$

5. 예제

MLE 계산 및 적합성 테스트를 포함하는 소프트웨어신뢰도 계산의 EAR F적용수치 예를 표현한다. 소프트웨어 테스트공정은 몬테칼로 기법으로 모의하고 EARF에 적용했던 파라미터 λ0=0.1, n0=40, p=0.8, q=0.15, r=0.05을 이용한다. 모의에서는 39번의 시도(m=39)를 포함한다.

표1에서는 프로그램 고장 데이타를 보여주고 있다.

표 1 몬테칼로 모의 데이타

k	xk	ik	Jk	k	xk	ik	Jk
1	0.18	0	0	21	0.254	2	1
2	0.259	0	0	22	0.729	3	1
3	0.045	1	0	23	0.094	3	1
4	0.11	1	0	24	0.02	3	1
5	1.46	1	0	25	0.137	3	1
6	1.385	1	0	26	0.372	3	1
7	0.457	1	0	27	1.051	3	1
8	0.12	1	0	28	0.143	3	1
9	0.007	1	0	29	0.043	4	1
10	0.147	1	0	30	1.097	5	1
11	0.278	1	0	31	0.333	5	1
12	0.235	1	0	32	0.688	5	1
13	0.516	1	0	33	0.066	5	1
14	1.507	1	0	34	0.323	5	1
15	0.043	1	1	35	0.003	5	1
16	0.165	1	1	36	0.783	5	1
17	0.328	2	1	37	1.055	5	2
18	0.194	2	1	38	0.876	5	2
19	0.338	2	1	39	3.363	5	2
20	0.497	2	1				

5.1 파라미터산출

m=39에 대해서 방정식(19)에 im, jm을 넣는다. 그래서 수정공정파라미터

$\hat{p}=0.821$, $\hat{q}=0.128$, $\hat{r}=0.051$s을 얻는다. 그리고 $\hat{p}$, $\hat{q}$, $\hat{r}$, $\sum x_k$, $\sum k x_k$(표1)을 방정식(25)에 도입하여 다음과 같은 동시비선형방정식을 얻는다.

$$\frac{39}{\hat{\lambda}_0} - 19.701\ \hat{n}_0 + 355.2 = 0,$$

$$\sum_{k=1}^{39} \frac{1}{\hat{n}_0 - 0.77k} - 19.701\ \hat{\lambda}_0 = 0 \tag{26}$$

동시 비선형 방정식을 풀어서 고장 공정 파라미터

$\hat{\lambda}_0 = 0.076$, $\hat{n}_0 = 43$, $\hat{\lambda}\ \hat{n}_0 = 3.268$을 얻는다.

39개의 프로그램 고장(13) 후 EARF는

$$\lambda_{39} = [\ \hat{n}_0 - k(\hat{p} - \hat{r})]\lambda_0 = 0.986$$

이고소프트웨어신뢰도는

$$R_2(x|39) = e^{-0.986x}$$

이고 평균 TBF는

$$MTBF = \frac{1}{0.986} = 1.014$$

이다.

참고문헌

[1] C. V. Ramamoorthy, F. B. Bastani, "Software reliability - Status and perspectives", IEEE Trans. on Software Eng., vol. SE-8, pp354-371, 1982 Aug.

[2] S. Yamada, H. Ohtera, H. Narihisa, "Software reliability growth models with testing- efforts", IEEE Trans. Reliability, vol. R-35, pp19-23, 1986 Apr.

[3] Michael A. Friedman, Jeffrey M. Voas, " Software Assessment : Reliability, Safety, Testability", John Wiley & Sons, Inc., pp199-204, 1995

[4] Syed A. Hossain, Ram C. Dahiya, " Estimating the Parameters of a Non-homogeneous Poisson-Process Model for Software Reliability", IEEE Trans. Reliability, vol. 42, no.4, pp604-612, 1993 Dec.

[5] Peter Spreij, " Parameter Estimation for a Specific Software Reliability Model", IEEE Trans. on Reliability, vol. R-34, no. 4, pp323-332, 1985.Oct

[6] Tapan Kumar Nayak, " Software Reliability:Statiscal Modeling & Estimation", IEEE Trans. on Reliability, vol. R-35, no.5, pp566-570, 1986 Dec.

[7] S. Yamada, J. Hishitani, S. Osaki, "Software - Reliability Growth with a Weibull Test-Effort : A Model & Application", IEEE Trans. Reliability, vol. 42, no.1, pp100-106, 1993 March

[8] S. Yamada, S. Osaki, "Cost-reliability optimal release policies for software systems", IEEE Trans. on Reliability, vol. R-34, 1985 Dec., pp422-424

[9] Amrit L. Goel, Kazu Okumoto, "Time - Dependent Error - Detection Rate Model for Software Reliability and Other Performance Measure", IEEE Trans. on Reliability, vol R-28, No.3, 1979.8. pp206-211

[10] Chin-Yu Huang, Sy-Yen Kuo, "Analysis of Incorporating Logistic Testing-Effort Function into Software Reliability Modeling", IEEE Trans. on Reliability, vol.51, pp261-270, 2002, Sep.

[11] Xuemei Zhang, Hoang Pham, "An analysis of factors affecting software reliability", The Journal of Systems and Software, 2000. pp43-56