한국산학기술학회논문지
Vol. 11, No. 4, pp. 1307-1317, 2010

현장적용이 용이한 간편한 소프트웨어 시스템 신뢰성 평가모형 개발과 기존 모형과의 비교분석에 관한 연구

김숙희[1*], 김종훈[1]
[1]동아대학교 컴퓨터공학과

A Study on Comparative Analysis with Existing Model to Development of Software System Reliability Estimation Model of Field Applicable to be Easy and Simply

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

요 약 응용 소프트웨어 시스템의 신뢰성을 평가하기 위한 모형은 매우 다양하게 개발되어 있다. 그러나 이렇게 개발된 대부분의 모형에서 보면, 모수의 추정방법이 너무 복잡하다. 그러므로 소프트웨어를 생산하는 산업현장에서 적용하기에는 불편한 점이 너무 많다. 그래서 본 연구에서는 간편하게 현장에서 적용할 수 있는 신뢰성 평가모형을 개발한다. 그리고 시스템을 이용하여 신뢰성을 평가하여 기존의 신뢰성 평가모형과 차이가 없음을 증명하고자 한다. 이렇게 효과가 입증되면, 현장에서 소프트웨어 시스템을 개발하고 있는 개발자들이 편리하게 신뢰성을 평가해 볼 수 있도록 하게 될 것이다. 따라서 소프트웨어 시스템에 대한 신뢰도를 높이고, 처리된 정보에 대한 신뢰도를 향상시키는데 크게 기여할 수 있을 것으로 본다.

Abstract The various models that estimate the reliability of application software system had been made. But most of the existing models are inconvenient to industrial fields because so complicated mathematic methods as method of estimation parameter have been used. The two purposes of this paper are to develop the reliability estimation model which was easily applied to industrial fields, and to prove no differences between the existing models and the developed model. Therefore the reliability of software system and handled informations are upgraded by far.

Key Words : Application Software System, Parameter Of Estimation, Software Reliability

1. 서론

기업은 업무의 성격에 따라 데이터를 처리하는 방법이 다르기 때문에 이를 처리하기 위해서 자체적으로 개발팀을 구성하여 소프트웨어를 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 비용 그리고 개발기간이 소요된다. 그러나 대부분의 기업에서는 이렇게 많은 인력과 비용 그리고 개발기간을 투자하여 개발한 시스템에 대해서 시스템의 신뢰성 평가도 거치지 않고 시스템을 개발한 개발팀의 능력만 믿고 현장 업무처리에 사용하는 경우가 대부분이다. 이렇게 개발된 응용 소프트웨어 시스템에 대해서 신뢰성 평가과정을 거치지 않고 현장의 데이터처리에 투입하는 가장 큰 이유는 지금까지 개발되어 있는 응용 소프트웨어 시스템의 신뢰성을 평가모형에 사용하는 모수의 추정방법이 너무 복잡하고, 사용하는 변수가 다양한데 있다고 본다. 그리

*교신저자 : 김숙희(shkim@kit.ac.kr)
접수일 10년 02월 02일 수정일 (1차 10년 03월 31일, 2차 10년 04월 08일) 게재확정일 10년 04월 09일

고 또 다른 이유는 기존의 신뢰성 평가방법을 사용하여 신뢰성을 평가하기 위해서는 많은 비용과 기간 그리고 인력이 소요되기 때문이다.[1,2] 결국, 개발된 응용 소프트웨어 시스템은 신뢰성 평가없이 현장에 투입되어 정보를 처리하게 되다보니 처리된 정보가 어느 정도의 신뢰성을 갖고 있는지를 정보 이용자들은 알 수가 없게 된다. 따라서 이렇게 처리된 정보에 대해서 이를 이용하는 이용자는 신뢰하기가 어려울 것이고, 결국에는 처리된 정보에 대한 불신만 키워 갈 것이다. 따라서 이용자로 하여금 처리된 정보를 신뢰하고 이용할 수 있게 하기 위해서는 현장에서 개발하여 사용하고 있는 응용 소프트웨어 시스템에 대하여 현장에서 간편하게 신뢰성을 평가할 수 있는 모형이 필요하다. 물론, 기업이 자체적으로 업무를 처리하기 위해서 개발하여 사용하는 응용 소프트웨어 시스템이라고 해도 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 현장 적용이 용이한 모형의 개발이 필요하다고 본다.[3,4] 그래서 본 연구에서는 현장에서 개발한 응용 소프트웨어 시스템에 대해서 오류발생 밀도를 이용한 간편한 신뢰성 평가모형을 개발하고[5], 기존 개발되어 있는 오류발생 밀도를 이용한 신뢰성 평가모형과 비교분석해 봄으로서 개발한 모형을 적용하게 되면[6] 기존의 모형보다 얼마나 간편하게 모수를 추정할 수 있고[7], 현장에서 편리하게 사용할 수 있는 지를 보여 줌으로서 산업현장에서 간편하게 신뢰성을 평가해 볼 수 있도록 하는데 목적이 있다.

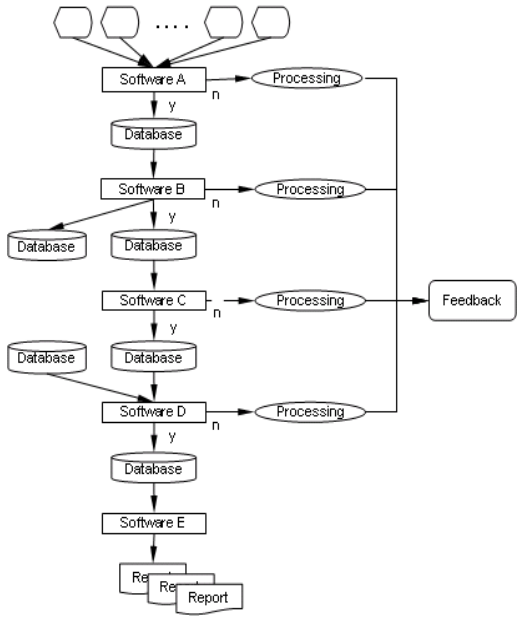
2. 응용 소프트웨어 시스템의 구성과 오류발생 데이터 조사

2.1 시스템 구성

그림 1은 K사가 수작업 중인 재고관리시스템을 전산화하기 위해서 개발하고자 하는 응용 소프트웨어 시스템의 흐름도이다.

2.2 테스트 단계에서 오류 데이터 조사

그림 1과 같은 업무처리 시스템을 수행하기 위해 응용 소프트웨어의 개발이 완료되면 현장의 업무용 데이터를 처리하기 전에 개발된 시스템을 종합적으로 테스트하면서 오류를 제거하면서 신뢰성이 높은 시스템이 되게 하기 위한 작업을 수행하게 된다. 따라서 이 단계에서 테



[그림 1] 응용 소프트웨어 시스템의 흐름도

스트 회수별로 발생하는 오류의 수를 이용하여 응용 소프트웨어 시스템의 신뢰성 평가모형을 개발하여 신뢰성을 평가할 수 있다. 이때, 신뢰성을 평가하기 위해서는 모수의 추정이 필요하게 되는데 본 연구에서는 테스트 과정에서 발생하는 테스트 횟수에 따른 오류발생 수를 변수로 하여 모수를 추정하게 된다. 그리고 이렇게 추정된 모수를 이용하여 신뢰성 평가모형을 개발하고, 시스템의 신뢰성을 평가한다.[8,9] 그래서 응용 소프트웨어 시스템의 테스트과정에서 발생한 테스트 횟수별 오류발생 수를 조사하여 정리한 것이 표 1이다. 따라서 표 1에 조사된 오류의 수는 소프트웨어 시스템을 개발하는 과정에서 발생할 수 있는 모든 오류의 수를 종합적으로 조사한 것이며, 이 오류는 소프트웨어 시스템에 경미한 영향을 줄 수 있는 오류에서부터 시스템에 치명적인 영향을 줄 수 있는 오류까지 모든 오류가 포함된 수이다.[10] 왜냐하면 시스템의 테스트 과정에서 발생하는 오류는 모두 시스템의 신뢰성에 영향을 주기 때문이다.

[표 1] 테스트회수별 오류 수

time	faults	time	faults	time	faults	time	faults
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. 기존 신뢰성 평가모형 개발에 사용한 수학적 배경

기존 개발되어 있는 이항형 모형과 지수형 모형을 이용하기 위해서 응용 소프트웨어 시스템의 신뢰성을 평가할 때 사용한 모형을 수식적으로 전개해 가는 과정을 살펴보기로 한다. 이렇게 기존 모형의 수식 전개과정을 살펴보는 이유는 개발하고자 하는 모형의 수식 전개과정과 비교해 보기 위해서이다.

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 $b_1 b_2, \ldots, b_k$의 집합

3.2 이항형을 이용한 수식모형의 개발

먼저 소프트웨어 시스템의 개발과정에서 발생하는 오류발생 수를 이용하여 신뢰성 평가모형을 개발한다. 소프트웨어 시스템을 개발하기 위해서는 개발과정에서 작업단위별 소프트웨어를 테스트하게 되고, 이 과정에서 오류가 발생하게 되고, 오류를 수정하고, 다시 테스트하는 과정을 반복하게 된다.[11,12] 따라서 개발하고 있는 소프트웨어 시스템의 테스트 회수와 각 회수에서 발생한 오류의 수를 구할 수 있기 때문에 이를 변수로 사용하면 신뢰성 평가모형에 사용할 소프트웨어 시스템에 남아있을 것으로 추정되는 오류 수를 추정한 잔존 오류 수 μ_0와 모수 $\widehat{\beta_1}$을 추정할 수 있다.

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, 이것이 실제 사용자 데이터를 처리하는 과정에서 오류로 발생하게 된다고 본다.

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

3.2.2 수식의 전개

이항형 모형에서 오류발생 밀도를 이용한 신뢰성 평가모형을 개발하기 위해서 먼저 발생한 오류의 수와 시행회수가 이항분포한다고 가정하고[13], 유한 오류가 발생한다는 조건에서 보면 조건부 최우추정 함수를 식 (1)과 같이 표현할 수 있다.[14,15] 이 함수를 사용하는 이유는 우도함수를 최대화하는 모수의 값을 찾을 수 있기 때문이다.

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

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

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

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

여기서 오류발생 밀도함수에 사용할 소프트웨어 시스템에 잔존하고 있는 오류의 수인 μ_0와 모수 $\widehat{\beta_1}$을 구하기 위해서 식 (2)과 식 (3)을 이용하여 최우추정법을 사용한다. 이렇게 하면, 소프트웨어 시스템에 잔존하는 오류의 수 μ_0와 모수 $\widehat{\beta_1}$의 추정치를 구하는 식을 구할 수 있는데 바로 식 (4)과 식 (5)을 이용하면 소프트웨어 시스템에 잔존하는 오류의 수 μ_0와 모수 $\widehat{\beta_1}$을 구할 수 있는 식 (6)과 식 (7)을 유도할 수 있다.

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

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

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

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

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.

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

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

여기서 표 1의 측정치를 식 (8)에 대입하여 소프트웨어 시스템에 잔존하는 오류 수 μ_0을 구한다. 그러나 μ_0와 i값을 동시에 변화시키면서 0(zero)이 되는 값을 찾아야 하기 때문에 단순하게 계산해서는 구할 수가 없다.

$$\lambda_{bin}(t) = \hat{u_0}\hat{\beta}_1 \exp(-\hat{\beta}_1 t) \quad (10)$$

그리고 이항형 모형에서 오류발생 밀도를 이용한 신뢰성 평가모형의 대표적인 것이 식 (10)이다. 이 식에 사용할 변수는 $\hat{u_0}$와 $\hat{\beta}_1$이며, 이때, 구해야 하는 값은 최우추정법을 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 (9)을 이용하여 $\hat{\beta}_1$을 구하면 된다. 이것 역시 컴퓨터를 이용해야 하기 때문에 프로그램이 필요하다.

3.3 지수형을 이용한 수식모형의 개발

3.3.1 가정

가정 1 : 시간 $t = 0$에서 소프트웨어에 존재하는 총 오류의 수는 평균이 지수분포를 한다고 가정한다.

가정 2 : 오류는 서로 독립적으로 발생한다고 본다.

3.3.2 수식의 전개

응용 소프트웨어의 신뢰성을 평가하는 모형을 개발하기 위해서 사용할 수 있는 변수는 매우 다양하다. 그러나 소프트웨어의 테스트 과정에 발생하는 오류의 수를 지수분포한다고 가정하고, 모수를 추정하고, 이를 이용하여 신뢰성을 평가모형을 개발하기 위한 수학적 과정을 보면 다음과 같다.[16,17]

$$\mu(t) = \mu(t;\beta) = \beta_1 \mu_0(t;\beta_A) \quad (11)$$

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 (12)과 같이 된다.

$$f(t_i|t_{i-1}) = -\frac{d}{dt_i}P[T_i > t_i | T_{i-1} - t_{i-1}] \quad (12)$$

여기서 다시 식 (13)과 식 (14)을 이용하여 식 (15)을 얻을 수 있다.

$$F_a(t|t_e) = 1 - \exp[\int_{t_e}^{t} z_a(x)dx] \quad (13)$$

$$P[T_i > t_i | T_{i-1} - t_{u-1}] = [1 - F_a(t_i|t_{i-1})]^{u_0 - i + 1} \quad (14)$$

$$P[T_i > t_i | T_{i-1} - t_{i-1}] = \exp[-(u_0 - i + 1)\int_{t_{i-1}}^{t_i} z_a(x)dx] \quad (15)$$

그리고 식 (12)과 식 (15)을 이용하여 식 (16)을 얻을 수 있다.

$$f(t_i|t_{i-1}) = (u_0 - i + 1)\, z_a(t_i) \exp[-(u_0 - i + 1)\int_{t_{i-1}}^{t_i} z_a(x)dx] \quad (16)$$

여기서 $f_a(t)$와 $F_a(t)$을 이용하면 식 (17)을 얻을 수 있다.

$$f(t_i|t_{i-1}) = \frac{(u_0 - i + 1) f_a(t_i)}{1 - F_a(t_i)} \left[\frac{1 - F_a(t_i)}{1 - F_a(t_{i-1})}\right]^{u_0 - i + 1} \tag{17}$$

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 (18)을 유도할 수 있다.

$$\prod_{i=1}^{m_e} f(t_i|t_{i-1}) = [1 - F_a(t_{m_e})]^{u_0 - m_e} \prod_{i=1}^{m_e} (u_0 - i + 1) f_a(t_i) \tag{18}$$

여기서 두 번째 요소를 계산하기 위해서 식 (19)과 식 (20)을 사용하여 식 (16)을 이용하면 식 (21)을 구할 수 있다.

$$z_a = \frac{f_a(t)}{1 - F_a(t)} \tag{19}$$

$$F_a(t) = 1 - \exp[-\int_0^t z_a(x) dx] \tag{20}$$

$$P[T_{m_e+1} > t_i | T_{m_e} = t_{m_e}] = [\frac{1 - F_a(t_e)}{1 - F_a(t_{m_e})}]^{u_0 - m_e} \tag{21}$$

이렇게 구해진 식에 대수를 취하면 다음과 같이 된다.

$$L(u_0, \beta_A) = [1 - F_a(t_e)]^{u_0 - m_e} \prod_{i=0}^{m_e} (\mu_0 - i + 1) f_a(t_i) \tag{22}$$

따라서 log-likelihood는 식 (23)과 같이 된다.

$$\ln L(u_0, \beta_A) = (u_0 - m_e) \ln[1 - F_a(t_e)] + \sum_{i=1}^{m_e} \ln(u_0 - i + 1) + \sum_{i=1}^{m_e} \ln f_a(t_i) \tag{23}$$

이렇게 되면 식 (24)과 같은 최우추정식을 유도할 수 있으며, 다시 정리하면 식 (25)을 구할 수 있다.

$$\frac{\partial \ln L(u_0, \beta_A)}{\partial u_0} - \ln[1 - F_a(t_e)] + \sum_{i=1}^{m_e} \frac{1}{u_0 - i + 1} = 0 \tag{24}$$

$$\frac{\partial \ln L(u_0, \beta_A)}{\partial \beta_k} - \frac{u_0 - m_e}{1 - F_a(t_e)} \frac{\partial F_a(t_e)}{\partial \beta_k} + \sum_{i=1}^{m_e} \frac{1}{f_a(t_i)} \frac{\partial f_a(t_i)}{\partial \beta_k} = 0, k = 1, 2, .., \omega \tag{25}$$

여기서 $\mu(t) = u_0 F_a(t)$이고, $\lambda(t) = u_0 f_a(t)$이기 때문에 이를 이용하면 식 (26)을 구할 수 있다.

$$-\frac{u_0 - m_e}{u_0 - \mu(t_e)} \frac{\partial u(t_e)}{\partial \beta_k} + \sum_{i=1}^{m_e} \frac{1}{\lambda(t_i)} \frac{\partial \lambda(t_i)}{\partial \beta_k} = 0, k = 1, 2, ..., \omega \tag{26}$$

그리고 식 (21)을 정리하면 식 (27)을 유도할 수 있다.

$$P[M(t_e) = m_e] = [\begin{matrix} u_0 \\ m_e \end{matrix}] [F_a(t_e)^{m_e} [1 - F_a(t_e)]^{u_0 - m_e} \tag{27}$$

여기서 조건부 최우추정방법을 이용하여 식을 구하면 식 (28)과 같이 된다.

$$L(u_0, \beta_A | m_e) = L(\beta_A | m_e) = \frac{m_e! \prod_{i=1}^{m_e} f_a(t_i)}{[F_a(t_e)]^{m_e}} \tag{28}$$

식 (28)에서 $u_0 F_a(t)$을 $\mu(t)$라고 두고, $u_0 f_a(t)$을 $\lambda(t)$라고 하면 식 (29)을 유도할 수 있다.

$$L(\beta_A | m_e) = \frac{m_e! \prod_{i=1}^{m_e} \lambda(t_i)}{[\mu(t_e)]^{m_e}} \tag{29}$$

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

$$\mu(t_e;\hat{\beta}) = \widehat{u_0}F_a(t_e;\widehat{\beta_A}) = m_e \quad (30)$$

따라서 식 (30)을 사용하게 되면 u_0의 값을 구할 수 있다. 결국, 조건부 최우추정의 대수함수는 식 (31)과 같이 된다.

$$\ln L(\beta_A|m_e) = \sum_{i=1}^{m_e} \ln\lambda(t_i) - m_e \ln\mu(t_e) \quad (31)$$

그리고 최우추정식은 식 (32)과 같다.

$$\sum_{i=1}^{m_e} \frac{1}{\lambda(t_i)} \frac{\partial\lambda(t_i)}{\partial\beta_k} - \frac{m_e}{\mu(t_e)} \frac{\partial\mu(t_e)}{\partial\beta_k} = 0, k = 1, 2, ..., \omega \quad (32)$$

따라서 식 (32)에서 오류발생 밀도함수 $\lambda(t)$을 구해보면 식 (33)과 같이 된다.

$$\lambda(t) = \widehat{u_0}\widehat{\beta_1}\exp(-\widehat{\beta_1}t) \quad (33)$$

4. 현장 적용이 용이한 간편한 신뢰성 평가모형의 개발

지금까지 살펴본바와 같이 개발된 대부분의 소프트웨어 시스템에 대한 신뢰성 평가모형은 최우추정법을 사용하여 모수를 추정하고 있다. 이것은 모수의 추정을 보다 정확하게 하여 신뢰성이 높은 모형을 구하기 위해서이다. 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.

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

$\widehat{u_{lr}}$: 테스트 단계별로 추정된 오류 수

m_e : 시간 t_e에서의 오류발생 수

$\hat{\beta}_1$: Model의 모수

$\lambda(t)$: 오류발생 밀도함수

x_i : 테스트 회수별 발생 오류 수

y_i : 테스트 회수

$\overline{x}$: 발생 오류의 평균치

$\overline{y}$: 테스트 회수의 평균치

4.1.2 대수-회귀형 오류발생 밀도함수에 사용할 모수추정방법

먼저 대수-회귀모형을 구하기 위해서 회귀분석에서 보면 절편에 해당하는 μ_0와 기울기에 해당하는 $\widehat{\beta_1}$을 구해야 한다[18]. 이를 구하기 위해서 기울기에 해당하는 $\widehat{\beta_1}$의 값에 상용대수를 취해서 대수-회귀 모수추정방법을 사용하여 모수를 추정한다.[19][20] 여기서 $\widehat{\beta_1}$을 구하기 위해서 식 (34)을 이용하면 된다.

$$\widehat{\beta_1} = \frac{\sum_{i=1}^{m_e}((x_i - \overline{x})*(y_i - \overline{y}))}{\sum_{i=1}^{m_e}(x_i - \overline{x})^2} \quad (34)$$

그리고 μ_0는 식 (35)을 이용하면 구할 수 있다.

$$\mu_0 = \overline{y} - \widehat{\beta_1}*\overline{x} \quad (35)$$

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 밀도함수를 $\lambda_{lr}(t)$이라고 한다. 그리고 $\widehat{\beta_1}$을 $\hat{\beta}_{lr}$이라고 하면 식 (36)과 식 (37)과 같이 형식으로 표현할 수 있다.

$$\mu_0 = \overline{y} - \hat{\beta}_{lr} * \overline{x} \tag{36}$$

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

$$\hat{\beta}_1 = Log(\beta_{lr}) \tag{37}$$

가 된다.

따라서 식 (36)을 이용하여 구한 μ_0와 식 (37)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 (38)과 같이 된다.

$$\lambda_{lr}(t) = \hat{u_0}\hat{\beta}_1 \exp(-\hat{\beta}_1 t) \tag{38}$$

이제 최소자승법을 이용하여 μ_0와 $\hat{\beta}_1$을 구하면 된다. 식 (34)과 식 (35)을 이용하여 $\hat{\beta}_1$을 구하면 0.068이 된다. 그리고 μ_0을 구하면 0.22가 된다. 이 값을 이용하여 오류발생 밀도함수인 식 (38)에 대입하면 식 (39)과 같이 된다.

$$\lambda_{lr}(t) = 0.22*0.068*\exp(-0.068*t) \tag{39}$$

4.2 대수-회귀모형을 이용한 신뢰성평가

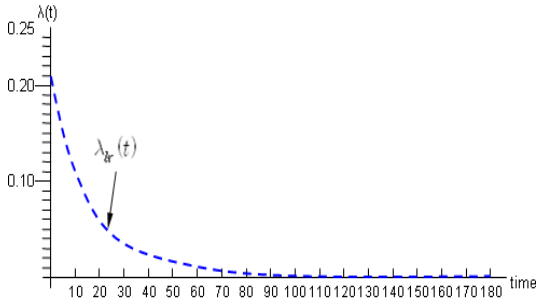
식 (39)을 이용하여 추정치를 구하고, 이를 테스트회수 10회 단위로 정리해 보면 표 2와 같다. 표 2에서 알 수 있는 것은 개발하고 있는 소프트웨어 시스템에 잠재한 오류의 수가 22개인데 이를 완전히 제거하고 신뢰도가 1인 시스템을 개발하기 위해서는 총 테스트 회수가 180회가 필요하다는 것을 알 수 있다.

[표 2] 대수-회귀모형에서 오류발생 밀도 추정

$time$	$\lambda_{lr}(t)$	$time$	$\lambda_{lr}(t)$	$time$	$\lambda_{lr}(t)$
0	0.22	70	0.01	140	0.008
10	0.11	80	0.01	150	0.007
20	0.06	90	0.01	160	0.005
30	0.03	100	0.01	170	0.001
40	0.02	110	0.01	180	0.0
50	0.01	120	0.01		
60	0.01	130	0.008		

4.3 모형분석

표 2의 추정치를 이용하여 오류발생 밀도의 추정치 곡선을 그려보면 그림 2와 같이 된다. 여기서 알 수 있는 것은 테스트 횟수가 증가하면서 소프트웨어 시스템에 잠재하고 있는 오류가 제거되고 있다는 것을 알 수 있다. 또, 테스트 회수로 볼 때, 100회 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 모형을 이용한 신뢰성 평가

기존 모형은 앞에서 언급한 사용자가 가장 많이 사용하고 있는 이항형 모형과 지수형 모형을 사용하여 신뢰성 평가모형을 구하고, 이를 이용하여 오류발생 밀도함수를 구하여 추정치를 구하고, 추정곡선을 구해 보기로 한다.

5.1 이항형 모형(bin. Model)

5.1.1 이항형의 신뢰성 평가모형

이항형 분포에서 오류발생 밀도함수는 식 (40)과 같다.

$$\lambda_{bin}(t) = \hat{u_0}\hat{\beta}_1 \exp(-\hat{\beta}_1 t) \tag{40}$$

식 (40)에서 사용할 모수 $\hat{u_0}$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 (8)을 이용해야 한다. 그러나 식 (8)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 시행해

야 한다.[21,22] 따라서 컴퓨터 프로그램을 이용하여 값을 구한다. 여기서는 모수인 $\widehat{u_0}$와 $\widehat{\beta_1}$을 추정하기 위해서는 먼저 식 (8)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 (9)을 이용하여 $\widehat{\beta_1}$을 구해야 한다. 이렇게 값을 구해 보면, $\widehat{u_0}$는 0.63이 되고, $\widehat{\beta_1}$은 0.02794가 된다. 따라서 추정한 모수의 값으로 오류발생 밀도함수식은 식 (41)과 같이 된다.

$$\lambda_{bin}(t)=0.63 * 0.02794 * \exp(-0.02794 * t) \qquad (41)$$

5.1.2 이항형 모형을 이용한 신뢰성 평가

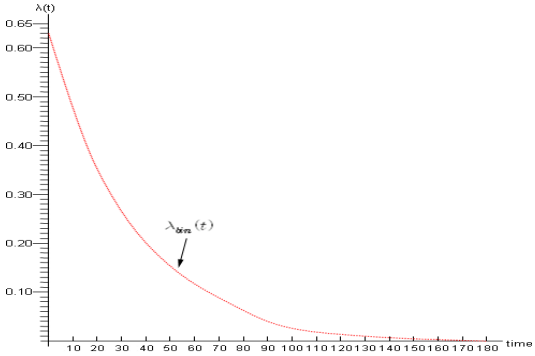
이항형 모형에서 오류발생 밀도함수의 식 (10)을 이용하여 추정치를 구하면 총 테스트 회수가 180회가 되면 개발하고 있는 소프트웨어 시스템의 신뢰도가 1이 된다는 것을 알 수 있다. 표 3에서 알 수 있는 바와 같이 테스트 횟수가 증가되면 될수록 오류발생 밀도함수에서 추정된 오류발생 밀도가 줄어들어 소프트웨어 시스템에 잠재한 오류의 발생밀도가 0에 가깝게 되는데 180회의 테스트에서 0이 되는 것을 알 수 있다. 표 3은 180회의 테스트 과정에서 오류발생 밀도가 줄어들어가는 것을 테스트 회수를 10회 단위로 정리한 것이다.

[표 3] 이항형 모형에서의 오류발생 밀도 추정

$time$	$\mu_{bin}(t)$	$time$	$\mu_{bin}(t)$	$time$	$\mu_{bin}(t)$
0	0.63	70	0.08	140	0.01
10	0.47	80	0.06	150	0.01
20	0.35	90	0.04	160	0.01
30	0.26	100	0.03	170	0.01
40	0.19	110	0.02	180	0.0
50	0.14	120	0.02		
60	0.11	130	0.01		

5.1.3 이산형 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, 130회 정도까지는 빠른 속도로 오류가 제거되어 가지만 그 이후에는 완만하게 오류가 제거되고 있다는 것도 알 수 있다. 그리고 오류를 완전히 제거하여 오류발생 밀도가 0에 가까워지면서 응용 소프트웨어 시스템의 신뢰도가 1에 가까워지기 위해서는 180회 정도의 테스트가 필요하다는 것을 알 수 있다.



[그림 3] 이항형(bin)모형의 신뢰성 곡선

5.2 지수형 모형(exp. Model)

5.2.1 지수형 모형을 이용한 신뢰성 평가

식 (10)이 지수형 모형에서 오류발생 밀도를 이용한 신뢰성을 평가할 수 있는 수식모형이기 때문에 이 함수에 사용할 모수 $\widehat{u_0}$와 $\widehat{\beta_1}$을 구하면 $\widehat{u_0}$는 0.70이며, $\widehat{\beta_1}$은 0.02794이다. 여기서도 모수 $\widehat{u_0}$와 $\widehat{\beta_1}$을 구하기 위해서는 컴퓨터를 이용한 프로그램으로 처리해야 한다. 따라서 지수형의 오류발생 밀도를 구하는 함수는 식 (42)과 같이 된다.

$$\lambda(t) = 0.70*0.02794*\exp(-0.02794*t) \qquad (42)$$

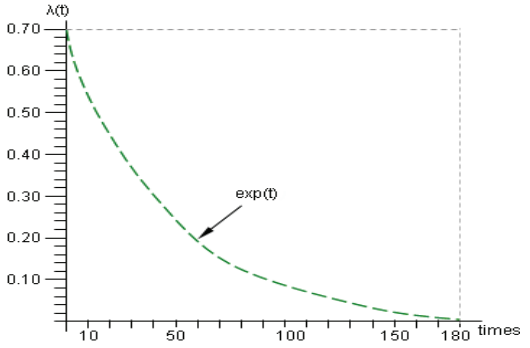
지수형 모형에서 오류발생 밀도함수의 식 (42)을 이용하여 추정치를 구하면 총 테스트 회수가 180회가 되면 오류발생 밀도가 0에 가까워지면서 개발하고 있는 소프트웨어 시스템의 신뢰도가 1이 된다는 것을 알 수 있다. 표 4에서 알 수 있는 바와 같이 테스트 횟수가 증가되면 될수록 오류발생 밀도함수에서 추정된 오류의 수가 소프트웨어 시스템에 잠재한 오류 수를 제거하여 180회 테스트에서 0이 되는 것을 볼 수 있다. 표 4는 테스트 회수를 10회 단위로 정리한 것이다.

[표 4] 지수형 모형에서의 오류발생 밀도 추정

$time$	$\mu_{\exp}(t)$	$time$	$\mu_{\exp}(t)$	$time$	$\mu_{\exp}(t)$
0	0.70	70	0.11	140	0.02
10	0.53	80	0.08	150	0.01
20	0.41	90	0.07	160	0.01
30	0.31	100	0.05	170	0.01
40	0.23	110	0.04	180	0.0
50	0.18	120	0.03		
60	0.13	130	0.02		

2.2 지수형 모형분석

표 4의 추정치를 이용하여 추정치 곡선을 그려보면 그림 4와 같이 된다. 여기서 알 수 있는 것은 테스트 횟수가 증가하면서 소프트웨어 시스템 잠재하고 있는 오류가 제거되면서 오류발생 밀도가 줄어들어가는 것을 알 수 있다.[23] 또, 테스트 회수로 볼 때, 130회 정도까지는 빠른 속도로 오류가 제거되어 가지만 그 이후에는 완만하게 오류가 제거되고 있다는 것도 알 수 있다. 따라서 오류를 완전히 제거하고, 오류발생 밀도가 0이 되는 시점이 응용 소프트웨어 시스템이 신뢰도가 1이 되는 시점이기 때문에 180회 정도의 테스트가 필요하다는 것을 알 수 있다.



[그림 4] 지수형(exp)모형의 신뢰성 곡선

6. 개발모형과의 기존 모형의 비교분석

6.1 오류발생 밀도의 추정치를 이용한 비교분석

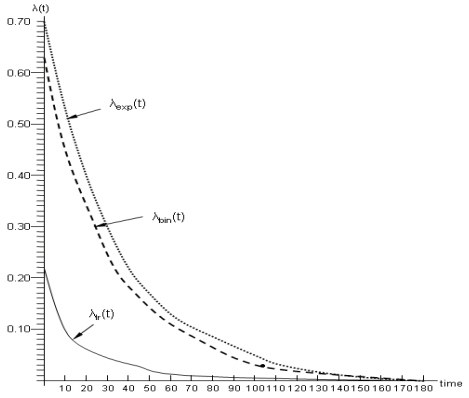
[표 5] 개발모형과 기존 모형에서 추정된 밀도의 비교

$time$	개발 모형	기존 모형		$time$	개발 모형	기존 모형	
	$\lambda_{lr}(t)$	$\mu_{bin}(t)$	$\mu_{\exp}(t)$		$\lambda_{lr}(t)$	$\mu_{bin}(t)$	$\mu_{\exp}(t)$
0	0.22	0.63	0.70	100	0.01	0.03	0.05
10	0.11	0.47	0.53	110	0.01	0.02	0.04
20	0.06	0.35	0.41	120	0.01	0.02	0.03
30	0.03	0.26	0.31	130	0.008	0.01	0.02
40	0.02	0.19	0.23	140	0.008	0.01	0.02
50	0.01	0.14	0.18	150	0.007	0.01	0.01
60	0.01	0.11	0.13	160	0.005	0.01	0.01
70	0.01	0.08	0.11	170	0.001	0.01	0.01
80	0.01	0.06	0.08	180	0.0	0.0	0.0
90	0.01	0.04	0.07				

기존의 이항형 모형과 지수형 모형을 이용한 신뢰성 평가와 개발 모형인 대수-회귀모형에서 구한 오류발생 밀도의 추정치를 테스트회수를 10회 단위로 정리하면 표 5와 같이 된다. 여기서 알 수 있는 것은 단순하게 추정치를 비교해 보면, 복잡한 최우추정법을 사용하여 모수를 추정한 이항형 모형과 지수형 모형의 신뢰성 평가모형과 간단한 최소자승법을 사용해서 모수를 추정해서 사용한 대수-회귀모형의 신뢰성 평가모형을 비교해 보면 어떤 모수추정방법을 사용해도 180회 정도의 테스트에서 신뢰도가 1에 가까워진다는 것을 알 수 있다.[24]

6.2 추정곡선을 이용한 비교분석

기존의 이항형 모형과 지수형 모형과 개발모형인 대수-회귀모형에서 추정한 오류발생 밀도를 곡선을 이용하여 비교해 보면 그림 5와 같다.



[그림 5] 기존 모형과 개발모형의 밀도함수 비교 곡선

그림 5에서 알 수 있는 것은 기존의 이항형 모형과 지수형 모형에서 추정된 잠재된 오류의 수는 약간의 차이가 있다. 그리고 개발 모형인 대수-회귀 모형과 비교해도 잠재된 오류의 수에는 차이가 있다. 그것은 신뢰성을 평가하는 모형에서 모수를 추정하는 과정에서 발생한 것으로 신뢰성을 평가하는 데는 아무런 문제가 되지 않는다고 본다. 이로 인해 3개의 모형에서 구한 오류발생 밀도에도 약간의 차이를 발생하고 있다. 그러나 그림 5를 보면, 복잡한 과정을 거치면서 추정한 모수를 이용하여 추정한 이항형 모형과 지수형 모형이 잠재된 오류를 제거하면서 오류발생 밀도를 0으로 하는데 필요한 테스트횟수가 180회라는 것을 알 수 있다. 그리고 본 연구에서 개발한 대수-회귀 모형에서도 기존의 모형들과 같이 오류발생 밀도를 0으로 하는 데는 180회의 테스트가 필요하다는 것을 알 수 있다. 따라서 3개의 모형에서 보면, 응용 소프트웨어 시스템에 잠재한 오류의 수를 완전히 제거하여 오류발생 밀도를 0으로 하는데 필요한 테스트회수가

동일하다는 것을 알 수 있다. 그리고 오류가 제거되어가는 곡선의 모양을 볼 때, 지수형과 이항형은 비슷한 모양을 보이지만 개발된 대수-회귀모형에서는 약간의 차이를 보이고 있다. 그러나 이것 역시 모수 추정과정에서 발생된 잠재된 오류의 추정치에서 차이가 발생했기 때문이다. 그러나 오류를 제거해 가는 방법은 거의 동일하다는 것을 그림 5에서 알 수 있다. 따라서 모수를 추정하는 방법의 차이는 소프트웨어 시스템의 신뢰성을 평가하는 모형을 결정짓는데 영향을 줄 뿐, 실제 신뢰성을 평가하는 데는 큰 영향을 주지 않는다는 것을 알 수 있다. 여기서 중요한 것은 2개의 기존 모형과 간편하게 신뢰성을 평가해 보기 위해 개발한 모형이 모두 오류발생 밀도가 0이 되는 시점이 180회 정도의 테스트가 이루어진 시점이라는 것이다. 따라서 3개의 모형 중에서 어떤 모형을 이용하여 소프트웨어 시스템의 신뢰성을 평가해도 동일한 결과가 나타난다는 것을 알 수 있다. 결론적으로 본 연구에서 개발한 간편한 대수-회귀형 모수추정방법으로 구한 모형을 이용하여 소프트웨어 시스템의 신뢰성을 평가해도 기존의 모형을 사용하는 것과 차이가 없다는 것을 알 수 있다.[25] 따라서 지금까지 개발된 대부분의 신뢰성 평가모형에서 사용하고 있는 복잡한 모수추정방법을 이용하는 것보다는 간편하게 모수를 추정하여 신뢰성을 평가해 볼 수 있는 대수-회귀 모형을 이용하는 것이 현장 적용성이 매우 높은 신뢰성 평가방법이 된다는 것을 알 수 있다.

6. 결론

기업은 일반적으로 관리업무에서 발생하는 데이터를 전산화하여 처리하기 위해서 자체적으로 많은 비용과 인력 그리고 개발기간을 투입하여 응용 소프트웨어 시스템을 개발하여 사용하고 있다. 따라서 이렇게 자체적으로 개발하여 사용하게 되는 소프트웨어 시스템에 대해서는 시스템의 신뢰성 문제가 대두되기 때문에 개발한 응용 소프트웨어 시스템의 신뢰성을 평가하는 것은 매우 중요하다. 왜냐하면, 관리업무를 수행하기 위해서는 처리된 정보에 대한 신뢰성이 확보되어야만 모든 이용자가 믿고 의사결정의 자료로 사용할 수 있게 될 것이기 때문이다. 만약, 개발된 소프트웨어 시스템에서 처리된 정보가 어느 정도의 신뢰성이 있는지를 알지 못한다면 정보를 이용하는 이용자는 정보를 신뢰할 수 없게 될 것이고, 업무에 활용하기를 꺼려 할 것이다. 따라서 이렇게 개발된 소프트웨어 시스템에 대한 신뢰를 개발자의 개발능력에 의존해서 정보를 처리한다는 것은 매우 위험한 일이라고 본다. 이렇게 되면 많은 비용과 인력 그리고 개발기간을 투자해서 개발한 소프트웨어 시스템이 무용지물이 될 수도 있기 때문이다. 따라서 반드시 개발된 응용 소프트웨어 시스템은 신뢰성을 평가하여야 하며, 평가의 결과, 어느 정도의 신뢰성을 가지고 현장의 데이터를 처리하고 있으며, 처리된 정보가 어느 정도의 신뢰성을 가지고 있는지를 알아야만 정보 이용의 범위를 정할 수 있게 될 것이다. 이렇게 하기 위해서는 현장에서 간편하고, 편리하게 신뢰성을 평가할 수 있는 모형이 필요하게 되는데 지금까지 개발된 신뢰성 평가모형들은 대부분이 너무 복잡한 모수추정방법으로 인해 현장에서 신뢰성을 평가해 본다는 것이 매우 어렵게 되어 있다. 그래서 본 연구에서는 대수-회귀방법을 적용한 모수추정방법을 사용하여 간편하고, 편리하게 모수를 추정하고, 이를 이용하여 응용 소프트웨어 시스템에 대한 신뢰성을 평가할 수 있는 모형을 개발하여 기존 개발되어 있는 2개의 모형과 비교평가를 하였다. 그 결과, 개발한 모형으로 응용 소프트웨어 시스템의 신뢰성을 평가해도 기존의 모형과 크게 차이가 없음을 알 수 있었다. 따라서 개발한 모형을 사용하게 되면, 현장에서 기업의 업무를 처리하기 위해 개발하고 있는 수 많은 응용 소프트웨어 시스템의 신뢰성을 간단하고 편리하게 평가해 볼 수 있을 것으로 보기 때문에 기업의 정보처리 능력을 향상시키고, 동시에 처리된 정보의 신뢰성을 향상시키는데도 크게 기여할 수 있을 것으로 본다.

참고문헌

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

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

[3] Amrit L. Geol, "Software Reliability Models: Assumptions, Limitations and Application", IEEE Transactions on Software Engineering, VOL. SE-11, NO.12, 1465-1467, 1985.

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

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

[6] Robert N. Charette, "Software Engineering Environments", McGraw-Hill, New York, 309-312, 1988.

[7] J. Voas, K. Miller, "Software Testability :The Verification",

IEEE Software, 17-28, 1995.
[8] 서진원, 김영태, 공헌택, 임재현, 깁치수, “온톨로지 기반의 소프트웨어 설계에러검출방법”, 한국산학기술학회논문지, Vol. 10, No. 10, 2676-2683, 2009.
[9] L. Morell, R. Noonan, D. Nicol, B. Murrill, "Estimating the Probability of Failure When Testing Revrals No Failure", IEEE Trans. on Software Engineering, 33-44, Jan. 1992.
[10] 岸根卓郎, “理論應用 統計學”, 養賢堂, pp. 247-266, 2006
[11] 김숙희, 김종훈, "이항분포를 이용한 보안 시스템의 소프트웨어 신뢰성 평가모형 개발에 관한 연구", 정보처리학회논문지, 제3권, 3호. 223-230, 2008, 12.
[12] Sheldon M. Ross, "Introduction to Probability Models", Academic Press, 349-352, 2004.
[13] Aalen, O, Husebye. E., "Statistical Analysis of Repeated Events Forming Renewals Processes", Statistics in Medicine, Vol. 10, 1227-1234. 1991.
[14] Sheldon M. Ross, "Introduction to Probability Models", Academic Press, 349-352, 2004.
[15] R. S. Freedmen, "Testability of Software Components", IEEE Transactions on Software Engineering, 553-564, 1991.
[16] 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.
[17] 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.
[18] 佐和隆光, 加納悟, “回歸分析の 實際”, 新曜社, pp. 60-68, 昭和 56.
[19] 김숙희, 김종훈, “응용 소프트웨어 시스템의 신뢰성 평가를 위한 간편한 모수추정방법 개발”, 한국산학기술학회논문지, Vol. 11, No. 2, 540-549, 2010.
[20] Norman F. Schneidewind, "Software Maintenance: The Need for Standardization", Proceedings of The IEEE, Vol. 77, No. 4, 618-624, 1989.
[21] N. F. Schneidewind, "The Use of Simulation in the Evaluation of Software", IEEE Transactions on Computers, 47-53, 1977.
[22] John C. Munson. Boca Raton, "Software specification and design: an engineering approach", Auerbach Publications, 120-124, 2006.
[23] M. L. James, G. M. Smith, J. C. Wolford., "Applied Numerical Methods for Digital Computation", Third Edition, Harper & 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, September, 124-143, 1987.
[25] 김숙희, 김종훈, "지수형분포를 이용한 네트워크 시스템 신뢰성평가에 관한 연구", 정보처리학회논문지, 제4권, 1호. 47-54, 2009, 1.

김 숙 희(Suk-Hee Kim) **[정회원]**

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

<관심분야>
소프트웨어공학, 홈네트워킹, RFID

김 종 훈(Jong-Hun Kim) **[정회원]**

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

<관심분야>
암호학분야, RFID 등