▩ 연구논문

다중오류들을 갖는 소프트웨어 신뢰성의 추정†

이인석 · 정원태

경북대학교 통계학과

정혜정

피어선대학교 전산통계학과

Estimation of Software Reliability with Multiple Errors

In-Suk Lee · Won-Tae Jung

Dept. of Statistics, Kyungpook National University

Hye-Jeong Jeong

Dept. of Computer Science & Statistics, Pierson University

Abstract

In this paper, we consider possibility that the multiple errors occur in each testing stage. At present, software reliability modelling is considered as a part of software reliability quality assurance in software engineering. However they dealt with the software growth model for the single error debugging at each testing stage until now. Hence it is necessary to study software reliability with multiple errors debugging.

Therefore we propose software reliability growth modeling and estimate the parameters in the proposed software reliability growth model for the multiple errors debugging at each testing stage.

† 이 논문은 1994년도 한국학술진흥재단의 공모과제 연구비에 의하여 연구 되었음.

1. 서론

소프트웨어 신뢰성은 소프트웨어공학과 신뢰성 성장모형 사이의 연결분야로 많은 학자들 사이에 연구가 되어왔다. 즉, Jelinski와 Moranda(1972), Littlewood와 Verrall (1973, 1974), Moranda(1975), Musa(1979)와 Shooman(1991) 등은 다양한 소프트웨어 신뢰성 성장모형(Software Reliability Growth Model, 이하 SRGM)을 제안 하였다.

소프트웨어 순환 사이클은 분석, 설계, 코딩 그리고 시험 및 유지로 구성되어 있으며, 소프트웨어 개발의 순환 사이클중 시험 및 유지 보수면에 보통 40%의 노력을 투자하고, 그 규모가 커지거나 복잡도에 따라 더 많은 노력이 요구된다.

Jelinski-Moranda 모형은 지금까지 소프트웨어 신뢰성 성장모형중 가장 좋은 것으로 평가 받고 있다. 그리고 역시 소프트웨어 신뢰성 공학자들은 Jelinski-Moranda 모형을 자주 수정하여 사용하였다. 그러나 이 SRGM 그룹은 단지 각 시험 단계에서 1개의 오류 디버깅인 경우에 적용하였다. 따라서 이 모형은 각 시험단계에서 다중오류가 발생하는 경우에는 사용할 수 없다. 그래서 우리는 다중 소프트웨어 오류가 발생하는 경우 다중 소프트웨어 오류들의 실패간시간들(interfailure time) 사이의 종속성을 고려하여 Gamma-Lomax SRGM 모형을 제안하였고, 최우추정법으로 추정된 모수를 사용하여 3개의 소프트웨어 신뢰성 측도를 계산하려는 것으로,

2절에는 다중 오류들의 실패간시간들 사이의 통계적으로 종속인 성질을 고려한 Gamma-Lomax SRGM을 제안하였고,

3절에서 다중 소프트웨어 실패율의 평균시간과 현재의 소프트웨어 신뢰도를 유도하였으며,

마지막 4절에서는 제안된 모형의 모수 추정을 위해 최우추정법을 구하여 보았다.

2. Gamma-Lomax 소프트웨어 신뢰도 성장모형

기존의 Jelinski-Moranda 모형은 다중 소프트웨어 오류가 일어나는 단계의 시험에는 사용이 부적절 하다. 따라서 다중 소프트웨어 오류가 발생하는 경우에 새로운 모형을 제안하기 위하여 다음과 같이 가정한다.

(1) i번째 시험 단계에서 n_i개의 다중 소프트웨어 오류들이 일어나고, 각 시험단계는 $i=1, 2, \cdots, m$ 이고 마지막 시험 단계 m은 고정된 미지의 상수로써 소프트웨어에 있는 최초의 총 오류수는 $N=\sum_{i=1}^{m} n_i$으로 가정한다.

(2) 각 시험 단계에서 발견된 모든 소프트웨어 오류들은 다음 시험 단계가 시작되기 전에 반드시 수정이 이루어진다고 가정한다.

(3) 다중 소프트웨어 실패간시간들, $T_i=X_{(i)}-X_{(i-1)}$, $i=1, 2, \ldots, m$은 독립이면서 각 단계에서 발생하는 오류수와 실패강도율 n_i와 $\lambda_i=(m-i+1)\phi$의 모수가 감마분포를 따른다고 가정한다.

여기서, $0=X_{(0)}\leq X_{(1)}\cdots\leq X_{(m)}$은 순서화된 실패시간들이고 소프트웨어 강도 ϕ는 미지이고 각단계의 실패수는 기지의 값으로 가정한다.

소프트웨어 시험환경하에서 위의 조건 (1) (2) (3)을 만족한다는 가정을 하고 소프트웨어 실패간시간들 $T_1, T_2, \ldots, T_m$은 독립이며 각각의 다중 소프트웨어 실패율 $\eta\lambda_1, \eta\lambda_2, \ldots \eta\lambda_m$을 갖는 감마분포를 따른다고 가정하고, 시험환경은 공통모수 η에 의해 모든 시험단계의 실패율이 변한다고 가정하고 여기에서의 η를 a와 b를 모수로 가지는 감마확률변수라 한다면, 비조건부 결합 확률밀도 함수를 아래와 같이 유도할 수 있다.

$$
\begin{aligned}
&f(t_1, t_2, \ldots, t_m)\\
&= GLm(a, b; \lambda_1, \ldots, \lambda_m; n_1, \ldots, n_m)\\
&= \int_0^\infty \left(\prod_{i=1}^{m} \frac{(\eta\lambda_i)^{n_i} t_i^{n_i-1}\exp(-\eta\lambda_i t_i)}{\Gamma(n_i)}\right) \times \frac{b^a\eta^{a-1}\exp(-\eta b)}{\Gamma(a)}\, d\eta\\
&= \left(\prod_{i=1}^{m} \frac{t_i^{n_i-1}\lambda_i^{n_i}}{\Gamma(n_i)}\right) \frac{b^a\Gamma(\sum_{i=1}^{m} n_i+a)}{\Gamma(a)} \left(\frac{1}{\sum_{i=1}^{m}\lambda_i t_i+b}\right)^{\sum_{i=1}^{m} n_i+a}
\end{aligned} \tag{2.1}
$$

단, $\lambda_i>0$, $t_i>0$, $a>0$ 그리고 $b>0$ 이다.

우리는 Nayak 연구에서 각 단계마다 하나의 오류가 발생하는 경우에 대한 소프트웨어 신뢰도 성장모형을 Lomax 분포로 칭한것과 구별하기 위하여 우리가 유도한 (2.1)식을 Gamma-Lomax 소프트웨어 신뢰도 성장모형(SRGM)이라 정의하고 이것을 $GLm(a, b; \lambda_1, \ldots, \lambda_m; n_1, \ldots, n_m)$와 같이 표기 하기로 한다.

정리 2.1 (Nayak (1987)) $r(\leq m)$번째 시험단계까지의 다중 소프트웨어 실패간시간들, $T_1, T_2, \ldots, T_r$, 의 주변 확률밀도함수는
$GLr(a, b; \lambda_1, \lambda_2, \ldots, \lambda_r; n_1, n_2, \ldots, n_r)$ 이다. 단, λ_i는 i번째 실험단계의 실패율이다.

Nayak(1986)은 1개의 소프트웨어 오류 수정의 신뢰도 성장모형을 제안 하였는데, 이때 실패시간들, $X_1, X_2, \ldots, X_r$, $r\leq m$의 주변 확률밀도함수는
$GLr(a, 1; \lambda_1, \ldots, \lambda_r; n_1, n_2, \ldots, n_r)$ 이라고 나타낼 수 있다.

정리 2.2 (Nayak (1987)) $T_1=t_1, T_2=t_2, \ldots, T_r=t_r$, $r\leq m$이 주어질 때 다중오류 실패간시간들, $T_{r+1}, T_{r+2}, \ldots, T_m$의 조건부 확률밀도함수는 다음과 같이 표현될 수 있다.

$$
\begin{aligned}
&GL_{(m-r)}(A_r, 1; \lambda^*_{r+1}, \lambda^*_{r+2}, \ldots, \lambda^*_m; n_{r+1}, n_{r+2}, \ldots, n_m)\\
&= f(t_{r+1}, t_{r+2}, \ldots, t_m \mid t_1, t_2, \ldots, t_r)\\
&= \left(\prod_{i=r+1}^{m} t_i^{n_i-1} \frac{\lambda_i^{*n_i}}{\Gamma(n_i)}\right)\left(\frac{1}{\sum_{i=r+1}^{m}\lambda_i^* t_i+1}\right)^{\sum_{i=1}^{m} n_i+a} \times \frac{G_m}{G_r}
\end{aligned}
$$

단, $\lambda^*_i = \dfrac{\lambda_i}{\sum_{j=1}^{r} \lambda_j t_j + b}$; 여기서 $i=r+1,\ r+2, \ldots, m$ 이고

$$G_r = \Gamma(A_r),$$
$$A_r = \sum_{i=1}^{r} n_i + a \text{ 이다.} \qquad (2.2)$$

또한, $t_i > 0$, $\lambda^*_i > 0$, $a > 0$, $b > 0$, 이고 λ_i는 i번째 시험단계의 실패율이다.

Nayak(1986)은 만약 각 실험 단계마다 소프트웨어 오류가 하나씩 발생한다면, $X_1 = x_1$, $X_2 = x_2, \ldots, X_r = x_r$이 주어질때 $X_{r+1}, \ldots, X_m$의 조건부 확률밀도함수는 아래와 같이 나타낼 수 있다.

$$GL_{(m-r)}(a^*, 1\,;\ \lambda^*_{r+1}, \lambda^*_{r+2}, \ldots, \lambda^*_m\,;\ 1, 1, \ldots, 1),$$

단, $\lambda^*_i = \dfrac{\lambda_i}{\sum_{i=1}^{r} \lambda_i x_i + 1}$ 이고,

$a^* = a + r$ 이다.

위의 정리 2.1과 정리 2.2을 이용하여 간단히 계산하면 정리 2.3과 정리 2.4를 얻을 수 있다.

정리 2.3 소프트웨어 실패간시간들, $T_1, T_2, \ldots, T_m$는 실패강도율이 단계마다 감소해 간다.

증명) 먼저 각 단계의 실패율을 구하기 위해서 실패시간들 사이에 종속성을 고려한 확률 밀도함수를 계산해 보면

$$f(t) = \int_0^{\infty} \frac{(\eta\lambda)^n \exp(-\eta\lambda t) t^{n-1}}{\Gamma(n)} \frac{b^a \eta^{a-1} \exp(-b\eta)}{\Gamma(a)}$$
$$= \frac{t^{n-1}\lambda^n b^a}{\Gamma(n)} \left(\frac{1}{b+\lambda t}\right)^{n+a} \frac{\Gamma(n+a)}{\Gamma(a)},$$

위의 확률밀도함수를 이용하여 실패시간의 분포함수를 구하여 보면

$$F(t) = Pr(T \le t)$$
$$= \int_0^t \frac{x^{n-1}\lambda^n b^a}{\Gamma(n)} \left(\frac{1}{b+\lambda x}\right)^{n+a} \frac{\Gamma(n+a)}{\Gamma(a)} dx$$
$$= B_{\frac{\lambda t}{\lambda t + b}}(a, n),\ 0 < t < \infty,$$

여기에서 $B_{\frac{\lambda t}{\lambda t+b}}(a,\ n)$은 불완전 베타분포이다. 따라서 시간 t의 실패율 $\lambda(t)=f(t)/(1-F(t))$는 시간에 따라서 감소한다. 베타분포의 특성에 따라 일변량의 경우처럼 다변량의 경우에도 위의 정리 내용이 성립함을 쉽게 알수 있다.

정리 2.4　다중 소프트웨어 실패간시간들, $T_1, \ldots, T_m$에 대해, $E(T_1^{r_1}, \ldots, T_m^{r_m})$는 다음과 같이 표현할 수 있다.

$$E(T_1^{r_1}, \ldots, T_m^{r_m})$$
$$= \begin{cases} \left(\prod_{i=1}^{m} \dfrac{\Gamma(n_i+r_i)}{\Gamma(n_i)\lambda_i^{r_i}}\right) \dfrac{\Gamma(a-\prod_{i=1}^{n_i} r_i)}{\Gamma(a)} \times b^{\sum_{i=1}^{m} r_i}, & \sum_{i=1}^{m} r_i < a \text{ 인 경우} \\ \infty & \text{그외의 경우.} \end{cases}$$

증명) 베타 분포의 모수 $a>2$ 경우,

$$E(T_i) = n_i b/\lambda_i(a-1),$$
$$Var(T_i) = n_i(n_i+a-1)b^2/\lambda_i^2(a-1)^2(a-2),$$
$$Cov(T_i, T_j) = n_i n_j b^2/\lambda_i\lambda_j(a-1)^2(a-2),$$

그리고

$$Cov(T_i, T_j) = \sqrt{(n_i n_j)}\,/\sqrt{(n_i+a-1)(n_j+a-1)}$$

와 같은 결과를 얻을 수 있고 이것을 이용하여 위의 정리를 얻을 수 있다.

3. 소프트웨어 신뢰도

먼저 시간 $t=\tau$, $x_{(x)} < \tau < x_{(x+1)}$ 에서 $\sum_{i=1}^{r} n_i$개의 오류들이 수정된 후 소프트웨어 신뢰도에 대하여 생각하고자 한다. 그러면 정리 2.1을 이용하여 $T_1=t_1$, $T_2=t_2$, …, $T_r=t_r$, $T_{r+1}>\bar{t}$이 주어질때 $r+1$번째 다중 소프트웨어 실패간시간 T_{r+1}의 조건부 확률밀도함수는

$$f(t_{r+1} \mid t_1, \ldots, t_r, T_{r+1} > \bar{t})$$
$$= \frac{((m-r)\phi)^{n_{r+1}}}{\Gamma(n_{r+1})} \frac{G_{r+1}}{G_r} \frac{Q_r}{1-B_{L_r(\bar{t})}(n_{r+1}, A_r)} \qquad (3.1)$$

단, $A_r = \sum_{i=1}^{r} n_i + a$

$$M_r = \sum_{i=1}^{r} (m-i+1)\phi t_i + b$$

$$Q_r = M_r^{A_r}$$

$$G_r = \Gamma(A_r)$$

$$L_r(\bar{t}) = \frac{(m-r)\phi\bar{t}}{(M_r+(m-r)\phi\bar{t})},$$

와 같이 얻을 수 있으면, 이 결과들을 이용하면 순서화된 실패간들 $x_{(1)}, x_{(2)}, \ldots, x_{(r)}$과 $X_{(r+1)} > \tau$이 주어질 때 T_{r+1}의 조건부 확률밀도함수 $g(t_{r+1} \mid x_{(1)}, \ldots, x_{(r)}, X_{(r+1)} > \tau)$는

$$g(t_{r+1} \mid x_{(1)}, \ldots, x_{(r)}, X_{(r+1)} > \tau) = \frac{((m-r)\phi)^{n_{r+1}}}{\Gamma(n_{r+1})} \frac{G_{r+1}}{G_r} \frac{H_r(x_{(r)})}{1-B_{U_r(\tau)}(n_{r+1}, A_r)} \times \frac{t_{r+1}^{n_{r+1}-1}}{(Pr(x_{(r)})+(m-r)\phi t_{r+1})^{A_{r+1}}} \quad (3.2)$$

와 같이 얻을 수 있다.

단, $Pr(\tau) = (\sum_{i=1}^{r} x_{(i)} + (m-r)\tau)\phi + b$

$$H_r(\tau) = Pr(\tau)^{A_r}$$

$U_r(\tau) = \dfrac{(m-r)\phi(\tau - x_{(r)})}{Pr(\tau)}$, 이며 A_r와 G_r는 (3.1)식에서 주어진 값이다.

따라서 이들의 결과를 이용하면 다음의 (3.3)식을 쉽게 구할 수 있다.

정리 3.1 $\sum_{i=1}^{r} n_i$개의 오류들이 수정된 후 시간 $\tau \in (x_{(r)}, (x_{(r+1)})$에서 소프트웨어의 평균 실패시간은 다음과 같다.

$$MTTF(t_{r+1}) = E(T_{r+1} \mid x_{(1)}, x_{(2)}, \ldots, x_{(r)}, X_{(r+1)} > \tau) = \frac{Pr(x_{(r)})n_{r+1}}{(m-r)\phi(A_r-1)(1-B_{u_r(\tau)}(n_{r+1}, A_r))} \quad (3.3)$$

여기서 $Pr(x_{(r)})$과 $u_r(\tau)$은 (3.2)식에서, A_r은 (3.1)에서 주어진 값이다.

증명) 소프트웨어에서 다중 오류가 발생한 경우에 다중 오류의 평균 실패시간은 위의 순서화된 실패시간들 사이에 조건부확률을 이용하면 쉽게 유도할 수 있다.

$$
\begin{aligned}
&E(T_{r+1} \mid x_{(1)}, x_{(2)}, \ldots, x_{(r)}, X_{(r+1)} > \tau) \\
&= \int_0^{\infty} t_{r+1} \times g(t_{r+1} \mid x_{(1)}, x_{(2)}, \ldots, x_{(r)}, X_{(r+1)} > \tau)\, dt_{r+1} \\
&= \frac{\{(m-r)\phi\}^{n_{r+1}} G_{r+1} H_r(x_{(r)})}{\Gamma(n_{r+1}) G_r \{1-B_{U_r}(\tau)(n_{r+1}, A_r)\}} \\
&\times \int_0^{\infty} t_{r+1}^{n_{r+1}} / (P_r(x_{(r)}) + \{(m-r)\phi\}^{A_{r+1}} dt_{r+1} \\
&= \frac{P_r(x_{(r)}) n_{r+1}}{(m-r)\phi(A_r-1)\{1-B_{U_r}(\tau)(n_{r+1}, A_r)\}}
\end{aligned}
$$

이번에는 각 시험단계에서 다중 오류가 일어나는 경우 $x_{(r)}$과 $\tau = x_{(r)} + \bar{t} < x_{(r+1)}$ 사이의 시간 $t_0(\le \bar{t})$에 대한 현재의 소프트웨어 신뢰도를 구하기 위하여 (3.2)식을 사용하면 정리 3.2와 정리 3.3의 결과를 계산할 수 있다.

정리 3.2 만약 각 시험단계에서 다중 소프트웨어 오류를 가정하면 $\sum_{i=1}^{r} n_i$ 개의 오류들이 수정된 후 시간 $t_0(\le \bar{t})$ 동안 소프트웨어 체계의 신뢰도는 다음과 같다.

$$
R(t_0) = \frac{1-B_{V_r(t_0)}(n_{r+1}, A_r)}{1-B_{u_r(\tau)}(n_{r+1}, A_r)}
$$

단, A_r, G_r, $H_r(x_{(r)})$, $Pr(x_{(r)})$, $U_r(\tau)$는 (3.1)식과 (3.2)식에 주어진 값이고,

$$
V_r(t_0) = \frac{(m-r)\phi t}{(Pr(x_{(r)}) + (m-r)\phi t_0)} \text{이다.} \tag{3.4}
$$

증명) $x_{(r)}$과 $x_{(r+1)}$ 사이에 실패가 하나도 발생하지 않을 확률을 구하여 특정 시간 동안의 신뢰도로 정의 한다면 다음과 같이 유도할 수 있다.

$$
\begin{aligned}
R(t_0) &= Pr(T_{r+1} > t_0 \mid x_{(1)}, x_{(2)}, \ldots, x_{(r)}, X_{(r+1)} > \tau) \\
&= \int_{t_0}^{\infty} g(t_{r+1} \mid x_{(1)}, x_{(2)}, \ldots, x_{(r)}, X_{(r+1)} > \tau)\, dt_{r+1} \\
&= \frac{\{(m-r)\phi\}^{n_{r+1}} G_{r+1} H_r(x_{(r)})}{\Gamma(n_{r+1}) G_r \{1-B_{U_r}(\tau)(n_{r+1}, A_r)\}} \\
&\times \int_{t_0}^{\infty} t_{r+1}^{n_{r+1}} / (P_r(x_{(r)}) + \{(m-r)\phi\}^{A_{r+1}} dt_{r+1} \\
&= \frac{1-B_{V_r}(t_0)(n_{r+1}, A_r)}{1-B_{U_r}(\tau)(n_{r+1}, A_r)} \text{.}
\end{aligned}
$$

정리 3.3 $x_{(1)}, x_{(2)}, \ldots, x_{(r)}, x_{(r+1)} > \tau$이 주어질때 $x_{(r)} + t_0 < x_{(r+1)}$에서 다중 오류가 발생한 경우의 소프트웨어 실패강도율은

$$\lambda(t_0) = \frac{((m-r)\phi)^{n_{r+1}}}{\Gamma(n_{r+1})} \frac{G_{r+1}}{G_r} \frac{H_r(x_{(r)})\, t_0^{n_{r+1}-1}}{(Pr(x_{(r)}) + (m-r)\phi t_0)^{A_{r+1}}} \times \frac{1}{(1 - B_{v_r(t_0)}(n_{r+1}, A_r))}$$

이다. 단, $A_r, G_r, H_r(x_{(r)}), Pr(x_r)$과 $V_r(t_0)$는 식 (3.1), (3.2)와 (3.4)에서 주어진 값이다.

증명) 위의 정리 3.2에서 유도한 특정 시간 동안의 신뢰도와 3절에서 소개한 조건부 확률을 이용한다면 다중 오류가 발생한 경우의 소프트웨어의 실패강도율은 쉽게 구할 수 있다.

4. 모수 추정

이제 소프트웨어 신뢰도 모형에서 미지의 모수들 m, ϕ와 n_{r+1}의 추정을 위하여 최우추정 절차를 생각해 보자. 여기서 우선 $\tilde{t} = \{t_1, t_2, \ldots, t_r\}$를 다중 소프트웨어 실패들간의 시간을 조사한 실험자료라 가정하고, 이 시간을 크기순으로 r번째까지 관측되었다고 가정하자.

순서를 고려한 다중 소프트웨어 실패시간은 $x_{(1)}, x_{(2)}, \ldots, x_{(r)}$, 이고 τ를 $\tau \in (x_{(r)}, x_{(r+1)})$으로 간주하고, 또한 $i = 1, \cdots, r$에 대해 $t_i = x_{(i)} - x_{(i-1)}$이며 $T_{(r+1)} > \bar{t} = \tau - x_{(r)}$이라 하자. 그럼 지금부터 각 단계마다 다중 오류가 발생한다는 가정하에 제한한 소프트웨어 신뢰도 성장모형에 대한 모수 추정 결과를 살펴보자.

[ML 1] $\tilde{t}$가 주어질때 m과 ϕ의 우도함수는

$$L_1(m, \phi \mid \tilde{t}) = \left(\prod_{i=1}^{r} \frac{((m-i+1)\phi)^{n_i} t_i^{n_i - 1}}{\Gamma(n_i)}\right) \frac{b^a \Gamma(\sum_{i=1}^{r} n_i + a)}{\Gamma(a)} \times \left(\frac{1}{\sum_{i=1}^{r} (m-i+1)\phi t_i + b}\right)^{\sum_{i=1}^{r} n_i + a} \tag{4.1}$$

이다.

[ML 2] $\tilde{t}$가 주어질때 $\bar{t}$의 조건부 확률밀도함수는

$$f(\bar{t}\mid\tilde{t}) = \frac{((m-r)\phi)^{n_{r+1}}}{\Gamma(n_{r+1})}\ \frac{\Gamma(\sum_{i=1}^{r+1} n_i+a)}{\Gamma(\sum_{i=1}^{r} n_i+a)}\times$$

$$\frac{\bar{t}^{n_{r+1}-1}(\sum_{i=1}^{r}(m-i+1)\phi t_i+b)^{\sum_{i=1}^{r} n_i+a}}{(\sum_{i=1}^{r}(m-i+1)\phi t_i+b+(m-r)\phi\bar{t})^{\sum_{i=1}^{r+1} n_i+a}} \text{ 이다.} \tag{4.2}$$

[ML 3] (4.1)식과 (4.2)식을 연결하여 $\tilde{t}$와 $\bar{t}$가 주어질때 m, ϕ과 n_{r+1}의 우도함수를 얻을 수 있다.

$$L_2(m,\phi,n_{r+1}\mid\tilde{t},\bar{t})$$
$$=(\prod_{i=1}^{r}\frac{((m-i+1)\phi)^{n_i}t_i^{n_i-1}}{\Gamma(n_i)})\ \frac{b^a}{\Gamma(a)}\ \frac{((m-r)\phi)^{n_{r+1}}}{\Gamma(n_{r+1})}\times$$
$$\frac{\bar{t}^{n_{r+1}-1}\Gamma(\sum_{i=1}^{r+1} n_i+a)}{(\sum_{i=1}^{r}(m-i+1)\phi t_i+b+(m-r)\phi\bar{t})^{\sum_{i=1}^{r+1} n_i+a}} \text{ 이다.} \tag{4.3}$$

그러므로 다음의 (4.4)식을 얻을 수 있고, 이것을 m, ϕ와 n_{r+1}에 대하여 편미분한 것이 0이 되도록 하면 (4.5)식을 얻을 수 있다.

$$\ln L_2(m,\phi,n_{r+1}\mid\tilde{t},\bar{t})$$
$$\propto \sum_{i=1}^{r+1} n_i\ln(m-i+1)+\sum_{i=1}^{r+1} n_i\ln\phi-\ln\Gamma(n_{r+1})$$
$$+(n_{r+1}-1)\ln\bar{t}+\ln\Gamma(\sum_{i=1}^{r+1} n_i+a)$$
$$-(\sum_{i=1}^{r+1} n_i+a)\ln(\sum_{i=1}^{r}(m-i+1)\phi t_i+b+(m-r)\phi\bar{t}) \tag{4.4}$$

$$\frac{\partial\ln L_2}{\partial m} = \sum_{i=1}^{r+1}\frac{n_i}{m-i+1}-(\sum_{i=1}^{r+1} n_i+a)\ \frac{\phi(\sum_{i=1}^{r} t_i+\bar{t})}{(\sum_{i=1}^{r}(m-i+1)\phi t_i+b+(m-r)\phi\bar{t})} = 0,$$

$$\frac{\partial\ln L_2}{\partial\phi} = \frac{\sum_{i=1}^{r+1} n_i}{\phi}-(\sum_{i=1}^{r+1} n_i+a)\ \frac{(\sum_{i=1}^{r}(m-i+1)t_i+(m-r)\bar{t})}{(\sum_{i=1}^{r}(m-i+1)\phi t_i+b+(m-r)\phi\bar{t})} = 0,$$

$$\frac{\partial \ln L_2}{\partial n_{r+1}} = \ln \bar{t} - \frac{\Gamma'(n_i + a)}{\Gamma(n_{r+1})} + \ln(m-r) + \ln\phi + \frac{\Gamma'(\sum_{i=1}^{r+1} n_i + a)}{\Gamma(\sum_{i=1}^{r+1} n_i + a)}$$

$$- \ln\left(\sum_{i=1}^{r} (m-i+1)\phi t_i + b + (m-r)\phi\bar{t}\right) = 0. \tag{4.5}$$

[ML 4] 위 (4.5)식을 동시에 풀면 아래와 같은 추정식을 얻을 수 있다.

$$\sum_{i=1}^{r+1} \frac{n_i}{m-i+1} = \frac{\sum_{i=1}^{r+1} n_i (\sum_{i=1}^{r} t_i + \bar{t})}{\sum_{i=1}^{r} (m-i+1) t_i + (m-r)\bar{t}}, \tag{4.6}$$

$$\phi = \frac{b}{a} \cdot \frac{\sum_{i=1}^{r+1} n_i}{\sum_{i=1}^{r} (m-i+1) t_i + (m-r)\bar{t}}, \tag{4.7}$$

$$\frac{(\sum_{i=1}^{r+1} n_i + a)(\sum_{i=1}^{r} (m-i+1) t_i + (m-r)\bar{t})}{\bar{t}(m-r)\sum_{i=1}^{r+1} n_i} = \exp\left[\frac{\Gamma'(\sum_{i=1}^{r+1} n_i + a)}{\Gamma(\sum_{i=1}^{r+1} n_i + a)}\right]$$, 이 되고

여기서 $\frac{\Gamma'(x)}{\Gamma(x)} = -\gamma + (1 - \frac{1}{x}) + (\frac{1}{2} - \frac{1}{x+1}) + \cdots + (\frac{1}{n} - \frac{1}{x+n-1}) + \cdots$

$-\gamma = \Gamma'(1) = \int_0^\infty \exp(-x) \ln x \, dx$ 이다.

최우추정의 절차 ML 1에서 ML 4에 따라 모수 m, ϕ와 n_{r+1}의 근사 추정치 $\hat{m}$, $\hat{\phi}$과 $\hat{n}_{r+1}$를 구할 수 있다. 따라서 (3.3), (3.4)와 (3.5)식에 위에서 추정한 추정치를 대입하여 소프트웨어 신뢰도 측도들을 구할 수 있다.

5. 결론 및 제언

각 실패시간들 사이에 오류가 연속으로 발생하여 실패들 사이의 시간이 0이 되는 경우를 우리는 MUSA DATA를 보면 알 수 있다. 이 경우는 기존에 제안된 단일 오류가 발생된 경우의 소프트웨어 신뢰도 성장 모형에 적용하여 신뢰도를 측정해 보면 엉뚱한 경우가 발생할 수 있는데 이러한 경우에 다중 오류 발생으로 취급하여 우리가 제안한 신뢰성 성장모형에 적용해 보면 더욱 더 타당성 있는 결과를 얻을 수 있다고 생각되며 신뢰도를 가지고 비교해 본 모의 실험 결과 〈표 1〉에서도 역시 우리가 제안한 다중 오류 수정의 경우가 더욱 더 현실적임을 알 수 있었다.

〈 표 1 〉 신뢰도 모의 실험 결과

T	JM(S)	JM(M)	T	JM(S)	JM(M)
171223(2)	0.4400268	0.5767819	222789(2)	0.9999905	0.9853616
0	1		0	1	
302345(3)	0.9227171	0.8164130	0	1	
0	1		389012(1)	0.9999999	0.9964133
0	1		153462(1)	1	0.9995296
222579(2)	0.9994618	0.9420706	197293(1)	1	0.9999999
0	1				

여기에서 T는 실패시간들 사이의 시간을 나타내며 JM(S)는 단일 오류가 발생한 경우에 대해 제안한 Jelinski-Moranda의 소프트웨어 신뢰도 성장모형을 실패시간들 사이에 종속성을 고려하여 제안한 Nayak의 소프트웨어 신뢰도 성장모형에 대입하여 신뢰도를 측정한 것이고 JM(M)은 다중 오류 신뢰도 성장 모형에서 우리가 제안한 소프트웨어 신뢰도 성장모형에다가 신뢰도를 계산한 값인데 실패가 연속적으로 발생하여 실패시간들 사이에 시간이 0인 경우는 기존의 단일 오류 신뢰도 성장모형에 적용하여 신뢰도를 계산해 보면 곧 아직까지 프로그램에는 오류가 남아 있는데도 불구하고 신뢰도는 1이 되어 버리는 엉뚱한 결과를 볼 수 있으므로 우리가 제안한 다중 오류 신뢰도 성장 모형이 이와 같은 상황에서 크게 현실성이 있음을 알 수 있다.

그리고 모수 추정에 있어서는 기존의 모형 즉, 각 단계에서 1개의 오류가 발생하는 경우에 제안된 소프트웨어 신뢰도 성장모형에서도 Jelinski-Moranda는 최우추정법으로 모수를 추정할 경우에 추정된 초기 오류는 추정 결과 0이나 ∞와 같이 엉뚱한 값으로 추정되는 문제가 자주 발생하는데, 우리가 이 연구에서 제안한 다중 오류를 고려한 소프트웨어 신뢰도 성장모형도 모수 추정 결과 똑같은 오류가 발생함을 알 수 있었다. 따라서 제안된 모형에서 이러한 오류가 모의 실험 결과 발생되지 않도록 하기 위해서 적당한 조건을 먼저 이론적으로 밝혀 낼 수 있도록 계속 연구하고자 한다.

참고문헌

[1] Barlow, R. E. and Scheuer, E. M. (1966), "Reliability growth during a development testing program," *Technometrics*, No. 8, pp. 53-60.

[2] Bittanti, S. (1988), *Software Reliability Modelling and Identification* (Lecture Notes in Computer Science), Spring-Verlag, Berlin Heidelberg.

[3] Jelinski, Z. and Moarnda, P. B. (1972), *Software reliability research, Statistical Computer Performance Evaluation*, W. Freiberger Ed., Academic Press, New York.

[4] Joe, H. and Reid, N. (1985), "Estimating the number of faults in a system," *Journal American Statatistical Association*, 80(389), pp. 222-226.

[5] Littlewood, B. and Verrall, J. L. (1973), "A Bayesian reliability growth model for computer software," *J. R. Statist. Soc., Series C*. 22(3), pp. 332-346.

[6] Moranda, P. B. (1975), "Predication of software reliability during debugging," *Proc. Ann. Reliability and Maintainability Symp.*, pp. 327-332.

[7] Miller, D. R. (1986), "Exponential order ststistic model of software reliability growth," *IEEE Trans. on Software Engineering*, SE-12, pp. 12-24.

[8] Musa, J. D. (1975), "A theory of software reliability and its application," *IEEE Trans. on Software Engineering*, SE-1, pp. 312327.

[9] Nayak, T. K. (1986), "Software reliability: Statistical modelling and estimation," *IEEE Trans. on Reliability*, R-35, pp. 566-570.

[10] Shooman, M. L. (1991), A micro software reliability model for prediction and test apportionment, Proceedings of 1991 International Symposium on Software Reliability Engineering, Texas Austin, pp. 52-59.

[11] Xie, M. (1991), *Software Reliability Modelling*, World Scientific Publishing Co., Singapore.

[12] Yamada, S. and Osaki, S. (1985), "Software reliability growth modeling: models and applications," *IEEE Trans. on Software Engineering*, SE-11, pp. 1432-1437.

[13] Zehna, P. W. (1966), "Invariance of maximum likelihood estimators," *Ann. Math. Statistics*, No. 66, pp. 891.