▨연구논문

다중 고장 유형과 불완전 수정하에서의 소프트웨어 신뢰도 모델

문숙경
목원대학교 응용통계학과

Modeling Software Reliability with Multiple Failure types and Imperfect Debugging

Sook Kyung Moon
Dept. of Applied Statistics, Mokwon University

Abstract

This paper presents a software reliability model that is based on a nonhomogeneous poisson process. The major contribution of this model is combining multiple failure types with imperfect debugging by use of S-shaped mean value function. The software reliability model allows for three different types of errors: Critical errors are the most difficult to detect and the most expensive to remove. Major errors are moderately difficult to detect and fairly expensive to remove. Minor errors are easy to detect and inexpensive to remove. The model also allows for the introduction of any of these types of errors during the removal of an error. A numerical example is provided to illustrate the above techniques.

1. 서론

최근 정보화 추세의 근간은 컴퓨터와 통신 분야의 눈부신 발전에 기인해왔다고 볼 수 있다. 컴퓨터는 나날이 그 용량이나 속도면에서 놀라운 발전을 거듭해오고 있으며, 이에 따라 소프트웨어 산업 역시 엄청난 규모를 자랑하며 사회 전반의 필수품화되어 가고있는 실정이며, 따라서, 이전엔 상상도할 수 없었던 대규모의 복잡한 프로그램 개

발에 박차를 가하고 있다. 이처럼, 프로그램이 대형화 고급화 할수록 많은 인력과 장비가 투입되는데 이 때, 최적의 설계, 개발 방법 및 최신의 툴(tools)들이 사용되어지지만, 프로그래머들에 의해 발생되어지는 에러들을 완벽하게 방지하기란 현재로서도 불가능한 것이 사실이다. 이처럼 이런 저런 결과로 야기된 에러들이 일부 사용자들에겐 일종의 불편함 정도로만 느껴질 수 있지만, 한편, 고도의 정밀함과 안전성이 요구되는 최첨단 시스템일 경우, 대체로 이런 경우 사람의 생명과 안전에 직결되는 경우가 허다하며, 이런 경우엔 비록 단 하나의 에러일지라도 치명적인 결과를 초래할 수 있다는 것이다. 본 논문에서는 소프트웨어 개발자들이 소프트웨어를 개발하는 과정에서 다음 2가지 사항들을 발생시킬 수 있다고 본다.: (1) 에러를 범하지 않는 완벽한 프로그래머는 없다. 그러므로 하나의 에러를 발견하여 수정할 시, 또한 새로운 에러를 유발시킬 수 있다는 것; 그리고 (2) 프로그램 제작 중에 발생되어 이미 프로그램 속에 내재되어 있는 에러들간의 유형들이 모두 같다고 보지 않는 것; (2)를 부연해서 설명하자면 각각의 에러들이 미치는 파급효과들이 서로 다를 것이며, 나아가 에러들마다 발견, 수정하는데 드는 노력과 방법 및 시간적 소모에 있어서도 많은 차이가날 것이므로 에러들간에도 몇가지 유형으로 구분되어진다는 것이다. 앞서 언급한 이 2가지 성격을 간략히 표현하자면 전자는 불완전 수정(imperfect debugging), 후자는 다중 고장 유형(multiple failure types)이라 불린다.

지금까지 다루어져온 많은 소프트웨어 신뢰도 논문에서는 [4, 5, 7, 10] 위에서 언급해온 2가지 성격 중 하나의 성격을 가정하여 연구되어진 논문들인데 반해, 1994년에 Tom Lynch, Hoang Pham and Way Kuo[9] 들은 앞서 언급한 불완전 수정과 3가지 유형의 에러가 발생하는 것을 동시에 가정하여 신뢰도 모델을 제시하였다. 이 때, 세가지 에러 유형은 에러를 발견하고(detect) 제거하는데 따른 어려움 정도에 따른 것으로써 이 중 첫번째 유형은 치명적 고장(critical errors, type 1)으로서 발견 및 제거가 몹시 어려운 경우이며, 두번째 유형은 중대 고장(major errors, type 2)으로서 발견 및 제거가 어려운 경우이며 끝으로, 세번째 유형은 사소한 고장(minor errors, type 3)이라 하며 이는 발견 및 제거가 비교적 쉬운 경우에 해당된다고 볼 수 있다. 그리고 발견된 고장수는 nonhomogeneous poisson process를 따른다고 가정하였다.

본 논문에서 제시하고자하는 것도 이들과 유사하나 그들이 사용한 exponential mean value function이 아닌 S-shaped mean value function을 사용하여 신뢰도 모델을 제시하려한다. 2절에서는 이 논문의 배경이 되는 설명을, 가장 중요한 모델 생성 및 모수들에 대한 최우추정량을 구하는 과정은 3절에서, 그리고 제 4절에서는 예제를 통하여 수치 결과를 보여준다.

2. 배경

t 시간까지 발생된 고장수 $\{N(t),\ t \geq 0\}$가 nonhomogeneous poisson process(이하 NHPP)를 따른다고 가정하는 신뢰도 모델들을 NHPP 모델이라 간주한다. NHPP

모델에서의 가장 중요한점은 어떤 시점까지 발견된 총 누적 고장수의 기대치 혹은 평균치를 나타내는 mean value function (이하 mvf, 주로 $H(t)$ 혹은 $m(t)$로 표시됨)을 결정하는 일이다. 지금까지 연구되어온 NHPP 모델들을 mvf에 의해 크게 2부류로 나눌 수 있으며, 그 중 하나가 지수형(exponential) 과 S형(S-shaped) mvf이다.

지수형에 해당되는 모델로서 Musa's exponential model[2]과 Goel and Okumoto's NHPP[1]를 들 수 있겠고, 또한 초기에 발견되는 에러와 종반 무렵에 발견되는 에러가 미치는 영향등이 다르다는 점을 고려한, 즉, 각 에러당 발견률(dectection rate)을 달리한 수정된 모델(modified exponential model)을 Yamada & Osaki[7]등이 제안하였다.

S형에 속하는 모델로서는 Yamada et al.이 제안한 delayed S-shaped 모델[8]과 Ohba가 제안한 inflection S-shaped 모델[3]들이 있다. 이 밖에도 수요 예측이나 경제성장 및 인구 예측에 실질적으로 많이 사용되어지는 logistic과 Gompertz 모델을 이용하여 프로그램 속에 내재된 에러 수를 예측하는 모델[8]들도 있다.

3. 소프트웨어 신뢰도 모델

이 절에서는 본 논문에서 사용되어지는 기호(notation)들에 대한 정의 및 필요한 기본적인 가정들을 언급하고 이를 바탕으로 신뢰도 모델을 생성하며, 이에 따른 모수들(parameters)의 최우추정량을 유도한다.

3.1 기본 가정 및 기호(Notation) 정의

3.1.1 기호(Notation) 정의

$N(t)$: t시간까지 발견된 총 누적 고장 수를 나타내는 counting process

$N_i(t)$: t시간까지 발견된 i번째 유형의 고장수

$H(t)$: mvf, 즉 $H(t) = E[N(t)]$

$H_i(t)$: i번째 에러 유형의 mvf

a : 시험 직전에 프로그램 속에 내재되어 있을 것으로 예측되는 총 고장 수

p_i : i번째 에러 유형이 차지하는 비율

$h(t)$: 고장률(failure rate) 혹은 intensity function
즉, $h(t) = d[H(t)]/dt$

b : 임의 시점에 각 에러당 발생율(고장률)

b_i : 임의 시점에 i번째 에러 유형에 있어서 각 에러당 발생율(고장률)

$d(t)$: t시점에 각 에러당 발생율
즉, $d(t) = h(t)/[a - H(t)]$

$d_i(t)$: t시점에 i번째 에러 유형의 각 에러당 발생율

β_i : i번째 에러 유형의 에러 debugging시 새로운 에러의 발생률

$n(t)$: 임의의 t시점에 발견된 총 에러수와 프로그램 속에 남아있을 에러수를 합한 수

3.1.2 기본 가정

본 논문에서는 다음 사항을 가정한다.

1. 에러 발생수가 NHPP를 따른다.
2. 발견된 에러를 수정 혹은 제거할 시에 또 다른 새로운 에러를 유발할 수 있다.
3. 어떤 시점에 하나의 에러를 발견할 확률은 그 때까지 프로그램 속에 남아있는 에러 수에 비례한다.
4. 발견된 에러의 수정 혹은 제거시 새로운 에러의 발생 확률은 0과 1 사이의 상수(constant)이다.
5. 에러에는 다음과 같은 3가지 유형이 존재한다.
   유형1 (치명적 고장) : 발견이 매우 어렵고 따라서 제거도 몹시 어려운 에러
   유형2 (중대 고장) : 발견이 어렵고 따라서 제거도 쉽지 않는 에러
   유형3 (경미한 고장) : 발견이 쉽고 따라서 제거도 쉬운 에러
6. $a, b_i(i = 1, 2, 3)$는 상수로 간주한다.

3.2 소프트웨어 신뢰도 모델 생성

위의 가정으로 볼 때, t시간 $(0, t]$ 동안 발생된 누적 에러수(cumulative number of errors)인 $\{N(t), t \geq 0\}$의 분포는 아래와 같고

$$P[N(t)=n]=\frac{[H(t)]^n}{n!}\exp[-H(t)], \quad t \geq 0 \ (n=0,1,2,\ldots)$$

또한, $a = H(\infty)$이므로,

$$\lim_{t\to\infty} P[N(t)=n]=\frac{a^n}{n!}\exp[-a], \quad (n=0,1,2,\ldots)$$

그리고 아래 식들로 표현되어질 수 있다.

$$E[N(t)] = H(t) = \int_0^t h(x)dx$$

$$d(t) = h(t)/[a - H(t)]$$

따라서 $d(t)$와 $H(t)$의 관계는 아래식과 같이 표현되어진다.

$$H(t) = a[1 - \exp(-\int_0^t d(u)\,du)]$$

한편, $\overline{N}(t)$를 임의의 t 시간동안 프로그램 내에 남아있는 에러 수를 나타낸다면 다음 식이 성립됨을 알 수 있다.

$$\begin{aligned} E[\overline{N}(t)] &= a - H(t) \\ &= a \cdot \exp[-\int_0^t d(u)\,du] \end{aligned}$$

여기서 본 논문은 Yamada et al.이 제시한 S-shaped mvf(H(t))를 사용하므로 다음과 같은 형태를 가진다.

$$H(t) = a[1-(1+bt)\exp(bt)], \quad b > 0$$

이 때, $d(t) = b^2 t/(1+bt)$ 임을 알 수 있다.

신뢰도를 정의하고 식을 유도하기 위해 우선 다음과 같은 확률변수를 정의한다.

X_k : $(k-1)$번째 고장으로부터 k번째 고장이 발생된 사이 시간,

S_k : K번째 고장이 발생된 시간, 즉, $\sum_{i=1}^{k} X_i$.

가장 최근에 발생된 마지막($k-1$번째라 가정함) 고장이 t시간에 발생($S_{k-1} = t$)하였다는 조건하에서 x시간 동안 고장이 발생하지 않을 확률을 소프트웨어 신뢰도 $R(x \mid t)$라 정의하고 위에 정의된 확률변수를 이용하여 표현하면 다음과 같다.

$$\begin{aligned} R(x|t) &= \Pr(X_k > x | S_{(k-1)} = t) \\ &= \exp[a(\exp(-\int_0^{t+x} d(u)\,du) - \exp(-\int_0^t d(u)du))] \qquad (1) \\ &= \exp[H(t+x) - H(t)] \end{aligned}$$

신뢰도 모델을 완성하기 위해선 mvf인 $H(t)$를 구하여야 한다. 이는 앞에서 제시된 여러 조건들 즉, 아래 식들을 동시에 만족시키는 해를 구하면 될 것이다.

$$H(t) = \sum_{i=1}^{3} H_i(t)$$

$$h(t) = \frac{d}{dt} H(t) = \sum_{i=1}^{3} h_i(t)$$

$$\frac{d}{dt} H_i(t) = b_i[n_i(t) - H_i(t)]$$

$$\frac{d}{dt} n_i(t) = \beta_i \frac{d}{dt} H_i(t)$$

$$n_i(0) = ap_i$$

$$H_i(0) = 0$$

$$n(t) = \sum_{i=1}^{3} n_i(t)$$

$$n(0) = a$$

이러한 등식들을 동시에 풀이하면 다음과 같은 식을 얻을 수 있다.

$$H_i(t) = \frac{ap_i}{(1-\beta_i)}[1-(1+b_it)^{1-\beta_i}\exp[-(1-\beta_i)b_it]], \tag{2}$$

$$h_i(t) = ap_ib_i(1+b_it)\exp[-(1-\beta_i)b_it], \tag{3}$$

$$n_i(t) = \frac{ap_i}{(1-\beta_i)}[1-\beta_i(1+b_it)^{1-\beta_i}\exp[-(1-\beta_i)b_it]]. \tag{4}$$

(2)식으로 부터 다음식을 유도할 수 있다.

$$\begin{aligned} H(t+x)-H(t) = \sum_{i=1}^{3} \frac{ap_i}{(1-\beta_i)}[&(1+b_it)^{1-\beta_i}\exp[-(1-\beta_i)b_it] \\ &-[1+b_i(t+x)]^{1-\beta_i}\exp[-(1-\beta_i)b_i(t+x)]] \end{aligned} \tag{5}$$

그리고 (5)식을 (1)식에 대입하면 다음과 같은 완전한 신뢰도 모형을 얻을 수 있다.

$$\begin{aligned} R(x|t) = \exp(-[\sum_{i=1}^{3} \frac{ap_i}{(1-\beta_i)}\{&(1+b_it)^{1-\beta_i}\exp[-(1-\beta_i)b_it] \\ &-[1+b_i(t+x)]^{1-\beta_i}\exp[-(1-\beta_i)b_i(t+x)]\}]) \end{aligned} \tag{6}$$

3.3 최우추정량

위에 제시한 모델을 실제 소프트웨어 개발 과정에서 사용하기 위해선 모수 $b_1, b_2, b_3(\overline{b})$와 a에 대한 값을 알아야할 것이다. 보편적으로 관측되어질 수 있는 자료들은 흔히 일정 시간동안 발생된 누적 에러수치이든지 혹은 실제 에러가 발생된 시점에서의 시간을 기록해논 것들인데, 두 경우 모두, 최우 추정법이 주로 많이 사용되어진다. 본 논문에서는 주어진 시간 동안 발생된 누적 에러수를 관측치로한 최우추정량(MLE)을 구하여 본다.

우선 자료 $(t_i, y_{ij}), i=1,2,\ldots,n, j=1,2,3$인 쌍으로된 자료들이 관측되어졌다고 하자. 이 때, t_i시간까지 j번째 타입의 y_{ij}개의 누적 에러수를 나타낸다고 가정하자. 이러한 자료들을 사용한 NHPP의 우도함수 $(L(a, \overline{b}))$는 다음과 같다.

$$L(a, \overline{b}) = P\{H_j(0)=0,\ H_j(t_1)=y_{1j}, \ldots, H_j(t_n)=y_{nj}\}$$

$$= \prod_{j=1}^{3}\prod_{i=1}^{n} \frac{[H_j(t_i)-H_j(t_{i-1})]^{y_{ij}-y_{i-1j}}}{(y_{ij}-y_{i-1j})!} \cdot \exp(-[H_j(t_i)-H_j(t_{i-1})]),$$

where

$$H_j(t_i)=\frac{ap_j}{(1-\beta_j)} \cdot [\ 1-(1+b_jt_i)^{1-\beta_j} \cdot \exp\{-(1-\beta_j)b_jt_i\}],$$

그리고 이 때, $t_0=0, y_0=0$로 간주한다.

양변에 각각 로그를 취하고 모수 $a, b_j(j=1,2,3)$ 각각에 대하여 편미분하여 0을 대입하여 풀면 다음과 같은 식을 얻을 수 있다.

$$\hat{a}=\frac{\sum_{j=1}^{3} y_{nj}}{\sum_{j=1}^{3}\frac{p_j}{1-\beta_j}[1-(1+b_jt_n)^{1-\beta_j} \cdot \exp\{-(1-\beta_j)b_jt_n\}]} \tag{7}$$

$$ap_jt_n(1+b_jt_n)^{-\beta_j} \cdot \exp[-(1-\beta_j)b_jt_n]$$

$$= \sum_{i=1}^{n}(y_{ij}-y_{i-1j}) \cdot (1-\beta_j) \tag{8}$$

$$\frac{t_i(1+b_jt_i)^{-\beta_j}\exp[-(1-\beta_j)b_jt_i]-t_{i-1}(1+b_jt_{i-1})^{-\beta_j}\exp[-(1-\beta_j)b_jt_{i-1}]}{(1+b_jt_{i-1})^{1-\beta_j}\exp[-(1-\beta_j)b_jt_{i-1}]-(1+b_jt_i)^{1-\beta_j}\exp[-(1-\beta_j)b_jt_i]}$$

위의 두 식 중 (8)식에서 b_j들을 a에 의한 함수로 각각 구한 후 (7)식에 대입하여 $a, b_j(j=1,2,3)$들의 추정치 $\hat{a}$와 $\hat{b_j}$들을 각각 구할 수 있다.

4. 예제 적용

아래 <표 1>의 소프트웨어 고장 데이타는 Misra[10] 논문에서 인용한 것으로써 이 자료에 의하면

$p_1 = 0.0173$ $p_2 = 0.3420$ $p_3 = 0.6407$ 이며,

$\beta_1 = 0.5$ $\beta_2 = 0.2$ $\beta_3 = 0.05$ 임을 가정한다.

이 자료에 의해 b_1, b_2, b_3와 a의 최우 추정치를 구해보면 다음과 같은 계산 결과를 얻을 수 있다.

$\hat{b_1} = 0.000311,$ $\hat{b_2} = 0.000359,$ $\hat{b_3} = 0.00038,$ 그리고 $\hat{a} = 436.$

이러한 모수들의 추정치로 부터 <표 2>처럼 신뢰도를 추정할 수 있다.

< 표 1 > 소프트웨어 고장 데이타(주 단위 집계)

주(week)	시험시간	제1종에러	제2종에러	제3종에러	주(week)	시험시간	제1종에러	제2종에러	제3종에러
1	62.5	0	6	9	20	25.0	0	2	3
2	44.0	0	2	4	21	12.0	0	1	1
3	40.0	0	1	7	22	55.0	0	3	2
4	68.0	1	1	6	23	49.0	0	2	4
5	62.0	0	3	5	24	64.0	0	4	5
6	66.0	0	1	3	25	26.0	0	1	0
7	73.0	0	2	2	26	66.0	0	2	2
8	73.0	0	3	5	27	49.0	0	2	0
9	92.0	0	2	4	28	52.0	0	2	2
10	71.4	0	0	2	29	70.0	0	1	3
11	64.5	0	3	4	30	84.5	1	2	6
12	64.7	0	1	7	31	83.0	1	2	3
13	36.0	0	3	0	32	60.0	0	0	1
14	54.0	0	0	5	33	72.5	0	2	1
15	39.5	0	2	3	34	90.0	0	2	4
16	68.0	0	5	3	35	58.0	0	3	3
17	61.0	0	5	3	36	60.0	0	1	2
18	62.6	0	2	4	37	168.0	1	2	11
19	98.7	0	2	10	38	111.5	0	1	9

< 표 2 > 신뢰도($R(x|t)$) 추정치

t	x	신뢰도 추정치 over(t, $t+x$)
120	5	0.7918170
	10	0.6763202
	20	0.3918085
1200	5	0.9653669
	10	0.9301953
	20	0.8327852

참고문헌

[1] A.A. Goel and K. Okumoto(1979), "Time-dependent error detection rate model for software and other performance measures," *IEEE Trans. on Reliability*, Vol. R-28, No. 3.

[2] J.D. Musa, et al.(1987), "Software reliability measurement, prediction, application," *New York: McGraw-Hill International.*

[3] M. Ohba(July 1984), "Software reliability analysis models," *IBM J. Res. Develop.*, Vol. 28, No. 4, pp. 428-443.

[4] N. Kareer, P.K. Kapur, and P.S. Grover(1990), "An S-Shaped software reliability 8growth model with two types of errors," *Microelectronics and Reliability*, Vol. 30, No. 6, pp. 1085-1090.

[5] P.K. Kapur and V.K. Bhalla(1992), "Optimal release policies for a flexible sotfware reliability growth model," *Reliability Engineering and System Safety*, Vol. 35, pp. 45-54.

[6] P.N. Misra(1983), "Software reliability analysis," *IBM Systems Journal*, Vol. 22, No. 5, pp. 262-270.

[7] Yamada and S. Osaki(1985), "Software reliability growth modeling: models and applications," *IEEE Trans. Software Engineering*, Vol. 11, pp. 1431-1437.

[8] S. Yamada, M Ohba and S. Osaki(Dec 1983), "S-shaped reliability growth modeling for software error detection," *IEEE Trans. Rel.*, Vol. R-32, pp. 475-478, 484.

[9] Tom Lynch, Hoang Pham and Way Kuo, "Modeling software-reliability with multiple failure-types and imperfect debugging," 1994 *proceedings Annual Reliability and Mainrainability Symposium*, pp. 235-240.

[10] Way Kuo(1983), "Software reliability estimation: A realization of compaction risk," *Microalactronics and Reliability*, Vol. 23, pp. 235-248.