Journal of the Society of Korea Industrial and Systems Engineering
Vol. 33, No. 2, pp.63－71, June 2010.

오류발생밀도함수를 이용한 현장 적용형 신뢰성 평가모형 개발과 기존 모형과의 비교평가에 관한 연구

김숙희† · 김종훈

동아대학교 컴퓨터공학과

A Study on Comparative Estimate with Development of Reliability Estimation Model in Applicable of Field to Existing Model Using Error Occurrence Density Function

Sukhee Kim† · Jonghun Kim

Division of Computer Engineering, Dong-A University

The existing reliability evaluation models which have already developed by the corporations are so various because of using Maximum Likelihood Method. The existing models are very complicated owing to using system designing methods. Therefore, it is very difficult to utilize the existing models in business fields of many corporations. The purposes of this paper are as follows: The first purpose is to study the simple estimated Parameter to be easily utilized in the business fields of the corporations. The second purpose is to testify the simplification of the developed Parameter of estimated method by comparing the developed reliability evaluation model with the existing reliability evaluation models which are used in the business fields of the corporations.

Keywords : Application Software System, Reliability, Evaluation Model, Parameter

1. 서 론

기업은 처리하고자 하는 업무의 성격에 따라서 데이터를 처리하는 방법이 다르기 때문에 이를 처리하기 위해서 자체적으로 개발팀을 구성하여 소프트웨어 시스템을 개발하여 사용하게 되는데 이것을 응용 소프트웨어 시스템이라고 한다. 따라서 동일한 업무라고 해도 기업에 따라서 요구하는 정보의 처리형태가 달라질 수 있기 때문에 시스템을 분석하고 설계하는 방법이 달라질 수도 있다.

이렇게 독자적으로 응용 소프트웨어 시스템을 개발하기 위해서는 많은 인력과 비용이 필요하게 된다. 그러나 대부분의 기업에서는 이렇게 개발된 응용 소프트웨어 시스템에 대해서 개발팀의 능력만 믿고 현장업무처리에 사용하는 경우가 많다. 결국, 이렇게 처리된 정보는 어느 정도의 신뢰성을 가지고 있는지 모르기 때문에 정보로서의 가치를 갖지 못하는 경우가 발생하게 된다. 따라서 기업이 자체적으로 개발하여 사용하는 응용 소프트웨어 시스템은 반드시 신뢰성을 평가해서 처리된 정보가 어느 정도의 신뢰성을 갖고 있는지를 정보 이용자에게 고지해야 할 것이다. 이렇게 하기 위해서는 개발하는 응용 소프트웨어의 시스템에 대한 신뢰성을 평가하기 위한 모형의 개발이 필요하게 된다[1-2]. 이런 취지에서 보면, 지금까지 연구되어 개발된 소프트웨어 신뢰성 평가모형을 사용할 수도 있겠지만 그러나 소프트

논문접수일 : 2010년 02월 02일 논문수정일 : 2010년 04월 12일 게재확정일 : 2010년 04월 19일
† 교신저자 shkim@kit.ac.kr

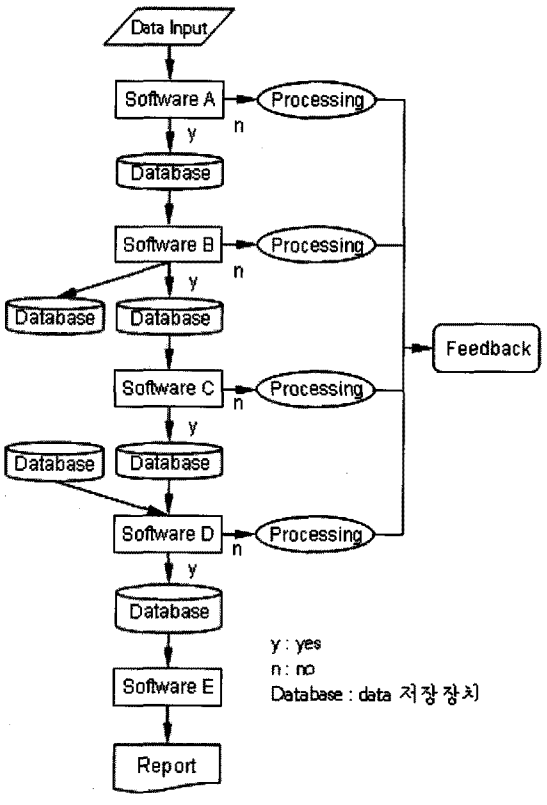
웨어를 개발하는 현장의 입장에서 보면, 짧은 개발기간에 복잡한 모수추정 방법이 적용되고 있는 현재 개발된 모형을 이용한 소프트웨어 시스템의 신뢰성을 평가한다는 것은 현장의 입장에서 보면 그렇게 쉽지 않다[3-4]. 이런 관점에서 보면, 현장에서 개발하여 사용하고 있는 응용 소프트웨어 시스템에 대한 간편한 신뢰성 평가모형이 필요하다고 본다[22].

본 연구에서는 지금까지 개발되어 사용되고 있는 소프트웨어 신뢰성 평가모형의 복잡한 모수추정 방법을 간소화한 새로운 모수추정방법을 개발하고, 이를 이용하여 현장에서 간편하게 신뢰성을 평가해 볼 수 있도록 하고자 한다. 그리고 간소화한 모수추정 방법을 이용한 신뢰성 평가방법과 기존의 개발되어 있는 신뢰성 평가모형을 비교 검토해 봄으로서 개발모형의 간편성을 입증해 보기로 한다.

2. 응용 소프트웨어 시스템의 구성과 오류발생 데이터 조사

2.1 시스템 구성

<그림 1>은 K사가 재고관리를 전산화하기 위해서 개발하는 응용 소프트웨어 시스템의 흐름도이다.



<그림 1> 응용 소프트웨어 시스템의 흐름도

2.2 테스트 단계에서 오류 데이터 조사

<그림 1>과 같은 응용 소프트웨어 시스템의 개발이 완료되면 시스템을 이용하여 현장의 데이터를 처리하기 전에 개발된 소프트웨어 시스템을 종합적으로 테스트하는 과정을 거치게 된다[21].

본 연구에서는 사용하는 기종, 언어, 개발자의 능력 등을 모두 고려한 신뢰성 평가모형을 개발하면서 어떤 기종을 사용했건, 어떤 언어를 사용했건, 개발자의 능력이 어느 정도이건 상관하지 않고 적용할 수 있는 모형을 개발하기 위해서 종합 테스트과정을 거치면서 발생한 테스트회수별 오류발생 수를 조사하여 이 데이터를 이용하여 소프트웨어의 신뢰성을 평가할 수 있는 모형을 개발한다[5-7]. 그래서 소프트웨어 시스템의 테스트 회수별에서 발생한 오류의 수를 변수로 하여 신뢰성 평가모형에 사용할 모수를 추정한다. 테스트 과정에서 발생한 오류의 수를 조사한 결과는 <표 1>과 같다. <표 1>에 조사된 오류의 수는 소프트웨어 시스템을 개발하는 과정에서 발생할 수 있는 모든 오류의 수로서 시스템에 영향을 줄 수 있는 경미한 오류에서부터 치명적인 오류까지 모든 오류가 포함된 수이다.

<표 1> 테스트 회수별 오류의 수

time	errors	time	errors	time	errors	time	errors
1	38	6	29	11	24	16	22
2	34	7	26	12	20	17	19
3	32	8	24	13	23	18	18
4	30	9	25	14	26	19	13
5	32	10	27	15	24	20	14

3. 기존의 이항 모형을 이용한 신뢰성 평가모형 개발

기존 개발되어 있는 이항형 모형을 이용하여 응용 소프트웨어 시스템의 신뢰성을 평가할 수 있는 수식 모형을 개발한다.

3.1 사용된 기호의 정의

사용하는 기호의 정의는 다음과 같다.

f_a : 확률밀도 함수

i : 순서대로 증가하는 수(Index)

m_e : 시간 t_e에서의 오류발생 수

t_e : 오류 데이터 마지막 발생한 시간

t_i : i번째 오류가 발생한 시간

μ_0 : 이항형 모형에서 잠재된 오류발생 수
β_k : Model의 모수
z_a : hazard rate/확률변수인 오류
λ_k : 오류발생 밀도 구성요소
L : 우도함수(Likelihood function)
$\lambda(t)$: 시간 t에서 오류발생 밀도함수; du/dt
b : 최초 정의된 $b_1, b_2, \cdots b_k$의 집합
$M(t_e)$: 시간 t_e에서 발생하는 오류 수의 확률변수
$F_a(t_e)$: t_e의 조건부 이산함수

3.2 이항형을 이용한 수식모형의 개발

먼저 소프트웨어 시스템의 개발과정에서 발생하는 오류발생 수를 이용하여 신뢰성 평가모형을 개발한다. 소프트웨어 시스템을 개발하기 위해서는 개발과정에서 작업 단위별 소프트웨어를 테스트하게 되고, 이 과정에서 오류가 발생하게 되고, 오류를 수정하고, 다시 테스트하는 과정을 반복하게 된다. 따라서 개발하고 있는 소프트웨어 시스템의 테스트 회수와 각 회수에서 발생한 오류의 수를 구할 수 있기 때문에 이를 변수로 사용하게 되면 신뢰성평가 모형에 사용할 수 있는 잔존오류의 수인 μ_0와 모수인 $\hat{\beta}_1$을 추정할 수 있다[8-9]. 여기서 기존의 이항형 모형에서 사용하는 오류발생밀도 함수는 $\lambda_{bin}(t) = \hat{u_0}\hat{\beta}_1 \exp(-\hat{\beta}_1 t)$와 같다.

3.2.1 가정

신뢰성 평가모형을 개발하는데 사용되는 가장 대표적인 형태로서 이 방법으로 신뢰성 평가모형을 개발하기 위해서는 보통 다음에서 정의하는 2가지 가정을 사용하게 된다.

가정 1 : 사용자가 개발한 소프트웨어 시스템에는 반드시 오류가 포함되어 있다고 전재하며, 이 것이 실제 사용자 데이터를 처리하는 과정에 서 오류로 발생하게 된다고 본다.

가정 2 : 오류의 발생은 언제나 랜덤(Random)하 게 발생하며, 위험한 정도에 따라서 발생한다.

3.2.2 수식의 전개

이항형 모형에서 오류발생밀도를 이용한 신뢰성 평가모형을 개발하기 위해서 먼저 발생한 오류의 수와 시행회수가 이항 분포한다고 가정하고, 유한 오류가 발생한다는 조건에서 보면 조건부 최우추정 함수를 식 (1)과 같이 표현할 수 있다.

이 함수를 사용하는 이유는 우도함수를 최대화하는 모수의 값을 찾을 수 있기 때문이다[22].

$$L(\beta_1 | m_e) = \frac{L(\beta_1)}{P(M(t_e) - m_e{}^*\beta_1)} \tag{1}$$

식 (1)의 우도함수와 식 (2)의 우도함수를 이용하여 확률을 구하기 위한 식 (3)을 구할 수 있다.

$$L(\mu_0, \beta_1) = [1 - F_a(t_e)]\mu_0 - m_e \prod_{i=1}^{m_e} (\mu_0 - i + 1) f_a(t_i) \tag{2}$$

$$P[M(t_e) = m_e] \binom{\mu_0}{m_e} [F_a(t_e)]^{m_e} [1 - F_a(t_e)]^{\mu_0 - m_e} \tag{3}$$

여기서 오류발생밀도 함수에 사용할 소프트웨어 시스템에 잔존하고 있는 오류의 수인 μ_0와 모수 $\hat{\beta}_1$을 구하기 위해서 식 (2)과 식 (3)을 이용하여 최우추정방법을 사용한다. 이렇게 하면, 소프트웨어 시스템에 잔존하는 오류의 수 μ_0와 모수 $\hat{\beta}_1$의 추정치를 구하는 식을 구할 수 있는데 바로 식 (4)과 식 (5)을 이용하면 소프트웨어 시스템에 잔존하는 오류의 수 μ_0와 모수 $\hat{\beta}_1$을 구할 수 있는 식 (6)과 식 (7)을 유도할 수 있다[10-12].

$$\frac{\partial \ln L(a, b)}{\partial b} = \ln[1 - F_a(t_e)] + \sum_{i=1}^{m_e} \frac{1}{\mu_0 - i + 1} = 0 \tag{4}$$

$$\frac{\partial \ln L(\mu_0, \beta_1)}{\partial \beta_1} = \frac{\mu_0 - m_e}{1 - F_a(t_e)} \frac{\partial F_a(t_e)}{\partial \beta_1} + \sum_{i=1}^{m_e} \frac{1}{f_a(t_i)} \frac{\partial f_a(t_i)}{\partial \beta_1} = 0 \tag{5}$$

$$-\hat{\beta}_1 * t_e + \sum_{i=1}^{m_e} \frac{1}{\mu_0 - i + 1} = 0 \tag{6}$$

$$-t_e * (\mu_0 - m_e) - \sum_{i=}^{m_e} t_i + \frac{m_e}{\hat{\beta}_1} = 0 \tag{7}$$

이렇게 되면, 식 (6)과 식 (7)을 이용하여 μ_0을 구하는 식을 유도하면 식 (8)이 되며, $\hat{\beta}_1$을 구하는 식을 유도하면 식 (9)과 같이 된다.

$$-\frac{m_e{}^* t_e}{\sum_{i=1}^{m_e} t_i - t_e{}^* (\mu_0 - m_e)} + \sum_{i=1}^{m_e} \frac{1}{\mu_0 - i + 1} = 0 \tag{8}$$

$$\hat{\beta}_1 = \frac{1}{\sum_{i=1}^{m_e} t_i + t_e{}^* (\mu_0 - m_e)} \tag{9}$$

여기서 <표 2>의 측정치를 식 (8)에 대입하여 소프트웨

어 시스템에 잔존하는 오류 수 μ_0을 구한다. 그러나 μ_0와 i값을 동시에 변화시키면서 0이 되는 값을 찾아야 하기 때문에 단순하게 계산해서는 구할 수가 없다[23].

$$\lambda_{bin}(t) = \hat{u_0}\hat{\beta_1}\exp(-\hat{\beta_1}t) \quad (10)$$

그리고 이항형 모형에서 오류발생밀도를 이용한 신뢰성 평가모형의 대표적인 것이 식 (10)이다. 이 식에 사용할 변수는 $\hat{u_0}$와 $\hat{\beta_1}$이며, 이때, 구해야 하는 값은 최우추정방법을 이용하여 구해야 하기 때문에 식 (8)과 식 (9)을 연립으로 풀어야 한다. 이렇게 되면 모수 $\hat{u_0}$와 $\hat{\beta_1}$을 구할 수 있는데 모수를 추정하는 순서는 다음과 같다.

먼저 식 (8)에서 $\hat{u_0}$을 구한다음 이 값을 식 (9)에 대입하여 $\hat{\beta_1}$을 구하면 된다. 이때, $\hat{u_0}$을 구하기 위해서는 실측치와 t_i의 변화를 이용하여 식 (8)의 좌측 값이 0이 되는 $\hat{u_0}$을 찾아야 한다. 따라서 수작업으로는 불가능하기 때문에 컴퓨터를 이용하여 프로그램으로 처리한다.

그리고 $\hat{u_0}$ 값이 구해지면 식 (7)을 이용하여 $\hat{\beta_1}$을 구하면 된다. 이것 역시 컴퓨터를 이용해야 하기 때문에 프로그램이 필요하다. 이를 알고리즘으로 타나내면 다음과 같다.

3.2.3 알고리즘

1단계 : 식 (8)을 이용하여 먼저 $\hat{u_0}$을 구한다.

step 1 : 조사된 m_e, t_e값을 set한다.

step 2 : 초기치 i 값을 1로 set한다.

step 3 : $\hat{u_0}$ 값의 초기치를 0.0001로 설정한다.

step 4 : i을 1씩 증가하여 20이 될 때까지 값을 계산하고 결과가 0이면 $\hat{u_0}$ 값을 읽고 stop한다. 아니면 i을 1로 set하고, $\hat{u_0}$에 0.0001을 증가시키고 step 4를 시행한다. 여기서 $\hat{u_0}$의 값을 0.0001씩 증가한 것은 정확한 $\hat{u_0}$의 값을 찾기 위해서이다.

2단계 : 식 (9)을 이용하여 $\hat{\beta_1}$을 구한다.

4. 간편한 신뢰성 평가모형의 개발

지금까지 개발된 대부분의 소프트웨어 시스템에 대한 신뢰성 평가모형은 최우추정 방법을 사용하여 모수를 추정하고 있다. 이것은 모수의 추정을 보다 정확하게 하여 신뢰성이 높은 모형을 구하기 위해이다. 그러나 대부분의 모형에서 사용하고 있는 모수추정 방법이 너무 복잡하여 응용 소프트웨어를 개발하고 있는 현장에서 사용하기는 어려운 점이 매우 많다[13]. 그것은 기업의 데이터를 처리하기 위해 개발하는 응용 소프트웨어 시스템은 개발기간이 짧고, 개발에 참여하는 인력도 최소의 인력으로 개발 작업을 수행하기 때문에 현장에서 복잡한 신뢰성 평가모형을 적용하여 신뢰성을 평가한다는 것은 거의 불가능하다. 따라서 현장적용이 가능한 간편한 신뢰성 평가모형이 필요하다. 그래서 본 연구에서는 간편하게 모수를 추정할 수 있는 대수-회귀 모수추정방법으로 모수를 추정하는 방법을 개발하고자 한다[20].

4.1 대수-회귀방법을 이용한 모형 개발

현장에서 사용하기 편리하게 하기 위해서는 모수의 추정방법이 간편해야 하기 때문에 여기서는 간편하게 모수를 추정할 수 있는 방법을 개발하고자 한다.

4.1.1 사용되는 기호의 정의

사용하는 기호의 정의는 다음과 같다.

μ_0 : 소프트웨어 시스템에 잠재된 총 오류 수

$\hat{u_{lr}}$: 테스트 단계별로 추정된 오류 수

m_e : 시간 t_e에서의 오류발생 수

$\hat{\beta}_1$: 신뢰성 평가 모형의 모수

$\lambda(t)$: 오류발생 밀도함수

x_i : 테스트 회수별 발생 오류 수

y_i : 테스트 회수

$\bar{x}$: 발생 오류의 평균치

$\bar{y}$: 테스트 회수의 평균치

4.1.2 대수-회귀형 오류발생 밀도함수

먼저 대수-회귀모형을 구하기 위해서 회귀분석에서 보면 절편에 해당하는 μ_0와 기울기에 해당하는 $\hat{\beta_1}$을 구해야 한다. 이를 구하기 위해서 기울기에 해당하는 $\hat{\beta_1}$의 값에 상용대수를 취해서 대수-회귀 모수추정방법을 사용하여 모수를 추정한다. 여기서 $\hat{\beta_1}$을 구하기 위해서 식 (11)을 이용하면 된다[14-15].

$$\hat{\beta_1} = \frac{\sum_{i=1}^{m_e}((x_i-\bar{x})*(y_i-\bar{y}))}{\sum_{i=1}^{m_e}(x_i-\bar{x})^2} \quad (11)$$

그리고 μ_0는 식 (12)을 이용하면 구할 수 있다.

$$\mu_0 = \bar{y} - \hat{\beta_1}*\bar{x} \quad (12)$$

이렇게 되면, 최소자승법을 사용하게 되기 때문에 간단하게 신뢰성 평가모형에 사용할 수 있는 모수를 추정할 수 있다. 먼저 최소자승법을 이용하여 모수를 구한 다음 대수-회귀 모수추정방법으로 변형시키기 위해 회귀계수를 이용하여 구한 값에 상용대수를 취한다. 이렇게 했을 때, 오류발생 밀도함수를 $\lambda_{lr}(t)$이라고 한다. 그리고 $\hat{\beta}_1$을 $\hat{\beta}_{lr}$이라고 하면 식 (13)과 식 (14)과 같이 형식으로 표현할 수 있다[16-17].

$$\mu_0 = \bar{y} - \hat{\beta}_{lr} * \bar{x} \tag{13}$$

여기서 $\dfrac{\sum_{i=1}^{m_e}((x_i-\bar{x})*(y_i-\bar{y}))}{\sum_{i=1}^{m_e}(x_i-\bar{x})^2}$을 β_{lr}라고 두면

$$\hat{\beta}_1 = Log(\beta_{lr}) \tag{14}$$

가 된다.

따라서 식 (13)을 이용하여 구한 μ_0와 식 (14)을 이용하여 구한 $\hat{\beta}_1$와 그리고 소프트웨어 시스템에 잠재해 있는 추정한 오류 수인 μ_0을 이용하여 개발하고자 하는 모형인 대수-회귀모형을 구해 보면, 오류발생 밀도함수 $\lambda_{lr}(t)$는 식 (15)과 같이 된다.

$$\lambda_{lr}(t) = \hat{u}_0 \hat{\beta}_1 \exp(-\hat{\beta}_1 t) \tag{15}$$

이제 최소자승법을 이용하여 μ_0와 $\hat{\beta}_1$을 구하면 된다. 식 (11)과 식 (12)을 이용하여 $\hat{\beta}_1$을 구하면 0.068이 된다. 그리고 μ_0을 구하면 22.0이 된다. 이 값을 이용하여 오류발생 밀도함수인 식 (15)에 대입하면 식 (16)과 같이 된다.

$$\lambda_{lr}(t) = 22.0*0.068*\exp(-0.068*t) \tag{16}$$

4.2 대수-회귀모형을 이용한 신뢰성평가

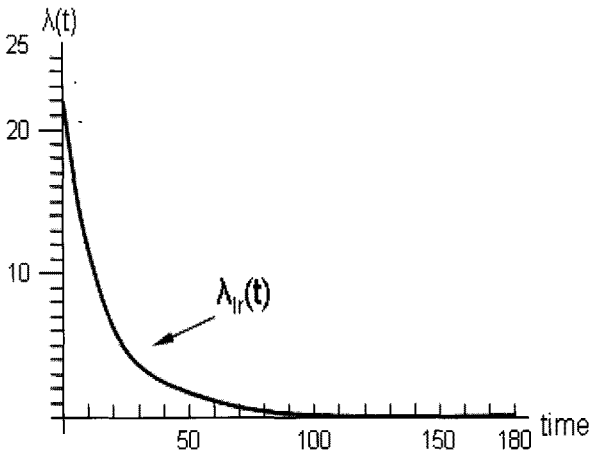
식 (16)을 이용하여 추정치를 구하고, 이를 테스트회수 10회 단위로 정리해 보면 <표 2>와 같다. <표 2>에 알 수 있는 것은 개발하고 있는 소프트웨어 시스템에 잠재한 오류의 수가 22개인데 이를 완전히 제거하고 신뢰도가 1인 시스템을 개발하기 위해서는 총 테스트 회수가 180회가 필요하다는 것을 알 수 있다.

<표 2> 대수-회귀모형에서 오류발생밀도 추정

$time$	$\lambda_{lr}(t)$	$time$	$\lambda_{lr}(t)$	$time$	$\lambda_{lr}(t)$
0	22.0	70	0.2	140	0.01
10	11.0	80	0.1	150	0.009
20	5.7	90	0.1	160	0.006
30	2.9	100	0.09	170	0.003
40	1.5	110	0.07	180	0.001
50	0.8	120	0.05		
60	0.4	130	0.03		

4.3 모형분석

<표 2>의 추정치를 이용하여 오류발생 밀도의 추정치 곡선을 그려보면 <그림 2>와 같이 된다. 여기서 알 수 있는 것은 테스트 횟수가 증가하면서 소프트웨어 시스템 잠재하고 있는 오류가 제거되고 있다는 것을 알 수 있다. 또, 테스트 회수로 볼 때, 120회에서 130회 정도까지는 빠른 속도로 오류가 제거되어 가지만 그 이후에는 완만하게 오류가 제거되고 있다는 것도 알 수 있다. 그리고 오류를 완전히 제거하는 데는 180회 정도의 테스트가 필요하다는 것을 알 수 있다.



<그림 2> 대수-회귀(lr)모형의 추정곡선

5. 기존 이항형 모형을 이용한 신뢰성 평가

5.1 이항형 모형(Bin. Model)

이항형 분포에서 오류발생 밀도함수는 식 (17)과 같다.

$$\lambda_{bin}(t) = \hat{u}_0 \hat{\beta}_1 \exp(-\hat{\beta}_1 t) \tag{17}$$

식 (17)에서 필요한 것은 $\hat{u_0}$와 $\hat{\beta_1}$의 추정치를 구해야 하는데 $\hat{u_0}$와 $\hat{\beta_1}$의 추정치를 구하기 위해서는 먼저 $\hat{u_0}$을 구해야 하는데 $\hat{u_0}$을 구하기 위해서는 식 (18)을 이용해야 한다. 그러나 식 (18)에서 $\hat{u_0}$을 구하기 위해서는 t_e와 m_e 그리고 t_i값을 변화시키면서 좌측의 값이 0이 되는 $\hat{u_0}$값을 찾아야 되기 때문에 매우 복잡한 계산을 반복 시행해야 한다. 따라서 컴퓨터 프로그램을 이용하여 값을 구한다[18-19].

$$-\frac{m_e t_e}{\sum_{i=1}^{m_e} t_i + t_e(\hat{u_0} - m_e)} + \sum_{i=1}^{m_e} \frac{1}{\hat{u_0} - i + 1} = 0 \quad (18)$$

그리고 $\hat{u_0}$값이 구해지면 $\hat{u_0}$값을 식 (18)에 대입하여 $\hat{\beta_1}$을 구한다.

$$\hat{\beta_1} = \frac{m_e}{\sum_{i=1}^{m_e} t_i + t_e(\hat{u_0} - m_e)} \quad (19)$$

여기서는 모수인 $\hat{u_0}$와 $\hat{\beta_1}$을 추정하기 위해서는 먼저 식 (18)에서 $\hat{u_0}$을 구해야 한다. 그러나 $\hat{u_0}$을 구하기 위해서는 컴퓨터 프로그램을 이용하여 시뮬레이션을 하면서 0이 되는 값을 찾아야 한다. 그리고 이렇게 구해진 $\hat{u_0}$을 이용하여 식 (19)을 이용하여 $\hat{\beta_1}$을 구해야 한다.

이렇게 값을 구해 보면, $\hat{u_0}$는 25가 되고, $\hat{\beta_1}$은 0.02794가 된다. 따라서 추정한 모수의 값으로 오류발생 밀도 함수식은 식 (20)과 같이 된다.

$$\lambda_{bin}(t) = 63 * 0.02797 * \exp(-0.02797 * t) \quad (20)$$

5.2 이항형 모형을 이용한 신뢰성 평가

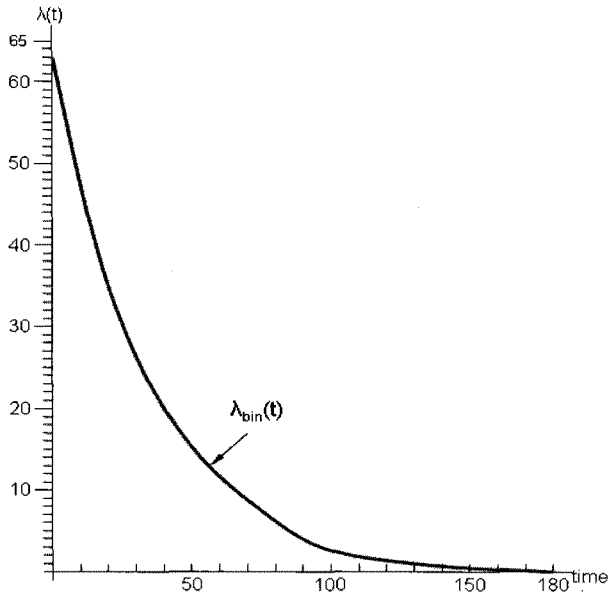
이항형 모형에서 오류발생 밀도 함수의 식 (20)을 이용하여 추정치를 구하면 총 테스트 회수가 180회가 되면 개발하고 있는 소프트웨어 시스템의 신뢰도가 1이 된다는 것을 알 수 있다. <표 3>에서 알 수 있는 바와 같이 테스트 횟수가 증가되면 될수록 오류발생 밀도 함수에서 추정된 오류의 수가 소프트웨어 시스템에 잠재한 오류 수를 제거하여 180회 테스트에서 0이 되는 것을 볼 수 있다. <표 3>은 테스트 회수를 10회 단위로 정리한 것이다.

<표 3> 이항형 모형에서의 오류발생밀도 추정

$time$	$\mu_{bin}(t)$	$time$	$\mu_{bin}(t)$	$time$	$\mu_{bin}(t)$
0	63.0	70	7.8	140	0.9
10	47.4	80	5.8	150	0.7
20	35.1	90	4.3	160	0.5
30	26.0	100	3.1	170	0.3
40	19.3	110	2.3	180	0.1
50	14.3	120	1.7		
60	10.6	130	1.3		

5.3 모형분석

<표 3>의 추정치를 이용하여 추정치 곡선을 그려보면 <그림 3>과 같이 된다. 여기서 알 수 있는 것은 테스트 횟수가 증가하면서 소프트웨어 시스템 잠재하고 있는 오류가 제거되고 있다는 것을 알 수 있다. 또, 테스트 회수로 볼 때, 130회 정도까지는 빠른 속도로 오류가 제거되어 가지만 그 이후에는 완만하게 오류가 제거되고 있다는 것도 알 수 있다. 그리고 오류를 완전히 제거하고, 응용 소프트웨어 시스템이 신뢰도가 1이 되기 위해서는 180회 정도의 테스트가 필요하다는 것을 알 수 있다.



<그림 3> 이항형(bin)모형의 신뢰성 곡선

6. 모형의 비교분석

오류발생밀도 함수를 이용하여 구한 추정치를 비교하

고, 이를 그래프를 이용한 곡선을 비교하여 결과를 분석하고자 한다.

6.1 추정치를 이용한 비교분석

이항형 모형과 대수-회귀모형에서 구한 오류발생 밀도의 추정치를 테스트회수를 10회 단위로 정리하면 <표 4>와 같이 된다. 여기서 알 수 있는 것은 단순하게 추정치를 비교해 보면, 최우추정 방법을 사용하여 모수를 추정해서 사용한 이항형 모형과 최소자승법을 사용해서 모수를 추정해서 사용한 대수-회귀모형을 보면 어떤 모수추정 방법을 사용해도 180회 정도의 테스트에서 신뢰도가 1에 가까워진다는 것을 알 수 있다.

<표 4> 두 모형에서 추정된 밀도의 비교

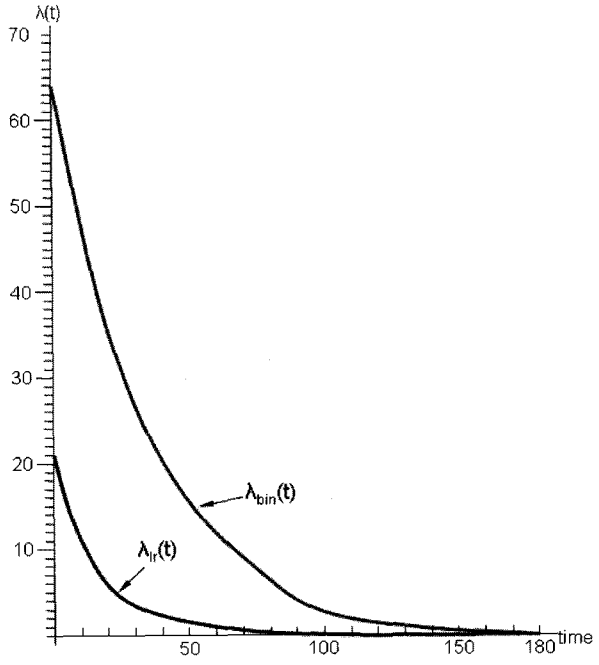
$time$	$\lambda_{lr}(t)$	$\lambda_{bin}(t)$	$time$	$\lambda_{lr}(t)$	$\lambda_{bin}(t)$
0	22.0	63.0	100	0.09	3.1
10	11.0	47.4	110	0.07	2.3
20	5.7	35.1	120	0.05	1.7
30	2.9	26.0	130	0.03	1.3
40	1.5	19.3	140	0.01	0.9
50	0.8	14.3	150	0.009	0.7
60	0.4	.106	160	0.006	0.5
70	0.2	7.8	170	0.003	0.3
80	0.1	5.8	180	0.001	0.1
90	0.1	4.3			

6.2 추정곡선을 이용한 비교분석

이항형 모형과 대수-회귀모형에서 추정한 오류발생 밀도를 곡선으로 비교해 보면 <그림 4>와 같다.

<그림 4>에서 알 수 있는 것은 두 모형에서 추정된 잠재된 오류의 수는 약간의 차이가 있다. 그것은 신뢰성을 평가하는 모형에서 모수를 추정하는 과정에서 발생한 것으로 본다. 따라서 두 모형에서 구한 오류발생 밀도에도 약간의 차이를 발생하고 있다. 그러나 이항형 모형에서 보면 잠재된 오류의 수가 63개이면서 모두 제거하는데 180회의 테스트가 필요하고, 대수-회귀 모형에서 보면 잠재된 오류의 수가 22개이며, 모두 제거하는데 180회의 테스트가 필요하다는 것을 알 수 있다. 따라서 두 모형에서 보면, 응용 소프트웨어 시스템에 잠재한 오류의 수를 완전히 제거하고, 신뢰도가 1이 되는 테스트회수를 보면 180회라는 것을 알 수 있다.

그리고 오류가 제거되어가는 곡선의 모양을 볼 때, 두 곡선이 비슷한 모양을 보고 있는데 이것은 오류를 제거해 가는 방법이 거의 동일하다는 것이다. 따라서 모수를 추정하는 방법에 나타난 문제가 소프트웨어 시스템의 신뢰성을 평가하는데 영향을 주는 것은 아니다.



<그림 4> 두 모형에서의 밀도 함수 비교 곡선

여기서 중요한 것은 두 모형에서 나타난 신뢰도가 1이 되는 시점이며, 그렇게 보면, 두 모형 모두 테스트회수가 180회 정도에서 신뢰도가 1이 된다는 것이다.

따라서 두 모형 중에서 어떤 모형을 이용하여 소프트웨어 시스템의 신뢰성을 평가해도 동일한 결과가 나타난다는 것이다. 이렇게 되면, 본 연구에서 개발한 간편한 대수-회귀형 모수추정 방법으로 구한 모형을 이용하여 소프트웨어 시스템의 신뢰성을 평가해도 기존의 모형을 사용하는 것과 차이가 없다는 것을 알 수 있다.

결국, 간편하게 모수를 추정하여 신뢰성을 평가해 볼 수 있는 대수-회귀 모형은 편리하게 현장에서 사용할 수 있을 것이라고 본다.

7. 결 론

기업은 특정 업무에서 발생하는 데이터를 처리하여 필요한 정보를 얻기 위해서 자체적으로 많은 비용과 인력 그리고 시간을 투입하여 응용 소프트웨어 시스템을 개발하여 사용하고 있다. 따라서 이렇게 자체적으로 개발하여 사용하는 소프트웨어 시스템에 대해서는 자체적으로 검정이 필요하기 때문에 개발한 응용 소프트웨어

시스템의 신뢰성을 평가해 보는 것은 매우 중요하다고 본다. 그 이유는 처리된 정보에 대한 신뢰성을 확보라는 부분도 있지만 무엇보다 많은 인력과 비용 그리고 시간을 투입했기 때문에 반드시 신뢰성 평가는 해야 할 것이다. 만약, 개발된 소프트웨어 시스템의 신뢰성이 어느 정도인지를 알지 못한다면 처리된 정보에 대한 신뢰성을 알지 못하는 것이 되기 때문에 정보를 이용하는 사람들에게 신뢰를 줄 수 없게 되고, 결국에는 신뢰성을 잃게 될 것이다. 그렇게 되면, 사용하지 않는 시스템이 될 것이고, 기업은 많은 손해를 입게 될 것이다.

따라서 반드시 개발된 응용 소프트웨어 시스템의 신뢰성은 평가되어야 하며, 어느 정도의 신뢰성을 가지고 현장의 데이터를 처리하고 있는지를 알아야만 처리된 정보에 대한 이용이 가능해 질 것이다. 그러나 지금까지 연구된 신뢰성 평가모형은 너무 복잡한 모수 추정방법을 사용하기 때문에 응용 소프트웨어를 개발하는 산업현장에서 적용하기에는 어려움이 많이 따르고 있다.

그러나 본 연구에서 개발한 대수-회귀방법을 적용한 모수추정 방법을 사용한다면 간단하게 모수를 추정하고, 이를 이용하여 응용 소프트웨어 시스템에 대한 신뢰성을 평가한다면 간편하게 신뢰성을 평가할 수 있을 것이다. 이렇게 되면 응용 소프트웨어 시스템의 테스트 단계별로 신뢰성을 평가할 수 있어서 시스템을 테스트하는 단계에서 시스템에 대한 신뢰성의 변화과정을 수치를 이용하여 확인할 수 있을 것이다. 이렇게 되면, 현장에서 개발하는 응용 소프트웨어 시스템의 신뢰성을 높이는데 크게 기여할 수 있을 것으로 기대한다.

참고문헌

[1] Abu-Youssef, S. E.; "A Goodness of Fit Approach to Testing Exponential Better than Used (EBU) Life Distributions," *International Journal of Reliability and Applications*, 9(1) : 71-78, 2008.

[2] Abuammoh, A. and Sarhan, A. M.; "Parameters Estimators for the Generalized Exponential Distribution," *International Journal of Reliability and Applications*, 8(1) : 17-25, 2007.

[3] Awad El-Gohary; "Estimation of Parameters in a Generalized Exponential Semi-Markov Reliability Models," *International Journal of Reliability and Applications*, 6(1) : 13-29, 2005.

[4] Francis R. Magee, Jr., and John G. Proakis; "Adaptive Maximum-Likelihood Sequence Estimation for Digital Signaling in the Presence of Inter-symbol Interference," *IEEE Transactions on Information Theory* : 143-148, 1973.

[5] George, J. Schick and Ray W. Wolverton; "An Analysis of Competing Software Reliability Models," *IEEE Transactions on Software Engineering*, Se-4(2) : 104-120, 1978.

[6] John, C. Munson and Boca Raton; "Software specification and design : an engineering approach," *Auerbach Publications*, 120-124, 2006.

[7] John D. Musa; "The Measurement and Management of Software Reliability," *Proceedings of The IEEE*, 68(9), 1131-1143, 1980.

[8] Kazuhira, Okumoto; "A Statistical Method for Software Quality Control," *IEEE Transactions on Software Engineering*, Se-11(12) : 1424-1430, 1985.

[9] Macro, Allen, "Software engineering : Concepts and management," N. Y., Prentice Hall : 210-218, 1990.

[10] Schneidewind, N. F.; "The Use of Simulation in the Evaluation of Software," *IEEE Transactions on Computers* : 47-53, 1977.

[11] Schneidewind, N. F. and Heinz-Michael Hoffmann; "An Experiment in Software Error Data Collection and Analysis," *IEEE Transactions on Software Engineering*, Se-5(3), 276-286, 1979.

[12] Norman, F. Schneidewind; "The State of Software Maintenance," *IEEE Transactions on Software Engineering*, Se-13(3) : 303-310, 1987.

[13] Norman, F. Schneidewind; "Software Maintenance : The Need for Standardization," *Proceedings of The IEEE*, 77(4) : 618-624, 1989.

[14] 佐和隆光, 加納悟; "回歸分析の 實際", 新曜社 : 60-68, 昭和 56.

[15] 岸根卓郎; "理論應用 統計學", 養賢堂 : 247-266, 2006.

[16] 김숙희, 김종훈; "이항분포를 이용한 보안 시스템의 소프트웨어 신뢰성 평가모형 개발에 관한 연구", 정보처리학회논문지, 3(3) : 223-230, 2008.

[17] 김숙희, 김종훈; "지수형분포를 이용한 네트워크 시스템 신뢰성평가에 관한 연구", 정보처리학회논문지, 4(1) : 47-54, 2009.

[18] Susan, L. Gerhart and Lawrence Yelowitz; "Observations of Fallibility in Applications of Modern Programming Methodologies," *IEEE Transactions on Software Engineering*, Se-2(3), 195-285, 1976.

[19] Szymanski, Robert A.; "Computers and Application Software," Columbus, Merrill : 89-94, 1988.

[20] 김숙희, 김종훈; "응용 소프트웨어 시스템의 신뢰성 평가를 위한 간편한 모수추정방법 개발", 한국산학기술학회논문지, 11(2) : 540-549, 2010.

[21] 서진원, 김영태, 공헌택, 임재현, 깁치수; "온톨로지 기반의 소프트웨어 설계에러검출방법", 한국산학기술학회논문지, 10(10) : 2676-2683, 2009.

[22] 전희배, 양해술; "소프트웨어 개발과정의 기술 리뷰 평가방법", 한국산학기술학회논문지, 9(5) : 1234-1241, 2008.

[23] James, M. L., Smith, G. M., and Wolford, J. C.; "Applied Numerical Methods for Digital Computation," Third Edition, Harper and Row, Publishers, New York : 230-256, 1998.

[24] Drake, J. E.; "Discrete Reliability Growth Models Using Failure Discounting," M. S Thesis Naval Postgraduate School, Monterey, CA : 124-143, 1987.