논문발표 B2

소프트웨어 개발 · 평가

- ◆ 병행처리 소프트웨어 시스템의 신뢰성 평가 모형
- ◆ 소프트웨어 안전성 평가를 위한 메트릭스들
- ◆ 소프트웨어 재사용에 따른 생산성 향상의 분석
- ◆ 소프트웨어 안전성 평가를 위한 시스템 결함 분석 기법
- ◆ 컴퍼넌트 재사용에 의한 소프트웨어 아키텍쳐 생성 프레임워크

KAIS '97 추계학술대회

병행처리 소프트웨어 시스템의 신뢰성 평가 모형

정은이 · 전철부 · 박만곤
부경대학교 자연과학대학 전자계산학과

요 약

소프트웨어 개발 수명주기 동안의 소프트웨어 테스팅 단계에서 Jelinski와 Moranda의 소프트웨어 고장 데이타 해석 모형이, 병행처리 소프트웨어 환경에서 나타날 수 있는 여러 개의 소프트웨어 오류가 각 테스팅 스테이지에서 다중적으로 발생하여 하나의 소프트웨어 고장의 원인이 되는 소프트웨어 테스팅 환경에서는 적절하지 않기 때문에, 다중 소프트웨어 오류가 하나의 고장을 유발하는 테스팅 데이타 해석을 위한 베이지안 소프트웨어 신뢰도 성장 모형을 제안하면서 몇가지 소프트웨어 신뢰성 측도들에 관해서 비정보 사전정보를 고려한 베이즈 추정량을 구한다. 그리고 제안된 베이지안 소프트웨어 신뢰도 측도들의 베이즈 추정량의 성능 평가를 위해서 몬테카를로 시뮬레이션을 수행하고 MSE와 Bias의 관점에서 성능을 비교한다.

1. 서 론

오늘날 세계 경제의 미래를 내다보면 정보 산업의 중요성은 빠르게 확산되고 있다. 이는 컴퓨터의 비용 효과(cost-effectiveness)의 증가와 특별한 관련이 있다. 이 비용 효과는 매 10년마다 약 1000가지의 요인에 의해 증가되어왔다[Musa[1985)]. 이와 같은 비율로 변화가 계속되는 동안, 컴퓨터에 의해 가장 경제적으로 조절될 수 있는 작업의 범위는 급속하게 증가할 것이다. 컴퓨터 시스템에 있어서 소프트웨어가 가장 중요한 부분인만큼 소프트웨어 공학 분야도 역시 빠른 속도의 발전을 기대할 수 있다. 오늘날 소프트웨어 공학분야에 영향을 끼치는 주된 요인들은 더욱 거대해지고 점점 증가하는 국제 사업 경쟁, 정보 시스템 자체 개발 비용과 정보 시스템의 실패에 따르는 비용, 더욱 빠르게 변화하는 컴퓨터 기술, 경영 정보 시스템 개발의 복잡도 증가등이 있다.

대부분의 정보 시스템 고객들은 사업가들이기 때문에 사업 경쟁이 더욱 심해질 수록 그들은 소프트웨어 제품의 필요성에 민감하며 동시에, 소프트웨어 제작자들이 더욱 경쟁을 해야 하는 것을 알기 때문에 그들은 점점 고도로 세련되고 많은 기능의 소프트웨어를 요구하게 된다. 소프트웨어 제작자들은 고객들의 요구를 완전하고 정확하게 이해해야 하고, 가장 중요한 요구 세가지는 요구된 품질의 수준, 양도시간, 그리고 비용이다.

품질, 양도시간, 비용은 주로 개발자의 특성들이기보다는 고객의 특성들이다. 양도시간, 비용에 대한 정량적 측정법은 존재하지만, 품질의 정량화는 어렵다. 소프트웨어 품질에 대한 구체적 측정법의 부재는 많은 소프트웨어 제품들이 빈약한 품질을 갖는 주된 이유일 것이다. 소프트웨어 신뢰도는 아마도 "소프트웨어 품질(software quality)"의 개념에서 가장 중요한 고유의 특성일 것이며, 신뢰도 그 자체는 얼마나 잘 소프트웨어가 고객의 요구와 일치하도록 동작하게 하느냐와 관련있기 때문이다. 따라서 소프트웨어의 신뢰도란 주어진 기간동안 주어진 조건하에서 소프트웨어 시스템이 고장나지 않고 사용할 수 있는 확률측도이다. 소프트웨어 공학과 신뢰도 분석의 학제간 연구분야가 되는 소프트웨어 신뢰성 공학은 다양한 소프트웨어 신뢰도 모형을 제안한 저자들에 의해 폭넓게 연구되어져 왔다[Jelinski and Moranda(1972),

Littlewood and Verall(1973, 1974), Moranda (1975), Musa(1979) and Shooman(1991)].

소프트웨어 개발과정에서 소프트웨어 시스템이 만족하게 수행될 것이라는 기대를 높이기 위해 소프트웨어 신뢰도 추정과 예측은 필수불가결하며, 설계단계에서 발생된 오류들은 테스팅단계에서 디버깅(debugging)되며, 그 테스팅 데이타에 의해서 소프트웨어 신뢰도는 예측되어진다.

소프트웨어의 신뢰성은 개발의 최종단계인 테스팅단계와 실제 운용단계에 있어서 소프트웨어 내에 잠재하는 오류수와 소프트웨어의 고장발생시간에 의해 평가할 수 있다. 소프트웨어 오류의 수가 감소하고 소프트웨어 고장간격시간이 늘어남으로써 테스팅단계 동안 오류 디버깅은 소프트웨어 시스템의 신뢰도를 향상시킨다. 이 현상을 소프트웨어 신뢰도 성장이라 한다.

소프트웨어 테스팅단계에서 소프트웨어 오류수와 고장간격시간에 의해 소프트웨어 고장현상을 수리적으로 모형화를 하면 소프트웨어에 대한 평가를 쉽게 할 수 있으며, 신뢰도 성장모형에 의해 소프트웨어 오류수, 소프트웨어 고장발생간격시간, 소프트웨어 신뢰도 및 고장률 등의 신뢰성 평가 측도들이 추정되어 예측할 수 있다.

본 논문에서는 소프트웨어 테스팅단계(testing phase)에서 Jelinski와 Moranda의 소프트웨어 고장 데이타 해석 모형이, 병행 처리 소프트웨어 환경에서 나타날 수 있는 여러 개의 소프트웨어 오류가 각 테스팅 스테이지(testing stages)에서 다중적으로 발생하여 하나의 소프트웨어 고장의 원인이 되는 소프트웨어 테스팅 환경에서는 적절하지 않기 때문에, 다중 소프트웨어 오류들에 의해 하나의 소프트웨어 고장을 유발하는 소프트웨어 테스팅 데이타 해석을 위한 베이지안 소프트웨어 신뢰도 성장 모형을 제 3절에서 제안하고, 제 4절에서는 몇가지 소프트웨어 신뢰성 측도들의 베이즈 추정량을 구한다. 그리고 제 5절에서는 제안된 베이지안 소프트웨어 신뢰도 측도들의 베이즈 추정량의 성능평가를 위해서 몬테카를로 시뮬레이션을 수행하고 MSE와 바이아스(Bias)의 관점에서 성능을 비교한다.

2. 소프트웨어 테스팅 환경에서의 모형선택

소프트웨어의 테스팅단계에서 테스팅 스테이지마다 하나의 소프트웨어 오류에 의해서 하나의 고장 데이타 발생을 고려하여 가장 일반적으로 사용되어지는 모형은 다음과 같은 가정을 두는 Jelinski와 Moranda모형이다.

[가정1] N은 소프트웨어에 존재하는 초기 오류의 총갯수이고, 미지이지만 고정된 상수이다.

[가정2] 테스팅 스테이지에서 고장의 원인은 한 개의 오류에 의해서 발생하고, 부대적인 오류는 발생하지 않으며 디버깅 절차는 완벽하다.

[가정3] 고장발생간격시간 $T_i = X_{(i)} - X_{(i-1)}$은 제각각 독립적으로 모수 $\lambda_i = (N-i+1)\phi$, i=1,2, ...,N인 지수분포를 따른다.

오늘날 병행처리 컴퓨팅 환경에서 병행처리 소프트웨어 시스템들을 테스팅하여 그 신뢰도를 평가하는 모형이 필요하게 되었다.

그러나 Jelinski와 Moranda 모형을 각 테스팅 스테이지에서 한 개의 오류에 의해서 한 개의 고장이 발생하기 때문에, 여러 개의 오류가 동시에 발생하여 한 개의 고장이 발생하는 경우 또는 한 개의 오류가 한 개의 고장으로 발생하더라도 연속적으로 고장 발생간격시간들이 거의 0에 가까울 때는 모형의 적용이 부적당하다. 따라서 Jelinski와 Moranda 모형에서 고장발생간격시간이 지수 분포를 따르는 가정을 좀 더 확장해야 할 필요성이 있다. 지수 분포의 자연스러운 확장은 감마분포이며 감마분포는 포아송 프로세스에서 여러 개의 연속적인 고장발생간격시간에 의해 유도되거나 일반적인 기하분포를 따르는 여러 개의 변수들의 합에 의해서도 구해질 수 있다. 그래서 여러 개의 오류가 동시에 발생하여 한 개의 고장이 발생하는 경우에는 다음과 같이 모형을 확장하여 가정을 설정할 수 있다.

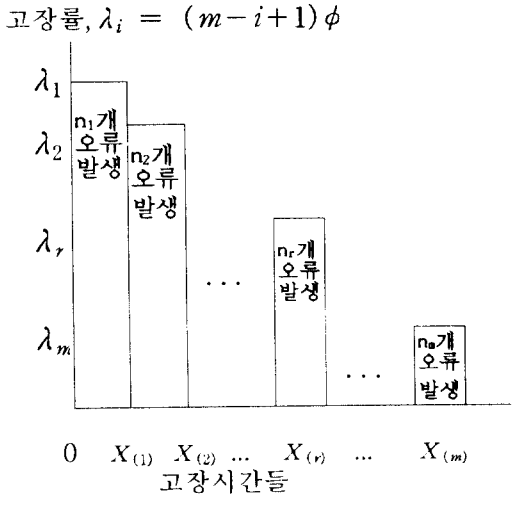
[가정1'] i번째 테스팅 스테이지에서 다중 소프트웨어 오류가 n_i번 발생하고 마지막 테스팅 스테이지 m은 미지이며 고정된 상수이다. 따라서 초기 오류의 총갯수 N은 $\sum_{i=1}^{m} n_i$이 된다.

[가정2'] 각 테스팅 스테이지에서 검출된 다중 소프트웨어 오류는 한 개의 고장을 발생시키며 다음 테스팅 스테이지이 시작되기전에 완전히 디버깅된다.

[가정3'] 테스팅 스테이지사이의 다중 소프트웨어 오류(Multiple Software Errors)에 의한 고장간격시간 $T_i = X_{(i)} - X_{(i-1)}$는 독립적으로 감마분포 $G(n_i, \lambda_i)$를 따르고, $\lambda_i = (m-i+1)\phi$, i=1,2,..., m, 이다. 고장간격시간 T_i의 확률밀도함수는

$$f(t_i|\lambda_i, n_i) = \frac{\lambda_i^{n_i} t_i^{n_i-1} \exp(-\lambda_i t_i)}{\Gamma(n_i)} \quad \cdots\cdots(1)$$

이다. 단, i=1,2,…,m에 대해서 $t_i, \lambda_i > 0$이고 $n_i \geq 1$이다. 단, ϕ는 소프트웨어 결함당 단위고장강도이다. $0 = X_{(0)} \leq X_{(1)} \leq \ldots \leq X_{(m)}$은 순서적 소프트웨어 고장시간이며 고장률은 <그림 1>과 같이 나타낸다.



<그림 1> 각 테스팅구간에 대응되는 소프트웨어 고장률

테스팅중인 소프트웨어 시스템의 고장간격시간 $T_1, T_2, \ldots, T_m$은 수정된 가정들 (1'), (2'), (3')의 테스팅 환경하에서 각각 다중 고장률 $\lambda_1, \lambda_2, \ldots, \lambda_m$을 가지는 독립적인 감마분포를 따른다. 따라서 소프트웨어 고장시간 $T_1, T_2, \ldots, T_m$의 결합확률밀도함수는 다음과 같이 구할 수 있다.

$$f(t_1, t_2, \ldots, t_m)$$
$$= \prod_{i=1}^{m} \frac{\lambda_i^{n_i} t_i^{n_i-1} \exp(-\lambda_i t_i)}{\Gamma(n_i)}$$
$$= \prod_{i=1}^{m} \frac{\{(m-i+1)\phi\}^{n_i} t_i^{n_i-1} \exp\{-(m-i+1)\phi t_i\}}{\Gamma(n_i)} \quad \cdots\cdots(2)$$

단, $\lambda_i = (m-i+1)\phi > 0$, $t_i > 0$ 이다.

3. 다중 오류 디버깅 프로세서를 위한 베이지안 모형화

Littlewood(1981)는 20년 이상 동안 최고의 소프트웨어 신뢰도 성장 모형으로 사용하여온 Jelinski와 Moranda 모형을 중심으로 하는 고전적인 모형에 대해 다음과 같은 몇가지 문제점이 있다고 지적하였다.

(1) 모형이 유도되어진 결함 카운팅 구조에서 N개의 결함에 모두 같은 양의 고장 강도가 고려되어진다.

(2) Formann과 Singpurwalla(1977, 1979)의 연구에서처럼 이 모형의 모수에 대한 표준적인 표본 이론의 접근으로 최우추정법의 사용이 잘못된 결과를 낳기로 한다.

따라서 이 문제를 해결하기 위한 많은 대체적인 모형이 제안되어져 왔다[Little -wood와 Verrall(1973), Goel과 Okumoto(1979)]. 그 중에서 많은 베이지안 모형들이 미지의 모수에 대해서 소위 사전정보를 결합하여 소프트웨어 고장 데이타를 해석하는 방법으로 제안되어져 왔다 [Littlewood와 Verrall(1973), Meinhold와

Singpurwalla(1983), Crow와 Singpurwalla(1984), Horigome(1984), Barlow와 Singpurwalla(1985), Langberg와 Singpurwalla(1985)].

그 중에서 Jelinski와 Moranda 모형을 바탕으로하여 베이지안 방법의 적용을 다룬 많은 연구는 Langberg와 Singpurwalla(1985), Jewell(1985), Littlewood와 Sofer(1987), Csenki(1990) 등이었다.

소프트웨어 신뢰성 분석 모형에 대한 베이지안 접근은 베이즈의 정리가 이용되며 다음과 같이 관계식으로 표현할 수 있다.

사후분포(*Posterior distribution*)

= 사전분포(*Prior distribution*)

$\times \frac{\text{우도함수}(Likelihood\ function)}{\text{주변분포}(Marginal\ distribution)}$.

여기에서 우도함수는 표본 소프트웨어 테스팅 데이타를 소프트웨어 신뢰도 측도의 사전정보로 변형하는 함수이며, 사전분포는 소프트웨어 신뢰도 측도들에 관한 알려진 또는 가정된 모든 정보를 표현하며, 사후분포는 관찰된 표본 소프트웨어 테스트 데이타를 기반으로하여 사전분포에 의해서 표현된 사전정보의 변경되고 업데이트된 정보를 나타내는 분포함수이다.

따라서 식(1)에서 Φ는 소프트웨어 결함(또는 오류) 단위당의 고장강도이며, m은 소프트웨어 테스팅을 시작하는 초기에 가정된 마지막 소프트웨어 테스팅 스테이지의 회차를 나타낸다. 베이지안 관점에서는 이 두 미지의 모수를 확률변수로 가정하여 사전정보를 할당할 수 있으며 본 논문에서 마지막 테스팅소단계 m은 모수 μ를 갖는 포아송사전분포 $P(\mu)$를 가정하였으며 m의 사전분포는

$$g_1(m=k)=\frac{\mu^k \exp(-\mu)}{k!},\ k=0,1,2,3\cdots\cdots,\ \mu>0, \quad \cdots\cdots(3)$$

와 같이 나타낼 수 있다.

또한 Jeffreys(1961)의 결과에 따라 Φ의 비정보(noninformative) 사전분포를 구해 보면

$$g_2(\phi) \propto I(\phi)^{\frac{1}{2}} \propto \frac{1}{\phi} \quad \cdots\cdots(4)$$

이며, 여기에서 피셔정보(Fisher's information)

$I(\phi)=-E\left(\frac{\partial^2}{\partial \phi^2}\ln L(\phi|\hat{t})\right)$, $0<\phi<\infty$이고,

$\hat{t}=\{t_1,t_2,\cdots,t_r\}$ 는 관찰된 소프트웨어 고장간격 시간들이다.

4. 비정보 사전지식을 이용한 소프트웨어 신뢰도 측도들의 베이즈 추정량

$\hat{t}=\{t_1,t_2,\cdots,t_r\}$이 r번째 테스팅소단계까지 관찰된 소프트웨어 고장간격시간들이고 k≥r 일 때, $\hat{t}, m=k$ 및 $\Phi=\phi$의 결합확률밀도함수는 다음과 같다.

$$\begin{aligned} &h(\hat{t}, m=k, \Phi=\phi) \\ &= f(\hat{t})\cdot g_1(m=k)\cdot g_2(\Phi=\phi) \\ &= \left(\prod_{i=1}^{r}\frac{((k-i+1)\phi)^{n_i}t_i^{n_i-1}\exp(-(k-i+1)\phi t_i)}{\Gamma(n_i)}\right) \\ &\quad \cdot \frac{\exp(-\mu)\mu^k}{k!}\cdot\frac{c}{\phi} \\ &= \left(\prod_{i=1}^{r}\frac{(k-i+1)^{n_i}t_i^{n_i-1}}{\Gamma(n_i)}\right)\frac{\exp(-\mu)\mu^k}{k!}\phi^{N_r-1} \\ &\quad \cdot \exp(-\phi M_r(k))\cdot c \end{aligned} \quad \cdots\cdots(5)$$

단, $N_r=\sum_{i=1}^{r}n_i$, $M_r(k)=\sum_{i=1}^{r}(k-i+1)t_i$, 그리고 c는 $g_2(\Phi=\phi)$에서 확률밀도함수의 조건을 만족시키기 위한 정규화상수(normalizing constant)이다.

여기서 베이즈 공식을 이용하여 $m=k$와 $\Phi=\phi$의 결합 사후분포를 구하면 다음과 같다.

$h(m=k, \Phi=\phi|\hat{t})$

$$= \frac{h(\hat{t}, m=k, \Phi=\phi)}{\sum_{k=r}^{\infty}\int_0^{\infty} h(\hat{t}, m=k, \Phi=\phi)d\phi}$$

$$= \frac{\left(\prod_{i=1}^{r}(k-i+1)^{n_i}\right)\frac{\mu^k}{k!}M_r(k)^{-N_r}}{\sum_{l=r}^{\infty}\left(\prod_{i=1}^{r}(l-i+1)^{n_i}\right)\frac{\mu^l}{l!}M_r(l)^{-N_r}}$$

$$\times \frac{M_r(k)^{N_r}\phi^{N_r-1}\exp(-\phi M_r(k))}{\Gamma(N_r)},$$

.....(6)

단, $M_r(k)$와 N_r은 식(5)에서 주어져 있다.

따라서 $\hat{t}$가 주어졌을 때 $m=k$의 주변 사후분포는

$$h(m=k|\hat{t})$$

$$= \frac{\left(\prod_{i=1}^{r}(k-i+1)^{n_i}\right)\frac{\mu^k}{k!}M_r(k)^{-N_r}}{\sum_{l=r}^{\infty}\left(\prod_{i=1}^{r}(l-i+1)^{n_i}\right)\frac{\mu^l}{l!}M_r(l)^{-N_r}}$$

.....(7)

이고, $\Phi=\phi$의 주변 사후 분포는

$$h(\Phi=\phi|m=k,\hat{t})$$

$$= \frac{M_r(k)^{N_r}\phi^{N_r-1}\exp(-\phi M_r(k))}{\Gamma(N_r)}$$

.....(8)

이다.

[정리1] 소프트웨어 테스팅 단계의 임의의 i번째 테스팅 스테이지에서 n_i개의 다중 소프트웨어 결함들이 i번째 소프트웨어 고장을 발생시켜서 이들의 디버깅을 수행하고 r번째 테스팅 스테이지까지의 고장간격시간 관찰 데이타 $\hat{t}=\{t_1,t_2,\cdots,t_r\}$를 얻었다. 지금까지의 최종 테스팅 스테이지 회차 m에 대해서 비정보 사전분포를 이용하여 제곱오차 결손함수하의 베이즈 추정량을 구하면

$$m^{BE} = \frac{\sum_{k=r}^{\infty}\prod_{i=1}^{r}(k-i+1)^{n_i}\frac{\mu^k}{(k-1)!}M_r(k)^{-N_r}}{\sum_{l=r}^{\infty}\prod_{i=1}^{r}(l-i+1)^{n_i}\frac{\mu^l}{l!}M_r(l)^{-N_r}}$$

.....(9)

이다. 단, $k\geq r>0, l\geq r$, $\mu>0$, $n_i\geq 1$이고

$M_r(k)=\sum_{i=1}^{r}(k-i+1)t_i$, $N_r=\sum_{i=1}^{r}n_i$다.

증명 베이즈 정리와 식(7)에 의해서

$$E(m|\hat{t})$$

$$= \sum_{k=r}^{\infty} k\cdot h(m=k|\hat{t})$$

$$= \frac{\sum_{k=r}^{\infty}\prod_{i=1}^{r}(k-i+1)^{n_i}\frac{\mu^k}{(k-1)!}M_r(k)^{-N_r}}{\sum_{l=r}^{\infty}\prod_{i=1}^{r}(l-i+1)^{n_i}\frac{\mu^l}{l!}M_r(l)^{-N_r}}.$$

[정리 2] 소프트웨어 테스팅 단계의 임의의 i번째 테스팅 스테이지에서 n_i 개의 다중 소프트웨어 결합들이 i번째 소프트웨어 고장을 발생시켜서 이들의 디버깅을 수행하고 r번째 테스팅 스테이지까지의 고장간격시간 관찰 데이타 $\hat{t}=\{t_1,t_2,\cdots,t_r\}$을 얻었다. 단위 고장강도 ϕ에 대해서 비정보 사전분포를 이용한 제곱오차 결손함수하의 베이즈 추정량을 구하면

$$\phi^{BE} = \frac{N_r}{M_r(k)}$$

.....(10)

이다.

증명 베이즈 정리와 식(8)에 의해서

$$E(\Phi|m=k,\hat{t})$$

$$= \int_0^{\infty}\phi h(\Phi=\phi|m=k,\hat{t})d\phi$$

$$= \int_0^{\infty}\frac{(\phi M_r(k))^{N_r}\cdot\exp(-\phi M_r(k))}{\Gamma(N_r)}d\phi$$

$$= \frac{N_r}{M_r(k)}$$

[정리 3] 소프트웨어 테스팅 단계의 임의의 i번째 테스팅 스테이지에서 n_i 개의 다중 소프트

웨어 결합들이 i번째 소프트웨어 고장을 발생시켜서 이들의 디버깅을 수행하고 r번째 테스팅 스테이지까지의 고장간격시간 관찰 데이타 $\hat{t}=\{t_1,t_2,\cdots,t_r\}$을 얻었다. $N_r=\sum_{i=1}^{r}n_i$개의 소프트웨어 결함이 r번째 테스팅 스테이지까지 검출되어 디버깅된 직후부터 어떤 시간구간 $t(<t_{r+1})$까지의 소프트웨어 신뢰도에 대해서 비정보 사전분포를 이용한 제곱오차 결손함수하의 베이즈 추정량을 구하면

$$R^{BE}(t)=1-\frac{\sum_{k=r+1}^{\infty}\prod_{i=1}^{r}(k-i+1)^{n_i}\frac{\mu^k}{k!}M_r(k)^{-N_r}}{\sum_{l=r}^{\infty}\prod_{i=1}^{r}(l-i+1)^{n_i}\frac{\mu^l}{l!}M_r(l)^{-N_r}}$$

$$\times\sum_{j=0}^{N_r-n_{r+1}}\binom{N_r}{n_{r+1}+j}P(k)^{n_{r+1}+j}(1-P(k))^{N_r-j-n_{r+1}}$$

.....(11)

이다. 단, $P(k)=\frac{(k-r)t}{M_r(k)+(k-r)t}$, $k\geq r$.

증명 베이즈 정리와 식(6)에 의해서 계산할 수 있으며, 또한 Park등(1994)의 결과에 따라 공액사전분포로 사용된 감마분포를 비정보 사전분포의 조건으로 바꾸면 쉽게 구할 수 있다.

5. 몬테카를로 시뮬레이션

최종 소프트웨어 테스팅 스테이지의 회차 m, 단위고장강도 ϕ, 그리고 r번째 테스팅 스테이지 직후부터 어떤 시간구간 $t(<t_{r+1})$까지의 소프트웨어 신뢰도 R(t)에 대한 베이즈 추정량 $m^{BE}, \phi^{BE}, R^{BE}(t)$에 대한 바이아스(Bias)와 평균제곱오차(Mean-Squared Error)의 관점에서 성능을 평가하였다. 성능의 평가는 IMSL 소프트웨어 패키지를 사용하여 몬테카를로 시뮬레이션을 수행하였으며 부경대학교 자연과학대학 전자계산학과의 MV-15000시스템을 사용하였다. 시뮬레이션 절차는 감마 확률변수를 발생시켜서 고장간격시간들을 생성하였으며 500번의 반복과정을 통해서 베이즈 추정량들의 바이아스와 MSE를 계산하여 <표 1>에서 <표 6>까지에 나타내었다.

<표 1> 지금까지의 최종 소프프웨어 테스팅 스테이지 회차 m에 대한 베이즈 추정량 m^{BE}의 MSE 성능 비교

m=k \ μ	1	3	5
3	0.07714	0.97995	3.59365
5	0.05990	0.62019	2.30008
7	0.04238	0.36602	1.40778
9	0.03208	0.32606	0.96281
15	0.03053	0.24891	0.57793
20	0.02413	0.19932	0.51532
30	0.02013	0.16304	0.36556
50	0.01897	0.15567	0.34901

MSE(m^{BE})

<표 2> 지금까지의 최종 소프트웨어 테스팅 스테이지 회차 m에 대한 베이즈 추정량 m^{BE}의 Bias 성능비교

m=k \ μ	1	3	5
3	0.02856	0.69972	0.882321
5	0.02372	0.50212	0.87506
7	0.01517	0.37224	0.66222
9	0.01415	0.30207	0.53611
15	0.00956	0.27342	0.48150
20	0.00943	0.16416	0.45904
30	0.00924	0.15634	0.40720
50	0.00873	0.13242	0.32549

Bias(m^{BE})

<표 3> 소프트웨어 결함당 단위고장강도 ϕ의 베이즈 추정량 ϕ^{BE}에 대한 MSE 성능비교

m=k \ μ	1	3	5
3	0.00751	0.28874	0.68832
5	0.00438	0.27776	0.68422
7	0.00378	0.23544	0.67212
9	0.00238	0.22034	0.65832
15	0.00206	0.18704	0.58153
20	0.00203	0.17470	0.53332
30	0.00181	0.11174	0.42234
50	0.00169	0.10998	0.39987

MSE(ϕ^{BE})

<표 4> 소프트웨어 결함당 단위고장강도 ϕ의 베이즈 추정량 ϕ^{BE}에 대한 Bias 성능비교

m=k \ μ	1	3	5
3	0.06475	0.07631	0.14461
5	0.06538	0.07375	0.11436
7	0.03719	0.07264	0.09842
9	0.03867	0.07010	0.08136
15	0.03762	0.06713	0.07036
20	0.00256	0.04008	0.05136
30	0.01561	0.04001	0.048076
50	0.01421	0.03665	0.03890

Bias(ϕ^{BE})

<표 5> r번째 테스팅 스테이지 직후부터 시간구간 t까지의 소프트웨어 신뢰도 R(t)에 대한 베이즈 추정량 $R^{BE}(t)$의 MSE 성능비교

m=k \ μ	1	3	5
3	0.00612	0.02280	0.04581
5	0.00389	0.01416	0.02328
7	0.00235	0.01098	0.01556
9	0.00223	0.00618	0.01286
15	0.00153	0.00457	0.01118
20	0.00057	0.00328	0.00635
30	0.00038	0.00145	0.00472
50	0.00033	0.00133	0.00445

MSE($R^{BE}(t)$)

<표 6> r번째 테스팅 스테이지 직후부터 시간구간 t까지의 소프트웨어 신뢰도 R(t)에 대한 베이즈 추정량 $R^{BE}(t)$의 Bias 성능비교

m=k \ μ	1	3	5
3	0.01349	0.02923	0.04372
5	0.01047	0.02277	0.03038
7	0.00784	0.01925	0.02349
9	0.00638	0.01278	0.02050
15	0.00543	0.01190	0.01962
20	0.00339	0.00891	0.01308
30	0.00277	0.00657	0.01194
50	-0.00245	-0.00655	-0.00999

Bias($R^{BE}(t)$)

5. 결 론

본 연구에서는 소프트웨어 개발 수명주기 동안의 소프트웨어 테스팅단계에서 Jelinski와 Moranda의 소프트웨어 고장 데이타 해석 모형은 오늘날의 병행/병렬처리 소프트웨어 개발환경에서 나타날 수 있는 여러 개의 소프트웨어 오류가 각 테스팅 스테이지에서 다중적으로 발생하여 하나의 소프트웨어 고장의 원인이 되는 소프트웨어 테스팅 환경에서는 적절하지 않기 때문에 다중 소프트웨어 오류들에 의해 하나의 소프트웨어 고장을 유발하는 소프트웨어 테스팅 데이타 해석을 위한 베이지안 소프트웨어 신뢰도 성장 모형을 제안하고 소프트웨어 신뢰성 측도들의 베이즈 추정량을 구하였다. 그리고 제안된 베이지안 소프트웨어 신뢰도 측도들의 베이즈 추정량의 성능평가를 위해서 컴퓨터 시스템상에서 IMSL을 사용하여 몬테카를로 시뮬레이션을 수행하고 MSE와 바이아스의 관점에서 성능을 비교한 결과 <표 1>에서 <표 6>까지의 결과를 얻었다. 이에 따라 다음의 결론을 얻을 수 있었다.

(1) 최종 소프트웨어 테스팅 스테이지의 회차 m에 대해서

▶ MSE(m^{BE})와 Bias(m^{BE})는 μ의 값이 클수록 증가한다.

▶ MSE(m^{BE})와 Bias(m^{BE})는 m=k의 값이 클수록 감소한다.

(2) 소프트웨어 결함당 단위고장강도 ϕ에 대해서

▶ MSE(ϕ^{BE})와 Bias(ϕ^{BE})는 μ의 값이 클수록 증가한다.

▶ MSE(ϕ^{BE})와 Bias(ϕ^{BE})는 m=k의 값이 클수록 감소한다.

(3) 소프트웨어 신뢰도 R(t)에 대해서

▶ MSE($R^{BE}(t)$)와 Bias($R^{BE}(t)$)는 μ의 값이 클수록 증가한다.

▶ MSE($R^{BE}(t)$)와 Bias($R^{BE}(t)$)는 m=k의 값이 클수록 감소한다.

(4) 테스팅 스테이지 r이 증가할수록 모든 베이즈 추정량의 MSE와 Bias가 감소한다.

(5) 비정보 사전분포의 사용은 공액사전분포의 사용보다 베이즈 추정량의 식이 간편하고 robust하다.

(6) 여러 개의 소프트웨어 오류가 병발하여 하나의 고장을 일으키는 테스팅 환경에서 이 다중 소프트웨어 오류를 디버깅하는 절차를 고려한 제안된 베이지안 모형은 대단히 적절하며 Littlewood가 지적한 문제점들을 모두 해결하고 있다.

참 고 문 헌

[1] Barlow, R. E. and N. D. Singpurwalla, "Assessing the Reliability of Computer Software and Computer Networks : an Opportunity for Partnership with Computer Science," American Statistician, vol. 39, pp. 88-94, 1985.

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

[3] Csenki, A., "Bayes Predictive Analysis of a Fundamental Software Reliability Model," IEEE Trans. on Reliability, vol. 39, pp.177-183, 1990.

[4] Crow, L. H. and Singpurwalla, N. D., "An Empirically Deceloped Fourier Series Model for Describing Software Failure," IEEE Trans. on Reliability, vol. 33, pp.176-183, 1984.

[5] Forman, E. H. and N. D. Singpurwalla, "An empirical stopping rule for debugging and testing computer software," J. American Statistical Association, 72, pp.750-757, 1977.

[6]Forman, E. H. and N. D. Singpurwalla, "Optimal time intervals for testing hypotheses on computer software errors," IEEE Trans. on Reliability, R-28, pp.250-253, 1979.

[7]Goel, A. L. and Okutomo, K., "Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures," IEEE Trans. on Reliability, vol. 28, pp.206-211, 1979.

[8] Horigome, M. et al., "A Bayes Empirical-Bayes Approach for Software Reliability

Growth,", In Computer Science and Statistics, (16th Symp, Interface, Atlanta, GA, 1984), North-Holland, pp.47-55, 1985.
[9] Jelinski, Z. and P. B. Moranda, *Software Reliability Research. Statistical Computer Performance Evaluation,* W.F reiberger Ed., Academic Press, pp.465-484, 1972.
[10] Jewell, W. S, "Bayesian Extensions to a Basic Model of Software Reliability," IEEE Trans. on the Software Engineering, vol. 11, pp.1465-1471, 1985.
[11] Langberg, N. and N. D. Singpurwalla, "A unification of some software reliability models," SIAM J. Scientific and Statistical Computation, 6, pp.781-790, 1985.
[12] Littlewood, B., "Stochastic reliability-growth : A model for fault-removal in computer-programs and hardware design," IEEE Trans. on Reliability, R-30(4), pp. 313-320, 1981.
[13] Littlewood, B. and A. Sofer, "A Bayesian Modificaion to the Jelinski-Moranda Software Reliability Grwoth Model," Software Engineering J., vol. 2, pp.30-41, 1987.
[14] Littlewood, B. and J. L. Verrall, "A Bayesian reliability growth model for computer software," J. R. Statist. Soc., Series C, 22(3), pp.332-346, 1973.
[15] Littlewood, B. and J. L. Verrall, "A Bayesian reliability model with a stochastically monotone failure rate," IEEE Trans. on Reliability, R-23(2), pp.108-114, 1974.
[16] Meinhold, R. J. and Singpuwalla, N. D., "Bayesian Analysis of a Commonly Used Model for DEscribing Sofrware Filures," The Statistician, vol. 32, pp.168-173, 1983.
[17] Moranda, P .B., "Prediction of software reliability during debugging," Proc. Ann. Reliability and Maintainability Symposium, pp. 327-332, 1975.
[18] Miller, D. R., "Exponential order statistic model of software reliability growth," IEEE Trans. on Software Engineering, SE-12, pp. 12-24, 1986.
[19] Musa, J. D., "A theory of software reliability and its application," IEEE Trans. on Software Engineering, SE-1, pp.312-327, 1975.
[20] Musa, J. D., "Validity of the excution time theory of software reliability," IEEE Transaction on Reliability, R-28, pp.181-191, 1979.
[21] Musa, J. D., "Software Engineering : The Future of a Profession," IEEE Software 2(1), pp.55-62, 1985.
[22] Musa, J. D., A.Iannino and K.Okumoto, *Software Reliability: Masurement, Prediction, Application,* McGraw-Hill Book Company, N.Y., 1987.
[23] Park, M. G. and R. H. Dean, "Bayesian Algorithm for Evaluation and Prediction of Software Reliability," The Journal of Korea Information Processing Society, Vol.1, No.1, pp.14-22, 1994.
[24] Park, M. G, I. S. Lee and H. J. Jeong, "Multiple Software Errors Debugging Process Based on the Littlewood-Verrall Models," Proc. of the 1st China-Korea Symposium on QM, pp.109-114, 1994.
[25] Park, M. G, I. S. Lee and H. J. Jeong, "Bayesian Software Reliability Estimation for Multiple Errors Debugging Process," Proc. 1994 Taipei International Quality Conference, pp.703-708, 1994.
[26] Shooman, M. L., *Probabilistic Models for Software Reliability Prediction. Statistical Computer Performance Evaluation,* W. Freiberger Ed., Academic Press, New York, 1972.
[27] Shooman, M. L., *Software Engineering : Design, Reliability, and Management,* McGraw-Hill Book Company, New York, 1983.
[28] Shooman, M. L., "A micro software reliability model for prediction and test apportionment," Proc. of 1991 International Symp. on Software Reliability Engineering, Texas Austin, pp.52-59, 1991.
[29] Xie, M.,*Software Reliability Modelling,* World Scientific Publishing Co. Singapore, 1991.
[30] Yamada, S. and S. Osaki, "Software reliability growth modeling : models and applications," IEEE Trans. on Software Engineering, SE-11, pp.1431-1437, 1985.