NHPP 분포를 이용한 S/W의 초기 에러 예측

장원석, 최규식, wschang@kytis.konyang.ac.kr, 건양대학교 정보전자통신공학부

Initial error estimation of software by NHPP distribution

Chang Won-Seok, Che Gyu-Shik, Konyang University

요약 - 소프트웨어의 신뢰도는 하드웨어의 신뢰도와 고장메카니즘이 다르므로 하드웨어의 신뢰도 모델을 그대로 이용할 수 없다. 소프트웨어의 신뢰도를 추정하기 위한 방법은 그동안 Jelinski-Moranda(JM) 모델을 비롯하여 많은 기법이 연구되었다. 그러나, 아직까지 만족하다고 인정할 만한 신뢰도모델링은 개발되지 않았다.

본 연구에서는 소프트웨어의 테스트를 통하여 검출되는 에러 갯수의 추세를 가지고 비제차포아송과정(NHPP)의 파라미터를 찾아 신뢰도함수를 구하고자 하며, 아울러, 테스트중단시간을 결정하고자 한다. 파라미터를 찾는 방법은 maximum likelihood estimate(MLE) 기법을 이용하며, 테스트 중단시간은 구해진 파라미터를 신뢰도 함수에 대입하여 결정한다.

1. 서론

S/W 고장은 S/W가 개발될 당시 결함으로서 잠재하고 있다가 어떤 특수 입력 자료에 의해 발견될 때 비로소 나타나게 된다. S/W의 신뢰도 관리는 어떻게 하면 이러한 결함을 S/W의 개발 단계에서 발생하지 않도록 예방하느냐 하는 것과, 이렇게 개발된 S/W중의 결함을 어떠한 테스트 방법을 통해서 검출할 것이냐, 그리고, 어느 정도 검출이 되었을 때 결함을 예측하여 테스트를 중단하고 그 예측에 근거하여 S/W를 발표하는 것이냐 하는 문제가 된다.

본 연구에서는 S/W의 테스트시 검출되는 에러에 의해서 개발초기부터 S/W 내에 존재하고 있는 에러의 수를 예측하는 방법을 제시하고, 테스트 중단시간을 결정하는 기법을 찾고자 하였다.

2. S/W 결함

S/W의 신뢰도는 그 S/W가 가지고 있는 결함에 의해서 좌우되므로 이러한 결함의 원인을 분석하는 것은 매우 의의가 깊다. Thayer의 연구에서 4 개의 프로젝트에서 오류를 수집하여 결함 분석한 결과 자료를 표 2-1과 같이 보고하고 있다.

이 표에 의하면 프로젝트 A, B, C, D 공히 논리 결함이 가장 큰 부분을 차지하고 있음을 알 수 있다.

표 2-1 오류 발생 빈도 순위

순위	프로젝트A	프로젝트B	프로젝트C	프로젝트 D			
				응용 S/W	시뮬레이터	OS	PA방법
1	논리	논리	논리	DB	논리	논리	논리
2	인터페이스	데이터 취급	데이터 취급	논리	연산	데이터 취급	데이터 취급
3	데이터 취급	인터페이스	인터페이스	연산	DB	I/O	I/O
4	연산	I/O	I/O	데이터 취급	데이터 정의	인터페이스	데이터 정의
5	I/O	연산	데이터 정의	인터페이스	I/O	데이터 정의	DB
6	DB	DB	DB	I/O	데이터 취급	DB	인터페이스
7	데이터 정의	데이터 정의	연산	데이터 정의	인터페이스	연산	연산

3. S/W 신뢰도 모델

S/W의 신뢰도를 평가하는 모델은 크게 시간 종속적 모델(time-dependent model)과 시간 독립적 모델(time-dependent model)로 나눌 수 있다.

시간종속적 모델은 S/W가 가동중인 시간에 의존하여 고장이 발생된다는 가정 하에 신뢰도를 평가하고자 하는 모델이며, 시간독립적 모델은 일정한 테스트 입력을 이용하여 소프트웨어를 시험한 결과를 근거로 신뢰도를 측정하는 모델이다.

소프트웨어의 신뢰도를 측정하는 최초의 모델이자 가장 많이 쓰이는 모델로는 Jelinski-Moranda(JM) 모델이 있으며, 이 모델에서 소프트웨어의 고장률은 어떠한 시간에서든지 현재 남아 있는 소프트 웨어의 결함(fault)의 수에 비례한다.

그 이후 Schik-Wolverton(SW), Littlewood-Verral Bayesian, Goel-Okumoto NHPP, Goel Generalized

NHPP, Shooman experimental, Mill's seeding, Nelson, Ramamoothy-Bastani 등의 모델이 제시되었다.

본 연구에서 적용하고자 하는 모델은 Goel-Okumoto의 비동차포아송과정(NHPP)이다. NHPP는 유한장고장모델이며, 기본모델은 아래와 같다.[1,2,3,4,5]

$$P[N(t)=y]=\frac{(m(t))^y}{y!}e^{-m(t)},\quad y=0,1,2,\ldots\ldots \quad (3.1)$$

여기서, $m(t)$는 시간 t 동안 검출되는 에러의 누적갯수이며, 평균치함수로서

$$m(t)=a(1-e^{-bt}) \quad (3.2)$$

이다. 고장강도 $\lambda(t)$는

$$\lambda(t)=m'(t)=abe^{-bt} \quad (3.3)$$

이다. 아울러 시간 t에서 검출에러가 m_o 되는 에러의 누적분포함수를 다음과 같이 정의한다

$$F[m_o]=\sum_{y=0}^{m_o}\frac{(m(t))^y}{y!}e^{-m(t)},\quad y=0,1,2,\ldots \quad (3.4)$$

4. S/W의 신뢰도 모델 개발

신뢰도 추정을 위해서는 위에서 도입한 파라미터를 구하는 것이 핵심이라 할 수 있을 정도로 소프트웨어의 예측에는 여기에 적용하는 파라미터를 구하는 것이 중요하다.

4.1 maximum likelihood estimate(MLE) 기법에 의한 파라미터 추정

MLE의 기본 이론은 likelihood 함수가 최대로 되도록 추정치를 선택하도록 하는 것이다. likelihood 함수는 관찰된 표본치가 가능한 파라미터값의 함수와 얼마나 같아지느냐 하는 것이다. $x_1, x_2, \cdots, x_n$을 관찰된 표본치라 하고, θ를 추정하고자 하는 파라미터라고 하면

$$L=pdf\{\theta \mid x_1, x_2, \cdots, x_n\}=\prod_{i=1}^{n}pdf\{x_i \mid \theta\} \quad (4.1)$$

으로 정의한다.[6] pdf는 확률밀도함수이며, 파라미터 추정치 $\hat{\theta}$는 likelihood 함수 L이 최대인 것으로 가정하는 θ의 평가치로 정한다. 함수가 최대치가 되게 하려면 미지 파라미터에 관하여 미분하여 그 값을 0으로 하는 값을 찾아내는 것이다. 예를 들어 고장시간 t_i가 지수적으로 분포되고 고장률이 상수 λ일 경우의 MLE 추정치를 구하면 아래와 같다.

$$L=\prod_{i=1}^{n}\lambda\exp[-\lambda t_i]=\lambda^n\exp[-\lambda\sum_{i=1}^{n}t_i] \quad (4.2)$$

식(4.1)의 정의에 의하여 식(3.1)과 같은 NHPP인 경우의 likelihood 함수는 다음과 같이 쓸 수 있다.

$$L(a,b)=\prod_{k=1}^{n}\frac{\{m(t_k)-m(t_{k-1})\}^{(y_k-y_{k-1})}}{(y_k-y_{k-1})!}\cdot e^{-m(t_k)+m(t_{k-1})} \quad (4.3)$$

여기서, y_k는 시간간격 $[0, t_k]$ (k=1,2.$\cdots$,n ; $0<t_1<t_2<\cdots<t_n$)에서 검출되는 고장의 누적갯수이고, t_o=0, y_o=0이다.

우함수 $L(a,b)$를 최대로 하는 파라미터 a, b를 구하기 위해서 a와 b 각각에 대해서 $L(a,b)$의 편도함수를 0으로 놓고 연립방정식을 풀어야 한다. 계산을 좀더 단순화하기 위해 양 변에 대수를 취하여 편미분한다.

$$\ln L(a,b)=\sum_{k=1}^{n}(y_k-y_{k-1})\cdot\ln[m(t_k)-m(t_{k-1})]-\sum_{k=1}^{n}[m(t_k)-m(t_{k-1})]-\sum_{k=1}^{n}\ln[(y_k-y_{k-1})!]$$
$$=\sum_{k=1}^{n}(y_k-y_{k-1})\cdot\ln[a\{1-e^{-bt_k}\}-a\{1-e^{-bt_{k-1}}\}]-a\{1-e^{-bt_n}\}-\sum_{k=1}^{n}\ln[(y_k-y_{k-1})!] \quad (4.4)$$

식(4.4)를 a와 b에 대하여 편미분하여 정리하면

$$a=\sum_{k=1}^{n}\frac{(y_k-y_{k-1})}{1-e^{-bt_n}},$$

$$\sum_{k=1}^{n}(y_k-y_{k-1})\frac{t_ke^{-bt_k}-t_{k-1}\cdot e^{-bt_{k-1}}}{e^{-bt_{k-1}}-e^{-bt_k}}-at_n\cdot e^{-bt_n}=0 \quad (4.5)$$

이다.

4.2 테스트 중단시간 결정

S/W를 개발하여 발행하기 전에 테스트를 많이 하면 할수록 소프트웨어의 신뢰성은 높아지게 된다. 그러나, 무조건 테스트를 많이 할 수는 없다. 한정된 자원과 시간 하에서 소프트웨어를 개발해야 하기 때문이다. 그러므로, 사용자가 만족할 수 있는 시점에서 소프트웨어 테스트를 중지하여 소프트웨어를 출하시켜야 한다. 하드웨어는 설계단계에서 이미 신뢰도가 결정되는 것에 비해 소프트웨어는 설계단계에서 소프트웨어의 신뢰도가 결정되는 것이 아니다. 일반적으로 테스트를 수행하면서 소프트웨어의 에러를 수정하면 소프트웨어의 신뢰도가 증가된다. 그래서, 본 장에서는 소프트웨어의 신뢰도를 만족시키기 위해서 테스트 시간을 정하는 것을 제안하고자 하는 것이다.

식(3.2)에서 최초에 S/W 내에 존재하고 있던 에러의 수는 $t\to\infty$일 때의 값 즉, $m(\infty)=m_\infty=a$이므로 시간 t_o에서 m_o 개의 에러가 발견되었을 때 테스트를 중단한다고 하면

$$m_o=a(1-e^{-bt_o})=m_\infty(1-e^{-bt_o}) \quad (4.6)$$

이므로, 중단시간을 구해보면

$$t_o=-\frac{1}{b}\ln\left(1-\frac{m_o}{m_\infty}\right) \quad (4.7)$$

이다.

4.3 파라미터 추정

지금까지의 경험에 의하면 소프트웨어의 에러는 한 번 발견되어 수정되면 이 부분은 다시 에러를 일으키지 않는 것으로 밝혀졌으므로, 에러를 수정하면 할수록 내장된 에러의 수가 줄어들어서 에러의 발견횟수도 줄어들게 된다. 그러므로, 테스트를 하면 할수록 S/W의 신뢰도는 높아지게 된다.

어느 S/W를 테스트하여 다음 표 4.1과 같은 에러가 발견된 경우에 대해서 고찰해보기로 한다

MLE 추정법에서 유도한 식(4.5)에 에러의 수를 입력으로 하는 프로그램에 의해 파라미터의 값을 구하면 a=254.988, b=0.129가 된다. 여기에 적합한 모델을 구해보면 에러의 예상 갯수는 $m(t)=254.988(1-e^{-0.129t})$이고, 고장강도는 $\lambda(t)=m'(t)=32.893e^{-0.129t}$이다. 이 모델의 정확도를 판단하기 위해 본래 테스트에 의해서 구

표 4.1 에러 검출

횟수	검출에러	누적에러	횟수	검출에러	누적에러
1	33	33	10	9	185
2	26	59	11	8	193
3	22	81	12	8	201
4	21	102	13	6	207
5	19	121	14	6	213
6	17	138	15	5	218
7	13	151	16	4	222
8	13	164	17	4	226
9	12	176	18	4	230

한 실제 값과 모델 $m(t)$에 의해서 해석적으로 구한 값들을 그림으로 나타내면 그림 4.1과 같으며, 그림에서 알 수 있는 바와 같이 실제 검출된 누적치는 때로는 들쭉날쭉한 모습을 나타내어 원활하지 못하지만, 해석적으로 구한 것은 원활한 곡선이 된다. 그럼에도 불구하고 양 그래프 사이에 큰 차이가 없음을 알 수 있다. 따라서, 위의 경우 S/W 내에 처음부터 포함되어있는 에러의 수는 $m(\infty)=m_\infty=255$개로 예측할 수 있고, 고장강도 $\lambda(t)$는 상수가 아니라 지수함수의 형태를 하고 있다. 고장강도는 그림 4.2와 같다.

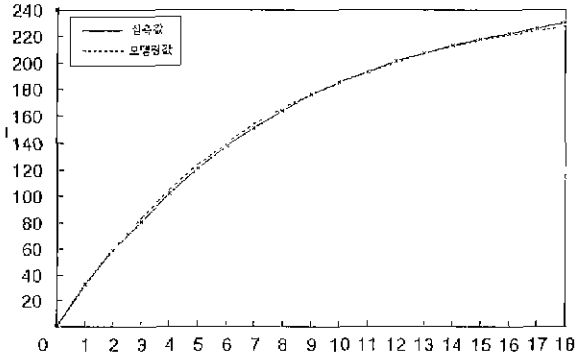

그림 4.1 누적에러의 수

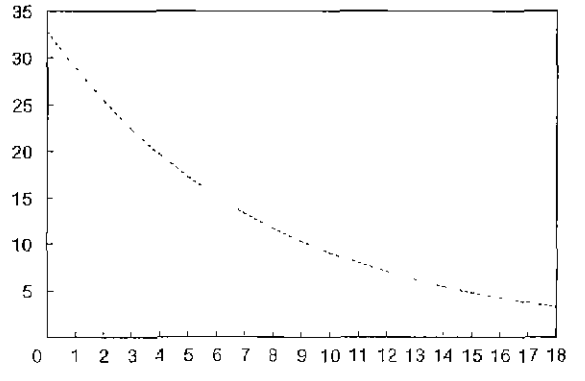

그림 4.2 고장강도

테스트에 의해서 검출되는 에러의 경향이 그림 4.1과 같으므로 어느 시점에서 테스트를 중단하고 S/W를 발행할 것인가를 결정하기 위해서 식(4.8)을 적용한다. 잔여 에러의 수가 5%이내로 되는 경우 즉, 검출된 에러의 수가 95%이상이 되는 경우를 예로 들면

$$t_0 = -\frac{1}{0.129}\ln(1-0.95) = 23.23$$

즉, 24회 이상 에러를 검출하면 에러의 수를 5% 이내로 줄일 수 있다.

5. 결론

S/W는 프로그램 내에 처음부터 존재하고 있던 결함에 의해 에러가 발생되고 경과되는 시간보다는 입력되는 데이터의 성질에 따라 고장을 일으킨다. S/W내에 존재하고 있던 에러를 검출하여 수정하면 이 부분은 다시 에러를 일으키지 않을 것이므로 에러를 수정하면 할수록 신뢰도가 좋아진다. 이것이 세월이 지남에 따라 신뢰성이 저하되는 H/W와 판이하게 다른 점이다.

S/W 신뢰도 모델기법은 JM 모델기법을 비롯하여 여러 가지 모델이 있으나, 본 연구에서는 NHPP 기법을 이용하여서 실제의 경우에 대하여 maximum likelehood estimation (MLE)기법을 이용하여 파라미터 a, b를 구하여 신뢰도함수를 완성하였다. 또한, 구한 누적에러곡선에 의해서 언제 S/W의 테스트를 중단해야 하는가 하는 테스트 중단시간도 정량적으로 예측하였다.

본 연구를 통하여 MLE 기법을 이용하면 검출되는 에러의 추세에 의해서 신뢰도 모델링함수의 파라미터를 구할 수 있고, 이 구해진 신뢰도함수에 의해서 테스트 중단시간도 결정할 수 있다는 것을 고찰하였다

96-03

본 연구는 한국전력공사의 지원에 의하여 기초전력공학공동연구소 주관으로 수행되었음

[참고문헌]

1. Hiroshi Ohtera, Shigeru Yamada, "Optimum Software-Release Time Considering an Error-Detection Phenomenon during Operation", IEEE Trans. on Reliability, vol.39.No.5, 1990.12

2. Shigeru Yamada, Hiroshi Ohtera, Hiroyuki Narihisa, " Software Reliability Growth Models with Testing Efforts", IEEE Trans. on Reliability, vol.R-35,No.1 1986.4

3. Shigeru Yamada, Jun Hishitani, Shunji Osaki, "Software-Reliability Growth with a Weibull Test-Effort : A Model & Application", IEEE Trans. on Reliability, vol.42.No.1, 1993.3

4. Hsin-Hui Lin, Kuang-Hwa Chen, "Nonhomogeneous Poisson Process Software-Debugging Models with Linear Dependence", IEEE Trans. on Reliability, vol.42.No.4. 1993.12

5. W. Y. Yun, D. S. Bai, "Optimum Software Release Policy with Random Life Cycle", IEEE Trans. on Reliability, vol.39.No.2, 1990.6

6. Michael A. Friedman, Jeffrey M. Voas, " Software Assessment -Reliability, Safety, Testability", John Wiley & Sons, Inc. pp171-192, 1996