한국산학기술학회논문지
Vol. 11, No. 2, pp. 540-549, 2010

응용 소프트웨어 시스템의 신뢰성 평가를 위한 간편한 모수추정방법 개발

김숙희[1*], 김종훈[1]
[1]동아대학교 컴퓨터공학과

The Development of an easy a simple of Parameter Estimation Method for Reliability Evaluation of Application Software System

Suk-Hee Kim[1*] and Jong-Hun Kim[1]
[1]**Dept. of Computer Engineering, Dong-A University**

요 약 기업이 특정 업무를 처리하기 위해서 자체적으로 개발하여 사용하고 있는 응용 소프트웨어의 신뢰성을 평가 방법은 시스템의 설계방법에 따라서 많은 차이가 있다. 그러나 이 많은 방법들에서 대부분은 모수를 추정하는 방법이 최후추정법을 적용하여 모형을 개발하는 과정을 거치다보니 복잡하여 현장에서 사용하기는 매우 어렵다. 그래서 본 연구에서는 간편한 모수추정방법을 개발하여 현장 적용이 용이한 모형을 개발할 수 있는 간단한 모수추정방법에 대해서 연구하고자 한다. 만약, 이렇게 되면, 현장에서 응용 소프트웨어 시스템을 개발할 때, 신뢰성을 평가해 보고자 하면 바로 사용할 수 있는 모수추정방법이기 때문에 매우 유용하게 사용될 것으로 기대한다. 그리고 이 모수추정방법의 정확성을 검정하기 위해서 기존 모수추정방법으로 신뢰성 평가모형을 구해 비교 평가함으로서 개발된 모수추정법의 간편함을 입증하고자 한다.

Abstract 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.

Key Words : Application software, Reliability, Maximum Likelihood Method, Least square method

1. 서론

정보화 사회에서 가장 중요한 것은 무작위로 발생되고 있는 다량의 원시 데이터를 어떻게 처리하여 자신이 필요로 하는 정보로 전환하느냐가 매우 중요하다. 현재 기업은 발생하는 원시 데이터의 양이 많아지면서 수작업으로 처리하던 방법을 버리고 컴퓨터라는 도구를 사용하여 데이터를 처리하고 있다. 그러나 컴퓨터를 이용한다고 해서 현장에서 발생하는 모든 데이터를 처리할 수 있는 것은 아니다. 여기서 우리는 무작위로 발생하는 다량의 데이터들 중에서 정량적인 데이터만이 컴퓨터 처리의 대상이라는 것을 알아야 한다. 따라서 정성적인 데이터는 컴퓨터로도 처리가 불가능하다. 결국, 정성적인 데이터의 처리는 인간에 의해서만 처리가 가능하다. 그러나 다량의 데이터들 중에서 대부분의 데이터는 정량적인 데이터이기 때문에 이를 처리하기 위해서 현장에서는 자체적으로 이 데이터를 처리하기 위해서 응용 소프트웨어 시스템을 개발하여 사용하고 있

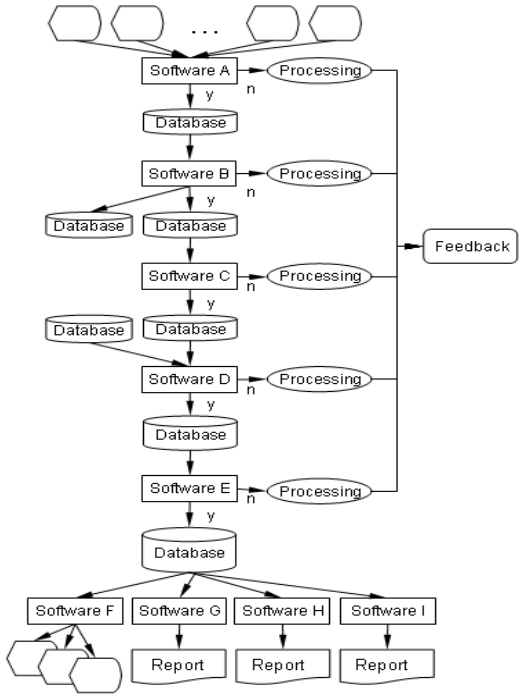
*교신저자 : 김숙희(shkim@kit.ac.kr)
접수일 09년 12월 01일 수정일 (1차 10년 01월 20일, 2차 10년 02월 08일) 게재확정일 10년 02월 24일

다. 이렇게 컴퓨터를 이용하여 데이터를 처리하기 위해 개발하는 응용 소프트웨어 시스템은 처리되는 데이터의 형태나 발생되는 양에 따라서 처리방법이 달라질 수 있기 때문에 규격화된 모듈(Module) 형태의 소프트웨어 시스템으로는 원하는 정보를 얻을 수가 없다. 따라서 사용자가 원하는 여러 가지 조건에 맞는 데이터 처리를 하기 위해서 기업은 여러 종류의 다양한 소프트웨어를 개발하여 사용하게 되는데 이것이 응용 소프트웨어 시스템의 개발이다. 그러나 기업이 자체적으로 개발하여 사용하는 응용 소프트웨어 시스템은 사용자가 요구하는 요구조건을 충족시키면서 데이터를 처리해야 하기 때문에 개발이라는 과정을 거칠 수 밖에 없다. 소프트웨어 시스템의 개발은 프로그래머에 의해서 이루어지는 작업이다. 그러나 프로그래머인 인간이 개발하는 소프트웨어 시스템에는 개발과정에서 인간이 범한 오류가 포함되게 된다. 그래서 우리는 개발한 소프트웨어 시스템이 어느 정도의 신뢰성을 가지고 데이터를 처리할 수 있는지를 확인하기 위해서 개발한 소프트웨어 시스템에 대해서 신뢰성을 평가하게 된다. 우리가 신뢰성을 평가하기 위해서는 기존 개발되어 있는 신뢰성 평가모형을 사용하기도 하고,[1] 새로운 모형을 개발하여 사용하기도 한다. 그러나 어느 쪽으로 신뢰성을 평가하던 신뢰성을 평가하기 위해서는 모형에 사용할 모수를 추정해야 하는데 이 모수의 추정방법이 현재 개발된 모형에서 보면, 매우 복잡하여 소프트웨어 시스템을 개발하는 현장에서 사용하기가 쉽지 않다는 것이 문제이다. 그래서 본 연구에서는 응용 소프트웨어 시스템을 개발하는 현장에서 편리하게 신뢰성을 평가해 볼 수 있는 간편한 모수추정방법에 개발하여 언제 어디서나 편리하게 적용해 볼 수 있는 신뢰성 평가 모형을 개발하고자 한다. 이렇게 되면, 현장에서 응용 소프트웨어 시스템을 개발하면서 신뢰성 평가가 요구되는 시점에 언제든지 개발하고 있는 응용 소프트웨어 시스템이 어느 정도의 신뢰성을 갖고 있는지를 평가해 볼 수 있기 때문에 현장에서 매우 유용하게 사용할 수 있게 될 것이다.

2. 소프트웨어 시스템의 구성과 데이터 조사

2.1 시스템 구성

산업현장에서 응용 소프트웨어 시스템을 개발하여 사용하고 있는 업무는 대부분이 관리업무이기 때문에 여기서는 관리업무의 전산화를 위한 소프트웨어 시스템을 개발하는 과정에서 시스템의 신뢰성을 평가해 보기로 한다. 여기서 일반적인 관리업무라고 하면, 생산관리업무, 품질관리업무, 인사관리업무, 자재관리업무 등을 말한다. 이런 소프트웨어 시스템은 사용자의 편리를 위해서 대부분이 자체적으로 개발하여 사용한다. 그림 1은 K 기업의 생산관리업무 시스템에서 발생하는 데이터를 처리하기 위한 응용 소프트웨어 시스템의 구성을 나타낸 것이다.



[그림 1] 소프트웨어 시스템의 흐름도

2.2 테스트 단계에서 오류 데이터 조사

그림 1과 같은 응용 소프트웨어 시스템은 개발이 완료되면 실제 현장의 데이터를 처리하기 전에 개발된 소프트웨어 시스템의 개발과정에서 문제는 없었는지를 종합적으로 점검하기 위해서 시스템 테스트라는 과정을 거치게 된다. 시스템의 종합 테스트 과정은 개발과정에서 발생한 모든 오류를 점검하여 수정하는 작업을 수행하게 된다. 여기서는 이렇게 종합 테스트과정을 거치면서 발생한 테스트회수별 오류발생 수를 조사하여 이 데이터를 이용하여 소프트웨어의 신뢰성을 평가하는 모형을 개발하고자 한다.[2] 따라서 이 소프트웨어 시스템의 테스트 회수별로 발생한 오류의 수를 조사하여 정리하면 결과는 표 1과 같다. 표 1에 조사된 오류

의 수는 소프트웨어 시스템을 개발하는 과정에서 발생할 수 있는 모든 오류의 수를 조사한 것으로 응용 소프트웨어 시스템에 치명적인 영향을 줄 수 있는 오류에서부터 경미한 오류까지 모든 오류가 포함된 수이다.[3]

[표 1] 테스트회수별 오류 수

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

3. 기존 모수추정방법과 이론적 배경

3.1 사용된 기호의 정의

사용하는 기호의 정의는 다음과 같다.

f_a : 확률밀도 함수

$f(t_i|t_{i-1})$: 누적밀도 함수

i : 순서대로 증가하는 수(index)

m_e : 시간 t_e에서의 오류발생 수

R : 오류 발생 확률

t : 시간

t_e : 오류 데이터가 마지막으로 발생한 시간

t_i : i번째 오류가 발생한 시간

u_0 : 소프트웨어 시스템에 잠재된 오류 수

β_A : β_0을 제외한 β의 집합($\beta_1,\beta_2,...,\beta_n$)

$\lambda(t)$: 오류발생밀도 함수($d\mu/dt$)

T : 확률변수를 정의하는 누적 시간

F_a : 누적 오류발생 확률함수

T_i : 확률변수를 정의하는 i번째 오류발생 시간

β_k : Model의 모수

z_a : hazard rate / 확률변수인 오류

$M(t)$: 시간 t에 의해 표현된 오류발생의 수로 정의된 확률변수

L : Likelihood faction(우도함수)

λ_k : 오류발생 밀도 구성요소

3.2 수학적 모형개발

응용 소프트웨어 시스템의 신뢰성을 평가하는 모형을 개발하고, 이 개발된 모형에 사용할 모수를 추정해야 하는데 이때, 사용할 수 있는 변수는 매우 다양하다. 그러나 본 연구에서는 응용 소프트웨어 시스템의 개발이 완료된 시점에서 마지막으로 수행하는 단계인 종합적 시스템 테스트 단계에서 발생하는 변수를 선택하여 사용하기로 한다. 따라서 이 단계에서 발생하는 변수는 테스트 회수와 회수별로 발생하는 오류의 수이기 때문에 이 변수를 이용하여 모수를 추정한다. 그리고 기존의 모형에서 가장 많이 사용하고 있는 지수분포를 이용하여 신뢰성 평가모형을 구하기 위해서 앞에서 언급한 오류발생의 수가 지수분포한다는 가정을 하고 모수를 추정한다.[4-7] 그리고 이를 이용하여 신뢰성 평가모형을 개발하고, 신뢰성을 평가해 보고자 한다. 지금까지 개발된 응용 소프트웨어 시스템에 적용하고 있는 신뢰성 평가모형의 가장 대표적인 모형이 Jelinski-Moranda Model과 Shooman Model[8]이다. 이들 모형에서 보면, 소프트웨어의 신뢰성을 평가하는 과정에서 발생한 오류가 대체적으로 지수분포한다고 가정하여 모형을 개발하여 사용하고 있으므로 본 연구에서도 지수분포한다고 가정한다. 따라서 여기서는 이들이 개발한 모형을 기존 모형으로 하여 사용하기 위해서 조사된 변수로 모수를 추정하고, 신뢰성을 평가하여 이를 개발할 모형과 비교분석해 보고자 한다.[9] 먼저 평균치를 구하는 수식 (1)에서 오류발생 평균치 함수를 유도해 보기로 한다.

$$\mu(t)=\mu(t;\beta)=\beta_1\mu_0(t;\beta_A) \tag{1}$$

만약, $\beta_1=u_0$이고, 그리고 $\mu_0(t;\beta_A)=F_a(t;\beta_A)$이라고 한다면 t_i와 t_{i-1}사이에서의 함수는 식 (2)과 같이 된다.[10,11]

$$f(t_i|t_{i-1})=-\frac{d}{dt_i}P[T_i>t_i|\,T_{i-1}-t_{i-1}] \tag{2}$$

여기서 다시 식 (3)과 식 (4)을 이용하여 식 (5)을 얻을 수 있다.

$$F_a(t|t_e)=1-\exp[\int_{t_e}^{t}z_a(x)dx] \tag{3}$$

$$P[T_i>t_i|\,T_{i-1}-t_{i-1}] \tag{4}$$

$$=[1-F_a(t_i|t_{i-1})]^{u_0-i+1}$$

$$P[T_i>t_i|\,T_{i-1}-t_{i-1}] \tag{5}$$

$$=\exp[-(u_0-i+1)\int_{t_{i-1}}^{t_i}z_a(x)dx]$$

위의 식을 이용하여 식 (2)과 식 (5)을 이용하여 식 (6)을 얻을 수 있다.

$$f(t_i|t_{i-1}) = (u_0 - i + 1)\, z_a(t_i) \tag{6}$$

$$\exp[-(u_0 - i + 1)\int_{t_{i-1}}^{t_i} z_a(x)dx]$$

여기서 $f_a(t)$와 $F_a(t)$을 이용하면 식 (7)을 얻을 수 있다.

$$f(t_i|t_{i-1}) = \frac{(u_0 - i + 1) f_a(t_i)}{1 - F_a(t_i)} \tag{7}$$

$$[\frac{1 - F_a(t_i)}{1 - F_a(t_{i-1})}]^{u_0 - i + 1}$$

그리고

$$L(\beta) = [\prod_{i=1}^{m_e} f(t_i|t_{i-1})] P[T_{m_e+1} > t_e | T_{m_e} - t_{m_e}]$$

에서 식 (8)을 유도할 수 있다.

$$\prod_{i=1}^{m_e} f(t_i|t_{i-1}) = \tag{8}$$

$$[1 - F_a(t_{m_e})]^{u_0 - m_e} \prod_{i=1}^{m_e} (u_0 - i + 1) f_a(t_i)$$

여기서 두 번째 요소를 계산하기 위해서 식 (9)과 식 (10)을 사용하여 식 (6)을 이용하면 식 (11)을 구할 수 있다.

$$z_a = \frac{f_a(t)}{1 - F_a(t)} \tag{9}$$

$$F_a(t) = 1 - \exp[-\int_0^t z_a(x)dx] \tag{10}$$

$$P[T_{m_e+1} > t_i | T_{m_e} = t_{m_e}] = \tag{11}$$

$$[\frac{1 - F_a(t_e)}{1 - F_a(t_{m_e})}]^{u_0 - m_e}$$

이렇게 구해진 식에 대수를 취하면 다음과 같이 된다.

$$L(u_0, \beta_A) = [1 - F_a(t_e)]^{u_0 - m_e} \tag{12}$$

$$\prod_{i=0}^{m_e} (\mu_0 - i + 1) f_a(t_i)$$

따라서 log-likelihood는 식 (13)과 같이 된다.

$$\ln L(u_0, \beta_A) = (u_0 - m_e) \ln[1 - F_a(t_e)] \tag{13}$$

$$+ \sum_{i=1}^{m_e} \ln(u_0 - i + 1) + \sum_{i=1}^{m_e} \ln f_a(t_i)$$

이렇게 되면 식 (14)과 같은 최우추정식을 유도할 수 있으며, 다시 정리하면 식 (15)을 구할 수 있다.

$$\frac{\partial \ln L(u_0, \beta_A)}{\partial u_0} - \ln[1 - F_a(t_e)] + \tag{14}$$

$$\sum_{i=1}^{m_e} \frac{1}{u_0 - i + 1} = 0$$

$$\frac{\partial \ln L(u_0, \beta_A)}{\partial \beta_k} - \frac{u_0 - m_e}{1 - F_a(t_e)} \frac{\partial F_a(t_e)}{\partial \beta_k} \tag{15}$$

$$+ \sum_{i=1}^{m_e} \frac{1}{f_a(t_i)} \frac{\partial f_a(t_i)}{\partial \beta_k} = 0, k = 1, 2, 3, ..., \omega$$

여기서 $\mu(t) = u_0 F_a(t)$이고, $\lambda(t) = u_0 f_a(t)$이기 때문에 이를 이용하면 식 (16)을 구할 수 있다.

$$-\frac{u_0 - m_e}{u_0 - \mu(t_e)} \frac{\partial u(t_e)}{\partial \beta_k} \tag{16}$$

$$+ \sum_{i=1}^{m_e} \frac{1}{\lambda(t_i)} \frac{\partial \lambda(t_i)}{\partial \beta_k} = 0, k = 1, 2, ..., \omega$$

그리고 식 (11)을 정리하면 식 (17)을 유도할 수 있다.

$$P[M(t_e) = m_e \tag{17}$$

$$= [\begin{matrix} u_0 \\ m_e \end{matrix}] [F_a(t_e)^{m_e} [1 - F_a(t_e)]^{u_0 - m_e}$$

여기서 조건부 최우추정방법을 이용하여 식을 구하면 식 (18)과 같이 된다.

$$L(u_0, \beta_A | m_e) = L(\beta_A | m_e) \tag{18}$$

$$= \frac{m_e! \prod_{i=1}^{m_e} f_a(t_i)}{[F_a(t_e)]^{m_e}}$$

식 (18)에서 $u_0 F_a(t)$을 $\mu(t)$라고 두고, $u_0 f_a(t)$을 $\lambda(t)$라고 하면 식 (19)을 유도할 수 있다.

$$L(\beta_A | m_e) = \frac{m_e! \prod_{i=1}^{m_e} \lambda(t_i)}{[\mu(t_e)]^{m_e}} \tag{19}$$

그러나 변수 u_0는 조건부 최우추정 함수로는 예측할 수가 없다. 여기서 u_0을 예측하기 위해서는 응용 소프트웨어 시스템에서 발생하는 오류의 발생 수를 관측한 관측치인 m_e을 이용하게 되면 구할 수 있다.

$$\mu(t_e; \hat{\beta}) = \widehat{u_0} F_a(t_e; \widehat{\beta_A}) = m_e \tag{20}$$

따라서 식 (20)을 사용하게 되면 u_0의 값을 구할 수 있다. 결국, 조건부 최우추정의 대수함수는 식 (21)과 같이 된다.

$$\ln L(\beta_A | m_e) = \sum_{i=1}^{m_e} \ln \lambda(t_i) - m_e \ln \mu(t_e) \tag{21}$$

그리고 최우추정식을 구해보면 식 (22)과 같다.

$$\sum_{i=1}^{m_e}\frac{1}{\lambda(t_i)}\frac{\partial\lambda(t_i)}{\partial\beta_k}-\frac{m_e}{\mu(t_e)}\frac{\partial\mu(t_e)}{\partial\beta_k} = 0, k=1,2,...,\omega \quad (22)$$

따라서 오류발생 밀도함수 $\lambda(t)$는 식 (23)과 같이 된다.

$$\lambda(t)=\widehat{u_0}\widehat{\beta_1}\exp(-\widehat{\beta_1}t) \quad (23)$$

그리고 본 연구에서 사용하기 위한 오류발생 평균치 함수를 구하기 위해서 $\mu(t)=u_0F_a(t)$을 이용하면 구하고자 하는 오류발생 평균치 함수는 식 (24)과 같이 된다.

$$\mu(t)=\widehat{u_0}[1-\exp(-\widehat{\beta_1}t)] \quad (24)$$

따라서 본 연구에서는 오류발생 평균치 함수를 이용한 모형을 사용하여 응용 소프트웨어 시스템의 신뢰성을 평가하기로 한다. 그리고 식 (24)에 사용할 변수는 $\widehat{u_0}$와 $\widehat{\beta_1}$이며, 이때, 구해야 하는 값은 최우추정치라야 하기 때문에 이를 구하기 위한 식을 유도하기 위해서 식 (14)과 식(15)을 연립으로 풀면 식(25)과 식(26)을 구할 수 있다.

$$-\widehat{\beta_1}t_e+\sum_{i=1}^{m_e}\frac{1}{\widehat{u_0}-i+1}=0 \quad (25)$$

$$-t_e(\widehat{u_0}-m_e)-\sum_{i=1}^{m_e}t_i+\frac{m_e}{\widehat{\beta_1}}=0 \quad (26)$$

위의 식 (25)을 $\widehat{\beta_1}$으로 정리하면 식 (27)과 같이 된다.

$$\widehat{\beta_1}=\frac{m_e}{\sum_{i=1}^{m_e}t_i+t_e(\widehat{u_0}-m_e)} \quad (27)$$

그리고 식 (24)에서 사용할 $\widehat{u_0}$의 최우추정치를 구하기 위해서 식 (25)과 식 (26)을 정리하면 식 (28)과 같이 된다.

$$-\frac{m_e t_e}{\sum_{i=1}^{m_e}t_i+t_e(\widehat{u_0}-m_e)}+\sum_{i=1}^{m_e}\frac{1}{\widehat{u_0}-i+1}=0 \quad (28)$$

식 (24)인 오류발생 평균치 함수를 이용하여 신뢰성을 평가하기 위해서는 이 식 (24)에서 사용할 모수 $\widehat{u_0}$와 $\widehat{\beta_1}$을 구해야 한다. 모수를 추정하는 순서는 다음과 같다. 먼저 식 (28)에서 $\widehat{u_0}$을 구한다음 이 값을 식 (27)에 대입하여 $\widehat{\beta_1}$을 구하면 된다. 이때, $\widehat{u_0}$을 구하기 위해서는 실측치와 t_i의 변화를 이용하여 식 (28)의 좌측 값이 0이 되는 $\widehat{u_0}$을 찾아야 한다. 따라서 수작업으로는 불가능하기 때문에 컴퓨터를 이용하여 프로그램으로 처리한다. 그리고 $\widehat{u_0}$값이 구해지면 식 (27)을 이용하여 $\widehat{\beta_1}$을 구하면 된다. 이것 역시 컴퓨터를 이용해야 하기 때문에 프로그램이 필요하다.

4. 간편한 모수추정방법 개발

지금까지 개발된 대부분의 소프트웨어 시스템에 대한 신뢰성 평가모형은 최우추정법을 사용하여 모수를 추정하고 있다.[12,13] 이것은 모수의 추정을 보다 정확하게 하여 신뢰성이 높은 모형을 구하기 위해이다. 그러나 대부분의 모형에서 사용하고 있는 모수추정방법이 너무 복잡하여 응용 소프트웨어를 개발하고 있는 현장에서 사용하기는 어려운 점이 매우 많다. 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 모수추정법으로 모수를 추정하는 방법을 개발하고자 한다.[14,15]

4.1 사용되는 기호의 정의

사용하는 기호의 정의는 다음과 같다.

μ_0 : 소프트웨어 시스템에 잠재된 총 오류 수

$\widehat{u_{lr}}$: 테스트 단계별로 추정된 오류 수

m_e : 시간 t_e에서의 오류발생 수

$\hat{\beta}_{lr}$: Model의 모수

$\mu_{lr}(t)$: 대수-회귀모형에서의 평균치 함수

x_i : 테스트 회수별 발생 오류 수

y_i : 테스트 회수

$\overline{x}$: 발생 오류의 평균치

$\overline{y}$: 테스트 회수의 평균치

4.2 대수-회귀 모수추정(lr)

먼저 대수-회귀모형을 구하기 위해서 회귀분석에서 보면 절편에 해당하는 μ_0와 기울기에 해당하는 $\widehat{\beta_1}$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 (28)을 이용하면 된다.

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{m_e}((x_i-\bar{x})*(y_i-\bar{y}))}{\sum_{i=1}^{m_e}(x_i-\bar{x})^2} \quad (28)$$

그리고 μ_0는 식 (29)을 이용하면 구할 수 있다.

$$\mu_0 = \bar{y} - \hat{\beta}_1 * \bar{x} \quad (29)$$

이렇게 되면, 최소자승법을 사용하게 되기 때문에 간단하게 신뢰성 평가모형에 사용할 수 있는 모수를 추정할 수 있다. 먼저 최소자승법을 이용하여 모수를 구한다음 대수-회귀 모수추정방법으로 변형시키기 위해 회귀계수를 이용하여 구한 값에 상용대수를 취한다. 이렇게 했을 때, 오류발생 평균치 함수를 $\mu_{lr}(t)$이라고 한다. 그리고 $\hat{\beta}_1$을 $\hat{\beta}_{lr}$이라고 하면 식 (30)과 식 (31)과 같이 형식으로 표현할 수 있다.

$$\mu_{lr} = \bar{y} - \hat{\beta}_{lr} * \bar{x} \quad (30)$$

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

$$\hat{\beta}_{lr} = Log(\beta_1) \quad (31)$$

가 된다.

따라서 식 (30)을 이용하여 구한 μ_{lr}와 식 (31)을 이용하여 구한 $\hat{\beta}_{lr}$와 그리고 소프트웨어 시스템에 잠재해 있는 추정한 오류 수인 μ_0을 이용하여 개발하고자 하는 모형인 대수-회귀모형을 구해 보면, 오류발생 평균치 함수 $\mu_{lr}(t)$는 식 (32)과 같이 된다.[16,17]

$$\mu_{lr}(t) = \mu_0 * [1-\exp(-\hat{\beta}_{lr})*t] \quad (32)$$

이렇게 되면, 모수의 추정치는 수작업으로도 가능하며, 프로그램을 작성하여 구하는 것도 가능하다. 따라서 두 가지 방법 중에서 어느 방법을 사용해도 모수를 추정할 수 있는 최소자승법이기 때문에 어느 방법을 이용하던 $\hat{\mu}_{lr}$와 $\hat{\beta}_{lr}$을 구할 수 있다. 식 (28)과 식 (29)을 이용하여 $\hat{\mu}_{lr}$을 구해 보면 32.0이 된다. 그리고 이 값을 이용하여 $\hat{\beta}_{lr}$을 구하면 0.068이 된다. 따라서 오류발생 평균치 함수식인 식 (32)에 대입하면 식 (33)과 같이 된다.

$$\mu_{lr}(t) = 32.0 * [1-\exp(-0.068)*t] \quad (33)$$

4.3 대수-회귀모형을 이용한 신뢰성평가

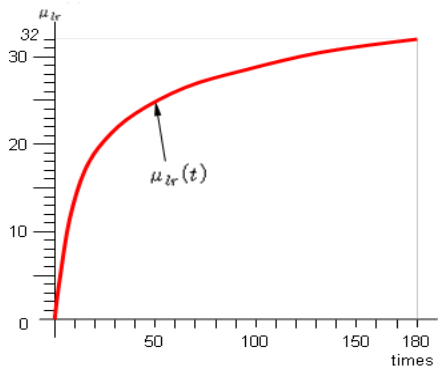
대수-회귀방식을 위한 모수 추정방법을 적용한 식 (33)을 이용하여 추정치를 구하고, 이를 테스트회수 10회 단위로 정리해 보면 표 2와 같다. 표 2에 알 수 있는 것은 개발하고 있는 소프트웨어 시스템에 잠재한 오류의 수가 32인데 이를 완전히 제거하고 신뢰도가 1인 시스템을 개발하기 위해서는 총 테스트 회수가 180회가 필요하다는 것을 알 수 있다.

[표 2] 대수-회귀모형에서 평균치 추정

$time$	$\mu_{lr}(t)$	$time$	$\mu_{lr}(t)$	$time$	$\mu_{lr}(t)$
0	0	70	31.6	140	31.9
10	15.7	80	31.7	150	31.9
20	23.7	90	31.8	160	31.9
30	27.7	100	31.8	170	31.9
40	29.7	110	31.8	180	32.0
50	30.8	120	31.8		
60	31.3	130	31.8		

4.4 모형분석

표 2의 추정치를 이용하여 오류발생 평균치의 추정치 곡선을 그려보면 그림 2와 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.[18,19] 그리고 오류를 완전히 제거하는 데는 180회 정도의 테스트가 필요하다는 것을 알 수 있다.



[그림 2] 대수-회귀(lr)모형의 추정곡선

5. 기존 지수형 모형을 이용한 신뢰성 평가

5.1 사용되는 기호의 정의

사용하는 기호의 정의는 다음과 같다.

m_e : 시간 t_e에서의 오류발생 수

t_e : 오류 데이터 마지막 발생한 시간

t_i : i번째 오류가 발생한 시간

$\mu(t)$: $E(M(t))$

t : 시간

β_1 : Model의 모수

5.2 지수형 모형(exp. Model)

지수형 분포에서 오류발생 평균치 함수는 식 (34)과 같다,

$$\mu_{\exp}(t) = \widehat{u_0}[1-\exp(-\widehat{\beta_1}t)] \quad (34)$$

식 (34)에서 필요한 것은 $\widehat{u_0}$와 $\widehat{\beta_1}$의 추정치를 구해야 하는데 $\widehat{u_0}$와 $\widehat{\beta_1}$의 추정치를 구하기 위해서는 먼저 $\widehat{u_0}$을 구해야 하는데 $\widehat{u_0}$을 구하기 위해서는 식 (35)을 이용해야 한다. 그러나 식 (35)에서 $\widehat{u_0}$을 구하기 위해서는 t_e와 m_e 그리고 t_i값을 변화시키면서 좌측의 값이 0이 되는 $\widehat{u_0}$값을 찾아야 되기 때문에 매우 복잡한 계산을 반복 시행해야 한다. 따라서 컴퓨터 프로그램을 이용하여 값을 구한다.

$$-\frac{m_e t_e}{\sum_{i=1}^{m_e} t_i + t_e(\widehat{u_0} - m_e)} + \sum_{i=1}^{m_e} \frac{1}{\widehat{u_0} - i + 1} = 0 \quad (35)$$

그리고 $\widehat{u_0}$값이 구해지면 $\widehat{u_0}$값을 식 (35)에 대입하여 $\widehat{\beta_1}$을 구한다.

$$\widehat{\beta_1} = \frac{m_e}{\sum_{i=1}^{m_e} t_i + t_e(\widehat{u_0} - m_e)} \quad (36)$$

여기서는 모수인 $\widehat{u_0}$와 $\widehat{\beta_1}$을 추정하기 위해서는 먼저 식 (35)에서 $\widehat{u_0}$을 구해야 한다. 그러나 $\widehat{u_0}$을 구하기 위해서는 컴퓨터 프로그램을 이용하여 시뮬레이션을 하면서 0이 되는 값을 찾아야 한다. 그리고 이렇게 구해진 $\widehat{u_0}$을 이용하여 식 (36)을 이용하여 $\widehat{\beta_1}$을 구해야 한다. 이렇게 값을 구해 보면, $\widehat{u_0}$는 25가 되고, $\widehat{\beta_1}$은 0.02794가 된다. 따라서 추정한 모수의 값을 식 (34)에 대입하면 구하고자 하는 오류발생 평균치 함수 식은 식 (37)과 같이 된다.

$$\mu_{\exp}(t) = 25^*[1-\exp(-0.02794^*t)] \quad (37)$$

5.3 지수형 모형을 이용한 신뢰성 평가

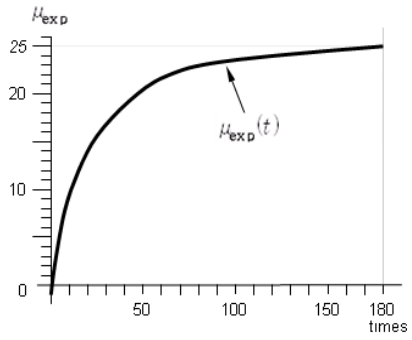
지수형 모형에서 오류발생 평균치 함수의 식 (37)을 이용하여 추정치를 구하면 총 테스트 회수가 180회가 되면 개발하고 있는 소프트웨어 시스템의 신뢰도가 1이 된다는 것을 알 수 있다. 표 3에서 알 수 있는 바와 같이 테스트 횟수가 증가되면 될수록 오류발생 평균치 함수에서 추정된 오류의 수가 소프트웨어 시스템에 잠재한 오류 수에 가까워지면서 180회 테스트에서 25가 되는 것을 볼 수 있다. 표 3은 테스트 회수를 10회 단위로 정리한 것이다.

[표 3] 지수형 모형에서의 평균치 추정

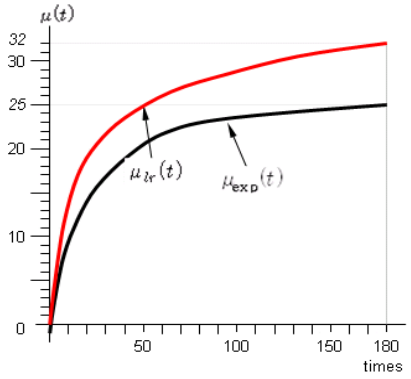
$time$	$\mu_{\exp}(t)$	$time$	$\mu_{\exp}(t)$	$time$	$\mu_{\exp}(t)$
0	0	70	21.5	140	24.5
10	6.10	80	22.3	150	24.6
20	10.7	90	23.0	160	24.7
30	14.2	100	23.5	170	24.8
40	16,8	110	23.9	180	25.0
50	18.8	120	24.1		
60	20.3	130	24.3		

5.4 모형분석

표 3의 추정치를 이용하여 추정치 곡선을 그려보면 그림 3과 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.[20,21] 그리고 오류를 완전히 제거하고, 응용 소프트웨어 시스템이 신뢰도가 1이 되기 위해서는 180회 정도의 테스트가 필요하다는 것을 알 수 있다. 이런 방법을 이용한 연구는 이항분포를 이용한 소프트웨어 신뢰성 평가와 지수형 분포를 이용한 시스템 신뢰성평가에서 이미 입증된 것이다.



[그림 3] 지수형(exp) 모형의 신뢰성 곡선



[그림 4] 두 모형에서의 평균치 함수 비교 곡선

6. 모형의 비교분석

6.1 추정치를 이용한 비교분석

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

[표 4] 두 모형에서 추정된 평균치의 비교

$time$	$\mu_{\exp}(t)$	$\mu_{lr}(t)$	$time$	$\mu_{\exp}(t)$	$\mu_{lr}(t)$
0	0	0	100	23.5	31.8
10	6.1	15.7	110	23.9	31.8
20	10.7	23.7	120	24.1	31.8
30	14.2	27.7	130	24.3	31.8
40	16.8	29.7	140	24.5	31.9
50	18.8	30.8	150	24.6	31.9
60	20.3	31.3	160	24.7	31.9
70	21.5	31.6	170	24.8	31.9
80	22.3	31.7	180	25.0	32.0
90	23.0	31.8			

6.2 추정곡선을 이용한 비교분석

지수형 모형과 대수-회귀모형에서 추정한 오류발생 평균치를 곡선으로 비교해 보면 그림 4와 같다.

그림 4에서 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 평균치에도 약간의 차이를 발생하고 있다. 그러나 기존의 방법인 지수형 모형에서 보면 잠재된 오류의 수가 25개이면서 모두 제거하는데 180회의 테스트가 필요하지만, 개발된 모수추정방법을 이용한 대수-회귀 모형에서 보면 잠재된 오류의 수가 32개이고, 이 잠재된 오류를 모두 제거하는데 180회의 테스트가 필요하다는 것을 알 수 있다. 따라서 두 모형 모두 잠재된 오류를 완전히 제거하고 신뢰도가 1이 되는데 필요한 테스트 회수가 180회라는 것을 알 수 있다. 따라서 간편하게 모수를 추정하여 신뢰성을 구해 볼 수 있는 대수-회귀 방법이 현장 적응성이 높은 소프트웨어 신뢰성 평가방법이 될 것이다. 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. 여기서 중요한 것은 두 모형에서 나타난 신뢰도가 1이 되는 시점이며, 그렇게 보면, 두 모형 모두 테스트 회수가 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-회귀 모수추정방법으로 구한 모형을 이용하여 소프트웨어 시스템의 신뢰성을 평가해도 기존의 최우추정법을 이용하여 구한 모수를 사용한 모형과 신뢰성을 평가하는 데는 차이가 없다는 것을 알 수 있다.[22] 따라서 개발한 모형을 사용하게 되면 현장에서 간편하게 모수를 추정할 수 있고, 응용 소프트

웨어의 신뢰성을 평가해 볼 수 있기 때문에 많은 소프트웨어 시스템을 개발하고 있는 현장에서 편리하게 사용할 수 있을 것이라 본다.

7. 결론

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 개발된 신뢰성 평가모형에서 너무 복잡한 모수 추정방법으로 인해 신뢰성을 평가해 본다는 것이 매우 어렵게 되어 있다. 결국, 지금 사용하고 있는 모수 추정방법은 현장에서 신뢰성을 평가하기 위해 사용하기에는 무리가 있다고 본다. 그러나 본 연구에서 개발한 대수-회귀방법을 적용한 모수추정방법을 사용하여 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. 지금까지의 결과를 보면, 복잡한 과정을 거치면서 모수를 추정하고, 이를 이용하여 신뢰성을 평가하는 기존의 모형보다는 간편하게 모수를 추정하여 신뢰성을 평가할 수 있는 개발모형을 사용하게 된다면 산업현장의 소프트웨어 가술 능력을 향상시키고, 처리된 정보에 대한 신뢰성을 높이는데 크게 기여할 수 잇을 것으로 본다.

참고문헌

[1] 김숙희, 김종훈, "이항분포를 이용한 보안 시스템의 소프트웨어 신뢰성 평가모형 개발에 관한 연구", 정보처리학회논문지, 제3권, 3호. 223-230, 2008,

[2] 서진원, 김영태, 공헌택, 임재현, 김치수, “온톨로지 기반의 소프트웨어 설계에러검출방법”, 한국산학기술학회논문지, Vol. 10, No. 10, 2676-2683, 2009.

[3] 전희배, 양해술, “소프트웨어 개발과정의 기술 리뷰 평가방법”, 한국산학기술학회논문지, Vol. 9, No. 5, 1234-1241, 2008.

[4] 김숙희, 김종훈, "지수형분포를 이용한 네트워크 시스템 신뢰성평가에 관한 연구", 정보처리학회논문지, 제4권, 1호. 47-54, 2009,

[5] Abu-Youssef, S. E, “A Goodness of Fit Approach to Testing Exponential Better than Used (EBU) Life Distributions", International Journal of Reliability and Applications, Vol. 9, No. 1, 71-78, 2008.

[6] Abuammoh, A, Sarhan, A. M, "Parameters Estimators for the Generalized Exponential Distribution", International Journal of Reliability and Applications, Vol. 8, No. 1, 17-25, 2007.

[7] Awad E1-Gohary, "Estimation of Parameters in a Generalized Exponential Semi- Markov Reliability Models", International Journal of Reliability and Applications, Vol. 6, No. 1, 13-29, 2005.

[8] Shooman M. L, "probabilistic Models for Software Reliability Predication and Quality Control", Fall Joint Computer Conference, AFIPS Conference Proceedings, 41, AFIPS Press, Montvale, NJ, 1972.

[9] John D. Musa, "The Measurement and Management of Software Reliability", Proceedings of The IEEE, Vol. 68, No. 9, 1131-1143, 1980.

[10] Kazuhira Okumoto, "A Statistical Method for Software Quality Control", IEEE Transactions on Software Engineering, Vol. Se-11, No. 12, 1424-1430, 1985.

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

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

[13] 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.
[14] 佐和隆光, 加納悟, “回歸分析の 實際”, 新曜社, pp. 60-68, 昭和 56.
[15] 岸根卓郎, “理論應用 統計學”, 養賢堂, pp. 247-266, 2006.
[16] George J. Schick and Ray W. Wolverton, "An Analysis of Competing Software Reliability Models", IEEE Transactions on Software Engineering, Vol. Se-4, No. 2, 104-120, 1978.
[17] John C. Munson. Boca Raton, "Software specification and design: an engineering approach", Auerbach Publications, 120-124, 2006.
[18] N. F. Schneidewind, Heinz-Michael Hoffmann, "An Experiment in Software Error Data Collection and Analysis", IEEE Transactions on Software Engineering, Vol. Se-5, No. 3, 276-286, 1979.
[19] Norman F. Schneidewind, "The State of Software Maintenance", IEEE Transactions on Software Engineering, Vol. Se-13, No. 3, 303-310, 1987.
[20] Norman F. Schneidewind, "Software Maintenance: The Need for Standardization", Proceedings of The IEEE, Vol. 77, No. 4, 618-624, 1989.
[21] Susan L. Gerhart and Lawrence Yelowitz, "Observations of Fallibility in Applications of Modern Programming Methodologies", IEEE Transactions on Software Engineering, Vol. Se-2 No. 3, 195-285, 1976.
[22] Szymanski, Robert A, "Computers and Application Software", Columbus: Merrill, 89-94, 1988.

김 숙 희(Suk-Hee Kim) **[정회원]**

- 1999년 2월 : 동아대학교 컴퓨터공학과(석사)
- 2007년 2월 : 동아대학교 컴퓨터공학과(박사과정수료)
- 1980년 3월 ~ 현재 : 경남정보대학 정보통신센터 팀장

<관심분야>
소프트웨어공학, 홈네트워킹, RFID

김 종 훈(Jong-Hun Kim) **[정회원]**

- 1986년 2월 : 경북대학교 컴퓨터공학과 공학박사
- 2009년 3월 ~ 현재 : 동아대학교 컴퓨터공학과 교수

<관심분야>
암호학분야, RFID 등