소프트웨어 개발단계의 신뢰도에 관한 연구

(A Study on the Reliability of S/W during the Developing Stage)

양 계 탁*
(Gye-Tak Yang)

요 약 1972년에 소프트웨어의 신뢰도 문제가 제기되면서부터 개발 중인 소프트웨어의 신뢰도를 평가하고 목표치까지 신뢰도를 성장시키는 방법이 연구되었으며, 테스트 기간 동안에 소요되는 비용 문제까지를 포함하여 적정 인도시기를 결정하는 여러 방법이 제안되었다. 이러한 모델들 중 많은 연구에서 소프트웨어 테스트 전 단계를 거쳐서 테스트 노력이 상수인 것으로 가정하거나 또는 아예 고려하지도 않았으나, 그 후 몇몇 논문을 통하여 테스트 노력을 고려한 소프트웨어의 신뢰도 평가가 중요한 인자인 것으로 발표되었다. 여러 산업 현장의 경험 데이터에 의하면 그 형태가 지수함수형, 레일레이형, 웨이불형, 로지스틱형 테스트 노력 함수 중 하나인 것으로 보고되었다.

따라서, 본 논문에서는 이 네 가지 형태의 테스트 노력을 가진 소프트웨어의 신뢰도 성장에 필요한 각종 파라미터를 구하는 방법에 대하여 제안한다. 개발 현장에서 관찰된 테스트 노력 데이터와 결함검출을 비교하여 어느 형태의 테스트 노력 곡선이 그 경우에 적합한가를 연구하는 한편, 목표 신뢰도에 맞는 발행 시기를 결정하는 문제를 연구한다.

핵심주제어 : SRGM, 평균치 함수, 결함 검출 비, 테스트 노력, 지수함수형, 레일레이형, 웨이불형, 로지스틱형, 목표 신뢰도, 최적 인도 시간

Abstract Many software reliability growth models(SRGM) have been proposed since the software reliability issue was raised in 1972. The technology to estimate and grow the reliability of developing S/W to target value during testing phase were developed using them. Most of these propositions assumed the S/W debugging testing efforts be constant or even did not consider them. A few papers were presented as the software reliability evaluation considering the testing effort was important afterwards. The testing effort forms which have been presented by this kind of papers were exponential, Rayleigh, Weibull, or Logistic functions, and one of these 4 types was used as a testing effort function depending on the S/W developing circumstances.

I propose the methology to evaluate the SRGM using least square estimater and maximum likelihood estimater for those 4 functions, and then examine parameters applying actual data adopted from real field test of developing S/W.

Key Words : SRGM, mea value function, fault detection rate, testing effort, exponentian function, Rayleigh function, Weibull function, logistic function, target reliability, optimum release time

1. 서 론

소프트웨어 신뢰도는 규정된 시간 동안 주어진 환경에서 소프트웨어가 고장 없이 원활하게

* 건양대학교 정보보호학과

동작할 확률[1]로서 소프트웨어의 품질을 평가할 때의 핵심 사항이다. 또한, 소프트웨어 신뢰도는 고객의 입장에서 본 소프트웨어 품질이다. 그러므로 소프트웨어 시스템의 신뢰도를 측정하고 계산 및 성장시키는 것이 매우 중요하며, 개발 기간 동안 모든 테스트 자원을 계획하고 제어하기 위하여 사용될 수 있어서, 우리 모두에게 소프트웨어의 정확성을 정량적으로 확인시켜 주는 것이다. 소프트웨어 신뢰도를 측정하기 위한 공통적인 접근법은 소프트웨어 고장으로부터 구한 가용한 데이터로부터 파라미터를 구하여 선정한 모델에 적용하는 것이다. 한편, 소프트웨어 개발에 있어서 상당한 양의 테스트 자원이 소프트웨어의 테스트에 쓰인다. 이러한 테스트 기간에 대한 테스트 자원의 소요곡선을 테스트 노력 곡선으로 표현할 수 있다. 테스트 노력은 테스트 기간중의 실행 테스트 케이스의 수, 인력의 양, CPU시간 등이다. 소프트웨어 테스트 기간 동안 소프트웨어의 신뢰도는 잠재 소프트웨어 결함을 검출하고 수정하는 데에 소요되는 개발자원의 양에 전적으로 의존한다. Yamada[2]는 역일시간, 테스트 노력량, 테스트 노력에 의하여 검출되는 결함의 수 사이의 관계를 명시적으로 기술하는 새롭고도 단순한 소프트웨어 신뢰도 성장모델을 제안하였다. 이는 테스트 노력의 영향과 관련된 소프트웨어 신뢰도 성장 모델을 제안한 것과 같다. 그들은 시간종속적 지수함수형과 레일레이함수형 곡선을 이용한 테스트 노력 소요량 거동을 기술하였다.

여러 가지 테스트노력을 반영한 소프트웨어의 신뢰도를 산출함에 있어서 이 함수들에 적용되는 파라미터를 구하는 것이 핵심사항이라 할 수 있다. [3]에서는 최대 가능성 함수(Maximum Likelihood Function)를 이용하여 함수를 최대화시키는 방법에 의해서 파라미터를 구하는 기법을 소개하였으며, 이 문헌 발행 전후로 여러 가지 연구 결과가 발표되었다. Syed와 Ram[4]은 로지스틱형 함수와 곰페르츠형 함수를 가진 평균치 함수에 대해서 가능성 함수를 이용하여 비동차포아송과정모델(Non-homogeneous Poisson Process Model)의 각종 데이타 집합의 파라미터를 산출하기 위한 연구를 하였다. Peter[5]는 테스트노력을 고려하지 않은 특정 함수에 대해서 프로그램 내에 처음부터 내재하고 있던 결함의 수를 추론하는 방법을 논하였다. 한편, Tapan[6]은 소프트웨어의 신뢰도를 평가함에 있어서 통계적인 모델을 선정하고 이러한 모델에 대한 파라미터 평가법을 제시하였다. 또한, Yamada[2]는 일반적인 테스트 노력에 대한 파라미터 산출법을 제시하고자 하였으며, Yamada는 또 다른 문헌[7, 8, 9]에서 웨이불 테스트 노력을 적용한 경우의 신뢰도 성장모델을 이용하여 파라미터 산출법을 제시하였다.

따라서, 본 논문에서는 소프트웨어의 신뢰도가 무엇인가를 설명하고 이 신뢰도 성장기법을 설명 및 제안하였다. 각종 테스트노력을 가진 신뢰도 함수를 제안하여 MLE를 이용한 파라미터 추론방법을 구현하며, 신뢰도함수에서 목표 신뢰도에 이르는 발행시각 결정방법을 제안하고, 이러한 방법에 의한 산출결과가 실제 현상과 얼마나 근사한가를 확인하기 위해 실례를 들어서 계산 및 고찰한다. 제 2 장에서는 소프트웨어의 신뢰도에 대한 일반적인 정의 및 의미를 기술하며, 제 3 장에서는 소프트웨어의 신뢰도를 성장시키기 위해 제시된 기법들에 관하여 고찰한다. 제 4 장에서는 각종 테스트 노력 함수에 대해서 고찰한다. 제 5 장에서는 이와 같은 테스트 노력 함수를 고려한 신뢰도를 구함에 있어서 파라미터를 구하는 방법에 대해서 연구하고, 실례로서 각각의 경우에 대하여 파라미터를 구한 후, 테스트 노력곡선과 신뢰도를 산출한다. 제 6 장에서는 결론으로서 본 연구의 결과를 고찰하고 앞으로 연구해야 할 분야를 기술하였다.

2. 소프트웨어의 신뢰도

소프트웨어 신뢰도는 규정된 환경 하에서 주어진 시간에 소프트웨어를 결함 없이 운영할 수 있는 확률인 것으로 정의[10]하며, 다음과 같이 조건확률로 표현할 수 있다.

$$R(x|s) = \Pr\{X_k > x | S_{k-1} = s\} \qquad (2.1)$$

이는 소프트웨어를 개발하여 결함검출 테스트를 시작한 후 계속 s 유니트 시간 동안 $(k-1)$번째 결함을 발견하여 수정한 후 k번 째 결함이 발견되기 전까지 운영될 x 시간동안에 고장 없이 소프트웨어가 동작할 확률인 것이다. 식(2.1)의 시간 간격 X_k에서 소프트웨어의 테스트 노력이 일정하다고 가정하고 테스트 공정이 NHPP를 따른다면 NHPP의 표준 이론으로부터 평균치 함수를

$$m(t) = a(1-e^{-bt}) \tag{2.2}$$

로 정의할 때[11] 임의의 $t \geq 0$과 $x > 0$에서

$$\Pr\{N(t+x)-N(t)=k\} = \frac{[m(t+x)-m(t)]^k}{k!} e^{\{-[m(t+x)-m(t)]\}} \tag{2.3}$$

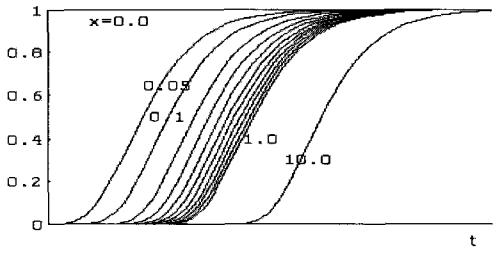
이므로, 테스트 노력이 일정한 경우 식(2.1)의 신뢰도는 다음과 같이 표현할 수 있다.[11]

$$R(x|t) \equiv \Pr\{N(t+x)-N(t)=0\} = e^{[-m(x)e^{-bt}]} \tag{2.4}$$

즉, 평균치 함수의 차이를 지수함수의 지수로 취한 형태를 하고 있다.

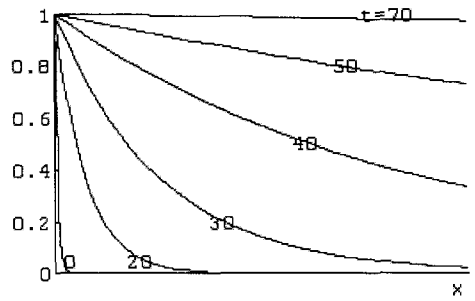
상기 식(2.4)에서 정의한 테스트 단계의 신뢰도 의미를 고찰해보기로 한다. $R(x|t)$는 시각 t에서 최종적으로 결함을 발견하여 완벽하게 수정한 후 $(t+x)$ 시간 동안 새로운 결함이 발견되지 않을 확률이다. 소프트웨어를 개발하여 결함 테스트를 하면 할수록 잔여결함의 수가 줄어들 것이므로 신뢰도가 성장되나, 잔여 겨함이 존재하는 한 결함 수정 후 경과시간이 길어지면 길어질수록 고장 발생 확률이 높아지기 때문에 소프트웨어의 신뢰도는 낮아진다. 단, 테스트 단계에서는 얼마나 오랜 시간 동안 결함이 발견되지 않을 것이냐가 중요한 것이 아니라, 현 단계에서 소프트웨어 내에서 발견되지 않고 잔존하는 결함의 수가 얼마나 되는가가 더 중요하다.

식(2.4)로 표시된 신뢰도의 특성을 이해하기 위해 테스트 시간과 신뢰도의 관계 및 최종 검출 결함 수정 후 경과시간과 신뢰도의 관계를 그림으로 나타내면 <그림 2.1>, <그림 2.2>와 같다.



<그림 2.1> 인도시각과 신뢰도와의 관계

<그림 2.1>의 경우, 최종 결함 수정 후 경과되는 각각의 시간에 대해서 인도시각과 신뢰도 성장과의 관계를 보여주고 있다. 일정한 경과시간 x에 대해서 시험시간 및 인도시기를 늦추면 늦출수록 신뢰도가 성장함을 알 수 있다. 또한, 비록 신뢰도가 성장하여 목표신뢰도 이상이 될 수 있으나, 결함 수정 후 경과시간이 길어지면 길어질수록 결함에 의한 고장 확률이 높아 신뢰도가 저하된다는 것도 알 수 있다. 그림에서 x = 0.0일 때는 시험 시간에 관계없이 신뢰도가 1이나, x의 값이 커지면 커질수록 곡선이 횡축의 우측으로 이동하여 신뢰도가 저하됨을 알 수 있다.



<그림 2.2> 결함 수정후 경과시간과 신뢰도와의 관계

이와 달리 인도 후 경과 시간에 따른 신뢰도의 저하를 보여주는 <그림 2.2>의 경우, 주어진 각각의 테스트 시간 t에 대해서 최종 결함 수정 후 경과시간 x와 신뢰도 성장과의 관계를 보여

주고 있다. 이 그림에서 비록 신뢰도가 성장하여 목표신뢰도 이상이 될 수 있으나, 인도 후 경과시간이 길어지면 길어질수록 결함이 발견될 확률이 높아 신뢰도가 저하된다는 것을 알 수 있다. 일정한 테스트시간 t에 대해서 경과시간 x가 작으면 작을수록 신뢰도가 높으며, x가 증가함에 따라 신뢰도가 급격히 감소된다. 이 그림에서 보듯 테스트 시간이 길면 길수록 경과시간에 대한 신뢰도의 저하가 작아짐을 알 수 있다.

3. 소프트웨어 신뢰도 성장 모델(SRGM)

3.1 개요

S/W 개발단계에서 테스트하는 구현 S/W 시스템에 대해서 고찰해보기로 한다. S/W 고장은 시스템 내에 잔존하는 오류에 의해 발생되는 프로그램 운전이 원치 않게 중지되는 것으로 정의한다. SRGM 영역에서 아래와 같은 일반적인 가정을 도입한다.

- ○ S/W 시스템은 S/W 오류에 의해서 무작위 시간으로 발생되는 S/W 고장에 달려 있다. 즉, 언제나 S/W 고장이 발생될 수 있다.
- ○ S/W 고장이 발생될 때마다 이것을 일으키는 S/W의 오류를 즉시 제거하며, 새로운 오류는 도입되지 않는다.

S/W의 테스트에 의해서 검출되는 누적 오류수의 시간종속적 동태를 기술하는 데어 일반적으로 오류의 검출 시간 단위로서 역일 시간이나 장비의 실행 시간과 같은 테스트 시간을 사용한다. $\{N(t), t\geq 0\}$를 시간간격$(0, t)$에서 검출되는 누적 오류(또는 고장)의 수를 나타내는 계수 공정이라 하면, NHPP의 평균치함수로 불리는 $N(t)$의 기대치는 $m(t)$로 정의한다. NHPP에 근거한 SRGM은 아래와 같이 정의한다.

$$\Pr\{N(t)=n\}=\frac{\{m(t)\}^n}{n!}exp[-m(t)] \qquad (3.1)$$

여기서,

$$m(t)=a[1-\exp(-bt)] \qquad (3.2)$$

로 표현되고, $m(t)$를 다음과 같이 놓으면

$$m(t)=\int \lambda(x)dx \;=abe^{-bt} \qquad (3.3)$$

$\lambda(t)$는 NHPP의 강도함수라 하고 순간 오류검출비를 의미한다. $a(=m(\infty))$를 최종적으로 검출될 기대누적오류의 수 즉, 산출할 최초의 기대오류라고 정의하면 다음과 같은 식을 쉽게 유도할 수 있다.

$$\lim_{t\to\infty}\Pr\{N(t)=n\}=\frac{a^n}{n!}e^{-a}, n=0,1,2,\cdots \qquad (3.4)$$

이는 $N(t)$가 오랜기간 동안 테스트를 한 후 평균치 a를 가진 Poisson을 따른다는 것을 의미한다. 매우 중요한 S/W 신뢰도 성장지수로서 테스트시각 t에서의 단위 오류당 단위 시간당 오류 검출비를 아래와 같이 정의하며,

$$d(t)=\lambda(t)/[a-m(t)] \qquad (3.5)$$

$d(t)$와 $m(t)$ 사이는 다음과 같은 관계가 있다.

$$m(t)=a[1-\exp(-\int_0^t d(u)du)] \qquad (3.6)$$

S/W 신뢰도 계산에 대한 정량적인 척도를 유도할 때 아래와 같은 확률변수를 정의한다.

S_k : k번째 고장발생 시각, $S_k=\sum_{i=1}^{k}X_i$

$\overline{N}(t)$: 테스트시각 t에서 시스템에 잔존하는 오류의 수, $\overline{N}(t)=N(\infty)-N(t)$

X_k : $(k-1)$번째 고장과 k번째 고장사이의 시간간격, k=1, 2, ···, n

이렇게 정의하면 $\overline{N}(t)$의 기대치와 분산(Var)은 아래와 같이 계산된다.

$$n_r \equiv E[\overline{N}(t)]$$
$$= a\exp[-\int_0^t d(u)du] = Var[\overline{N}(t)] \quad (3.7)$$

S/W 신뢰도는 $S_{k-1}=t$로 주어진 X_k의 조건 생존 확률이며, 아래와 같이 표현한다.

$$R(x|t) \equiv \Pr\{X_k > x | S_{k-1} = t\}$$
$$= \exp[-a\{e^{(-\int_0^{t+x} d(u)du)} - e^{(-\int_0^t d(u)du)}\}] \quad (3.8)$$

이는 k와 무관하다. S/W의 신뢰도는 $(t,\ t+x)$에서 고장이 발생하지 않을 확률을 표현한다.

3.2 기존의 SRGM

S/W 테스트 단계 기간 동안 S/W 오류 검출 공정에서 S/W의 시간 간격과 누적오류 검출 사이를 나타내는 S/W 신뢰도 성장 곡선을 관찰한다. S/W 신뢰도 성장 곡선에 대한 두 개 형태의 모양이 있다. 지수 및 S형 성장곡선이 그것이다. 지수 및 S형 S/W 신뢰도 성장곡선을 기술하는 SRGM은 각각 지수형 및 S형 SRGM이라 한다. NHPP에 근거를 둔 기존의 여러 가지 SRGM을 아래와 같이 간략하게 정리한다.

3.2.1 지수형 SRGM

고엘과 오꾸모또가 처음으로 NHPP에 근거한 SRGM을 제안하였다. 이 모델은 지수형 SRGM이라 부르며, S/W의 고장 현상을 기술해준다. 지수형 성장곡선을 보여주는 평균치함수는 아래와 같다.

$$m(t) = a[1-\exp(-bt)],\ b>0 \quad (3.9)$$

여기서, b는 임의의 시각에서의 오류당 오류 검출비를 나타낸다. 정의에 의해서

$$\lambda(t) = \frac{dm(t)}{dt} = ab\cdot\exp(-bt) \quad (3.10)$$

이다.

3.2.2 수정지수형 SRGM

지수형 SRGM에 대한 동차 오류 검출율과는 대조적으로 오류 검출 능력은 전 테스트 기간에 걸쳐서 비동차로 생각되며, 이는 앞에서 검출된 오류가 후에 검출되는 것과는 상이하기 때문이다. 그래서, Yamada와 Osaki는 두 가지 형태의 오류가 있다는 가정 하에 비동차 오류검출비 모델을 제안하였다. 여기서 두 가지 형태의 오류란 검출이 쉬운 형태의 오류(형태 1)와 검출이 어려운 형태의 오류(형태 2)를 말한다. 수정 지수형 SRGM이라 하는 이러한 NHPP 모델은 다음과 같은 평균치함수를 가진다.

$$m_p(t) = a\sum_{i=1}^{2} p_i[1-\exp(-b_i t)],\quad 0<b_2<b_1<1,$$
$$p_1+p_2=1,\ 0<p_1,p_2<1 \quad (3.11)$$

여기서, b_1, b_2 : 두 가지 형태의 오류당 오류검출비 p_1, p_2 : 형태 1, 2의 오류내용부분 즉, p_1a, p_2a는 각각 형태 1, 2 오류의 예상 초기오류 내용이다.

$$\lambda(t) = \frac{dm(t)}{dt} = a\sum_{i=1}^{2} p_i b_i\cdot\exp(-b_i t) \quad (3.12)$$

이므로, 테스트시각 t에서의 오류당 오류 검출비를 다음과 같이 쓴다.

$$d(t) \equiv d_p(t) = \frac{a\sum_{i=1}^{2} p_i b_i\cdot\exp(-b_i t)}{a - a\sum_{i=1}^{2} p_i\{1-\exp(-b_i t)\}}$$

$$= \sum_{i=1}^{2} \frac{p_i e^{-b_i t}}{p_1 e^{-b_1 t} + p_2 e^{-b_2 t}} \cdot b_i \quad (3.13)$$

$m_p(t)$가 단조증가 함수인 것은 명백하다.

3.2.3 지연 S형 SRGM

S/W 오류 제거 현상에서 테스트 공정이 S/W 고장 검출 공정 뿐만 아니라 S/W 오류 격리 공정으로도 구성된다는 것을 반드시 가정해야 한다. Yamada는 그러한 오류 검출 공정에 대해서 지연 S형 SRGM을 제안하였으며, 이 지연 S형 SRGM에서는 검출오류의 누적개수에 대한 관찰된 성장곡선이 S형이다. 이 NHPP모델은 아래와 같은 평균치 함수를 가진다.

$$m(t) = a[1-(1+bt)\exp(-bt)], \quad b>0 \quad (3.14)$$

$$\lambda(t) = \frac{dm(t)}{dt} = ab^2 t \cdot \exp(-bt) \quad (3.15)$$

이 식은 S형 성장곡선을 보여주고 있다. 파라미터 b는 고장검출비(와 오류 격리비)를 나타낸다.

$$d(t) \equiv d_M(t) = \frac{ab^2 t \cdot \exp(-bt)}{a - a[1-(1+bt)\cdot\exp(-bt)]} = \frac{b^2 t}{(1+bt)} \quad (3.16)$$

이것은 테스트시각 t에서 단조증가이다.

3.2.4 굴곡 S형 SRGM

Ohba는 또다른 S형 SRGM을 제안하였다. 이 모델은 굴곡 S형 SRGM이며, 검출 오류 상호간의 종속성을 가진 S/W 고장 검출 현상을 기술한다. 오류 검출 공정에서 고장을 더 많이 검출하면 할수록 검출되지 않은 고장이 더욱 더 검출되기 쉽게 된다. 이 NHPP모델은 아래와 같은 평균치 함수를 가진다.

$$I(t) = a[1-\exp(-bt)]/[1+c\exp(-bt)], \quad b>0,\ c>0 \quad (3.17)$$

$$\lambda(t) = \frac{dm(t)}{dt} = \frac{ab(1+c)\cdot\exp(-bt)}{[1+c\cdot\exp(-bt)]^2} \quad (3.18)$$

이 식은 S형 성장곡선을 보여주고 있다. 파라미터 b와 c는 각각 고장검출비와 굴곡인자를 나타낸다.

$$d(t) \equiv d_I(t) = \frac{ab(1+c)\cdot\exp(-bt)}{[1+c\cdot\exp(-bt)]^2} \Big/ \left\{a - \frac{a[1-\exp(-bt)]}{1+c\cdot\exp(-bt)]}\right\} = \frac{b}{1+c\exp(-bt)} \quad (3.19)$$

이것은 테스트시각 t에서 단조증가이다.

3.2.5 결정론적 SRGM

S/W 시스템의 오류 내용을 산출하기 위해 상기에서 기술한 확률적 SRGM 외에 로지스틱 및 곰페르츠 성장곡선을 적합시켜 결정론적 SRGM을 널리 사용해왔다. 일본에서는 얼마간의 컴퓨터 제작업체와 컴퓨터 취급업체들이 로지스틱 및 곰페르츠 성장곡선모델을 적용하고 있다. 본래는 수요 경향, 경제성장, 또는 장래의 인구를 예측하기 위해 성장곡선을 개발하였다. 시각 t까지 검출되는 오류의 누적예측개수는 아래와 같은 로지스틱 성장곡선모델로 표현한다.

$$n_L(t) = k/[1+m\cdot\exp(-pt)], \quad m>0,\ p>0, k>0 \quad (3.20)$$

또한, 곰페르츠 성장모델은 아래와 같이 표현한다.

$$n_G(t) = k a^{(b^t)}, \quad 0<a<1,\ 0<b<1,\ k>0 \quad (3.21)$$

k, p, m, a, b는 점화분석에 의해서 산출하는 상수파라미터이다. 두 모델에서 상수 k는 S/W 시스템에서 초기에 존재하고 있던 기대오류 내용이다.

4. 테스트 노력 함수

주어진 기간에 맞추어 소프트웨어 시스템을 개발하고자 할 때 시간, 자금, 인력과 같은 자원들이 소요된다. 특히, 소프트웨어 테스트에 필요한 자원들이 소프트웨어 신뢰도에 상당한 영향을 미친다. 소프트웨어 개발 자원 전체의 약 40-50%가 테스트 단계에서 소요된다. 테스트 노력은 CPU 시간의 양, 실행된 테스트 케이스의 수 등으로 나타낼 수 있다. 이러한 테스트 기간에 대한 테스트 자원의 소요곡선은 테스트 노력 곡선으로 생각할 수 있다. 소프트웨어 신뢰도 모델링 분야에서는 소프트웨어의 개발 노력을 전통적인 지수함수, 레일레이함수, 또는 이들의 일반적인 형태인 웨이불 곡선으로 표현한다. 그러나, 많은 소프트웨어 테스트 환경에서 이러한 3개의 노력함수 곡선만으로 소프트웨어 테스트 노력함수를 기술하는 것은 어려운 일이다. 본 논문에서는 로지스틱 테스트 노력 함수도 소프트웨어 개발/테스트 노력 곡선으로 표현될 수 있다는 것을 보여주고자 한다.

4.1 평균치 함수 및 신뢰도

초기치를 고려한 누적 테스트 노력 함수를 $W^*(t)$로 가정하여 소프트웨어를 시각 t에서 발행하는 경우, 발견되는 누적 결함 분포는 평균치 함수[2]

$$m(t) = a(1 - e^{-\gamma W^*(t)}) \tag{4.1}$$

이므로, 잔여 분포

$$\overline{a} = a - a(1 - e^{-\gamma W^*(t)}) = a \cdot e^{-\gamma W^*(t)}$$

이다. 이것이 소프트웨어를 발행해서 운전하는 초기 결함수이므로

$$m(t+x) = a \cdot e^{-\gamma W^*(t)} \cdot (1 - e^{-\gamma W^*(x)}) + m(t) \tag{4.2}$$

로서,

$$R(x|t) = e^{[-a \cdot e^{-\gamma W^*(t)}(1 - e^{-\gamma W^*(x)})]} \tag{4.3}$$

이 된다. 따라서, 시각 t=T에서 발행된 소프트웨어의 신뢰도는 경과시간 x에 대해서 지수 함수적으로 감소한다. 목표 신뢰도를 $R(x|T) = R_0$라 하면

$$W^*(t) = \frac{1}{r} log \frac{1 - e^{-r W^*(x)}}{\frac{1}{a} log \frac{1}{R_0}} \tag{4.4}$$

인 방정식을 유도할 수 있으며, 이로부터 최적인도 시각 T^*=T를 구한다.

4.2 웨이불형 테스트노력 함수

Yamada[7]가 제안한 웨이불형 테스트 노력함수에 의하면 소프트웨어의 테스트 노력이 테스트 단계 전체에 걸쳐서 일정하다고 가정하는 것은 비현실적이다. 그리고, 순간적인 테스트 노력이 결국은 테스트 수명주기 동안 감소한다. 그 이유는 누적 테스트 노력이 유한치에 접근하기 때문이다. 그 어떤 소프트웨어 개발회사도 소프트웨어 개발에 무한정으로 자원을 투입하지 않기 때문에 이러한 가정이 합리적이라 할 수 있다. 여러 관련 문헌에 의하면 테스트 노력이 웨이불형 분포로 설명될 수 있고, 아래와 같은 3개의 케이스를 가진다는 것을 보여주고 있다.

1) 지수함수형 곡선 : (0, t]에서 소요되는 누적 테스트 노력은

$$W(t) = N[1 - e^{-\beta t}] \tag{4.5}$$

로서 웨이불 함수의 m=1인 경우에 해당되며,

신뢰도

$$R(x|t) = \exp[-a \cdot e^{-rN(1-e^{-\beta x})} \cdot (1-e^{-rN(1-e^{-\beta x})})] \quad (4.6)$$

를 유도할 수 있다.

2) 레일레이형 곡선 : 소요되는 누적 테스트 노력은

$$W(t) = N[1-\exp^{-\frac{\beta}{2} \cdot t^2})] \quad (4.7)$$

로서 웨이불함수의 m=2인 경우에 해당된다. 신뢰도는

$$R(x|t) = \exp[-a \cdot e^{-rN(1-e^{-\frac{\beta}{2}t^2})} \cdot (1-e^{-rN(1-e^{-\frac{\beta}{2}x^2})})] \quad (4.8)$$

와 같이 유도한다.

3) 웨이불형 곡선 : 소요되는 누적 테스트 노력은

$$W(t) = N[1-e^{-\beta t^m})] \quad (4.9)$$

로서 웨이불 함수의 일반적인 경우, 즉 m>2인 경우에 해당된다. 신뢰도는

$$R(x|t) = \exp[-a \cdot e^{-rN(1-e^{-\beta t^m})} \cdot (1-e^{-rN(1-e^{-\beta x^m})})] \quad (4.10)$$

와 같다. 웨이불형 곡선(4.9)에 대해서 m=1 또는 m=2일 때 그 결과는 각각 지수함수 곡선이거나 레일레이 곡선이다. 따라서, 이들은 웨이불 함수의 특수한 경우에 해당된다.

4.3 로지스틱형 테스트노력 함수

실제 테스트 노력 데이타가 여러 가지 소요 패턴을 나타내므로 때때로 테스트 노력 비용을 지수함수나 레일레이 곡선만으로 설명하기는 어렵다. 웨이불형 곡선은 일반적인 소프트웨어 개발 환경 하에서 데이터에 잘 맞고, 소프트웨어 신뢰도 모델링[12]에 널리 쓰이지만 m>3일 때 피크현상을 가진다. 그 대안으로 제시된 것이 로지스틱형 테스트노력 함수이다. 이 함수는 실제 프로젝트 탐사에 의해서 보고된 바와 같이 매우 정확하다. (0, t]에서의 누적 테스트 노력 소요는

$$W(t) = \frac{N}{1+A \cdot e^{-\alpha t}} \quad (4.11)$$

이고, 신뢰도는

$$R(x|t) = \exp[-a \cdot e^{-rN(\frac{1}{1+Ae^{-\alpha t}} - \frac{1}{1+A})} \cdot (1-e^{-rN(\frac{1}{1+Ae^{-\alpha x}} - \frac{1}{1+A})})] \quad (4.12)$$

로 표현된다. 웨이불형 테스트 노력 함수와 비교하여 로지스틱 테스트노력 함수인 경우 $W(0) \neq 0$ 이다. 현재의 테스트 노력 소요량은 W(t)의 미분치로서

$$w(t) = \frac{dW(t)}{dt} = \frac{NA\alpha \cdot \exp(-\alpha t)}{[1+A \cdot \exp(-\alpha t)]^2} = \frac{NA\alpha}{[\exp(\frac{\alpha t}{2}) + A \cdot \exp(-\frac{\alpha t}{2})]^2} \quad (4.13)$$

와 같이 표현된다. 이 양은 $t=\frac{1}{\alpha}\log A$ 일 때 최대치 $\frac{N\alpha}{4}$ 를 중심으로 좌우 대칭형이다.

4.4 발행시각 결정

SRGM을 분석하여 목표 신뢰도에 가장 근접하는 유일한 시각을 구할 수 있다. 최적 소프트웨어 발행 시각은 미리 규정한 소프트웨어 목표 신뢰도에 가장 근접하는 시각이다. 웨이불형인

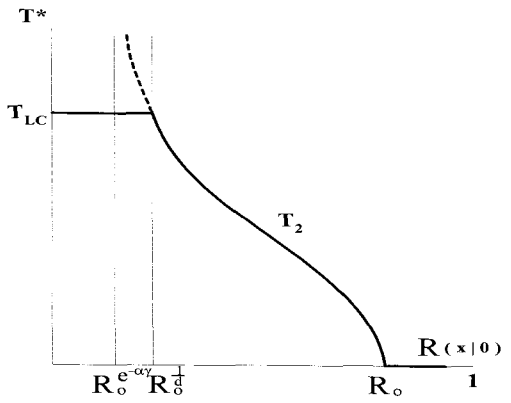
경우를 예로 들면 발행시각 T에서의 신뢰도는 식(4.10)과 같으므로 여기서 목표신뢰도를 만족시키는 발행시각을 구하면 다음과 같다. 목표신뢰도를 R_o라 하면

$$\exp\left\{-ae^{-\alpha\gamma(1-e^{-\beta T^m})}[1-e^{-\alpha\gamma(1-e^{-\beta T^m})}]\right\}=R_o \tag{4.14}$$

이다. 이를 만족하는 발행시각을 T^*라 하면

$$T^*=\left\{-\frac{1}{\beta}\ln\left[1+\frac{1}{\alpha\gamma}\ln\frac{\ln R_o}{\ln R(x|0)}\right]\right\}^{\frac{1}{m}} \tag{4.15}$$

로 표현된다. $R(x|0)$와 목표신뢰도에 이르는 시각 T^*와의 관계는 <그림 4.1>과 같다.



<그림 4.1> 목표신뢰도와 발행시각 곡선

5. 파라미터의 산출법

5.1 파라미터 산출법

위의 각 경우에서 정의된 테스트 노력 함수에서 파라미터 N, A, a, β는 최소자승법(LSE)으로 산출하는 것을 제안한다. 최대가능성 평가자(MLE)는 한 집합의 동시방정식을 풀어서 파라미터를 산출하며, s-신뢰 구간을 구동하는데 더 좋은 방법이다. 그러나, 그 방정식이 너무 복잡하므로 보통은 수치 해석적으로 푼다. LSE는 실제로 관찰/획득한 것과 예측한 것 사이의 차이를 제곱해서 더한 총값을 최소화하는 것이다. LSE는 중간 크기의 표본에 최적인 것으로 알려지고 있으며, 최적점 산출을 제공한다. 최소자승법을 적용하기 위한 산출 공식$S(N, A, a)$은 다음과 같다.

$$S(N,\ A,\ a)=\sum_{k=1}^{n}[W_k-W(t_k)]^2 \tag{5.1}$$

W_k = (0, t]기간동안 실제로 소요되는 누적 테스트 노력

$W(t_k)$ = 테스트 노력 함수에 의해서 산출된 누적 테스트 노력

S를 N, A, a에 관하여 미분하여 그 편미분치를 0으로 놓으면, 그리고 이 항들을 재정비하면 이러한 형태의 비선형 최소 자승 문제를 푼다. 또한, 상기에서 정의한 테스트 노력 함수의 산출 파라미터를 이용하여 MLE에 의해서 $m(t)$의 신뢰도 성장 파라미터 a와 r을 구할 수 있다. 비동차포아송과정(NHPP)의 표준이론으로부터 임의의 $t\geq 0$, $x>0$에서

$$\Pr\{N(t+x)-N(t)=k\}=\frac{\{m(t+x)-m(t)\}^k}{k!}\cdot e^{-m(t+x)+m(t)} \tag{5.2}$$

이므로, 여기에 $m(t+x)=m(t_k)$, $m(t)=m(t_{k-1})$, $k=m_k-m_{k-1}$를 대입하면

$$\Pr\{N(t_k)-N(t_{k-1})=m_k-m_{k-1}\}=\frac{\{m(t_k)-m(t_{k-1})\wedge m_k-m_{k-1}(m_k-m_{k-1})!}{\cdot e^{-m(t_k)+m(t_{k-1})}} \tag{5.3}$$

로서 주어진 시간간격 $(0,\ t_k)(0<t_1<t_2<\cdots<t_n)$에서 검출결함의 누적 갯수 m_k에 대한 데이터를 관찰할 수 있다. 따라서, $m(t)$를 가지는 NHPP의 미지 파라미터 a와 r에 대한 조인트 확률밀도함수 즉, MLE 함수는

$$L=\Pr\{N(t_i)=m_{i,\ i=1,\cdots,\ n}\}$$

$$= \prod_{k=1}^{n} \frac{[m(t_k) - m(t_{k-1})]^{(m_k - m_{k-1})}}{(m_k - m_{k-1})!} \cdot e^{-(m(t_k) - m(t_{k-1}))} \quad (5.4)$$

로 표시되므로, 다음과 같은 방법으로 파라미터를 구하도록 제안한다.

$$\log(L) = \sum_{k=1}^{n} (m_k - m_{k-1}) \cdot \log[m(t_k) - m(t_{k-1})] - \sum_{k=1}^{n} \log[m(t_k) - m(t_{k-1})]! - \sum_{k=1}^{n} \log[m(t_k) - m(t_{k-1})] \quad (5.5)$$

이것을 a, r에 관하여 편미분한다.

$$a = \frac{m_n}{1 - e^{-rW^*(t_n)}} = \frac{m_n}{1 - \Phi_n} \quad (5.6)$$

여기서,

$$\phi_k = e^{-rW^*(t_k)} \quad (5.7)$$

$$aW^*(t_n)e^{-rW^*(t_n)} = \sum_{k=1}^{n} (m_k - m_{k-1}) \cdot \frac{-W^*(t_{k-1})e^{-rW^*(t_{k-1})} + W^*(t_k)e^{-rW^*(t_k)}}{e^{-rW^*(t_{k-1})} - e^{-rW^*(t_k)}} \quad (5.8)$$

$$aW^*(t_n)\Phi_n = \sum_{k=1}^{n} (m_k - m_{k-1}) \cdot \frac{-W^*(t_{k-1})\Phi_{k-1} + W^*(t_k)\Phi_k}{\Phi_{k-1} - \Phi_k} \quad (5.9)$$

으로 표현된다.

5.2 발행시각 결정

구한 파라미터를 적용하여 각각의 경우에 대한 최적 발행 시각 결정 방법을 다음과 같이 제안한다.

1) 지수함수형

발행시각을 $t=T$라 하고, 목표신뢰도를 $R(x|T)=R_0$라 하면

$$R_0 = \exp\left[-ae^{-rN(1-e^{-\beta T})}(1-e^{-rN(1-e^{-\beta x})})\right] \quad (5.10)$$

에서 발행시각을 구하면

$$T = -\frac{1}{\beta}\log\left(1 + \frac{1}{rN}\log\frac{\log R_0}{\log R(x|0)}\right)$$

로 표현된다.

2) 레일레이형

$$R_0 = e^{[-ae^{-rN(1-e^{-\beta^2 T^2}}(1-e^{-rN(1-e^{-\beta^2 x^2})})]} \quad (5.11)$$

에서

$$T = \left\{-\frac{2}{\beta}\log\left(1 + \frac{1}{rN}\log\frac{\log R_0}{\log R(x|0)}\right)\right\}^{1/2} \quad (5.12)$$

이다.

3) 웨이불형

$$R_0 = \exp[-ae^{-rN(1-e^{-\beta T^m})}(1-e^{-rN(1-e^{-\beta x^m})})] \quad (5.13)$$

에서

$$T = \left\{-\frac{1}{\beta}\log\left(1 + \frac{1}{rN}\log\frac{\log R_0}{\log R(x|0)}\right)\right\}^{1/m} \quad (5.14)$$

이다.

4) 로지스틱형

$$R_0 = \exp\left[-ae^{-rN\left(\frac{1}{1+Ae^{-\alpha T}} - \frac{1}{1+A}\right)} \cdot (1-e^{-rN\left(\frac{1}{1+Ae^{-\alpha T}} - \frac{1}{1+A}\right)})\right] \quad (5.15)$$

에서

$$T=-\frac{1}{\alpha}log\frac{1}{A}\left(\frac{1}{\frac{1}{1+A}-\frac{1}{rN}log\frac{\log\frac{1}{R_0}}{a\{1-e^{-rW(x)}\}}}-1\right)$$

$$=-\frac{1}{\alpha}log\frac{1}{A}\left(\frac{1}{\frac{1}{1+A}-\frac{1}{rN}log\frac{\log R_0}{\log R(x|0)}}-1\right) \quad (5.16)$$

이다. 단, 모든 경우에 대해서 $e^{-rW(T_{LC})}=p$ 라 할 때 $R^p<R(x|0)<R_0$ 인 경우에만 발행시각에 관한 문제가 의미를 갖게 되어 이 때 유일하고도 유한한 해가 존재한다.

5.3 적용 예

3항에서 연구한 NHPP 모델의 수치 적용 예를 보이기 위해서 실제적인 소프트웨어 오류 데이타로서 [13]에서 조사된 데이터 집합을 적용하였다. 이 데이터를 이용하여 위의 방정식들에 의해서 각종 파라미터를 산출한 결과는 <표 5.1>과 같다.

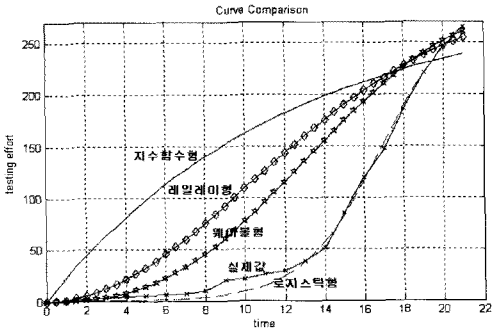
<표 5.1> 파라미터 산출

구분	지수함수형	레일레이형
N	292.4698	291.4158
β	0.0820	0.00938
r	0.04396	0.04548
a	146.30	148.60
m	-	-
A	-	-
a	-	-
T^*	10.8	10.3

<표 5.1> 파라미터 산출(계속)

	웨이불형	로지스틱형
N	292.6700	295.8442
β	0.001154	-
r	0.07992	0.32568
a	147.35	146.00
m	2.390	-
A	-	4673.86
a	-	0.5013
T^*	9.9	10.4

식을 만족하는 계수를 구하기는 일반적으로 쉽지 않으므로, 수치해석적인 방법에 의하여 반복적인 연산을 수행하였으며, 이론치와 적용 데이타의 차이가 최소로 되는 시점의 값을 최적값으로 정하였다. <표 5.1>의 값을 가지고 각 경우의 파라미터에 의한 테스트 노력곡선이 실제의 현상과 어떤 연관관계를 가지는가를 검토하기 위하여 추이곡선을 그렸으며, 그 결과는 다음 <그림 5.1>과 같다.

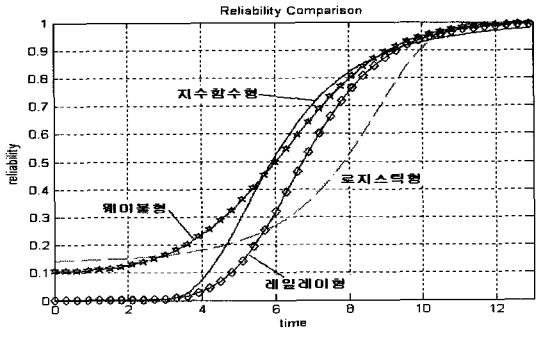


<그림 5.1> 테스트 노력 곡선 비교

그림에서 지수함수형은 실선으로, 레일레이형은 다이어먼드형으로, 웨이불형은 별표형으로, 로지스틱형은 중간이 일정한 간격으로 끊어진 실선으로 표시하였다. 4가지 경우의 테스트 노력 곡선에 대한 계수를 구하여 실제 관측된 데이터 곡선과 비교하였을 때, 그림에서 보는 바와 같이 테스트 중간에서 지수함수형과 레일레이형, 웨이불형 모두가 실제와 큰 차이를 나타내고 있다. 단, 테스트 시간이 길어지면 누적치가 모두 실제값에 접근해가고 있다. 이와 대조적으로 로지스틱형은 테스트 초창기만 약간 차이를 보이고 있으나, 테스트 시간이 경과함에 따라 실제 현상과 거의 근접한 것을 알 수 있다. 초창기에 차이가 나는 것은 테스트 초기에는 테스트 여건이나 테스트자의 숙련도 등이 미숙하거나 불확실한 이유 때문으로 사료된다. 따라서, 본 사례의 경우에는 로지스틱형이 적합하다는 것을 알 수 있다. 각각의 조건에 따라서 나머지 형태에 적합한 테스트 노력이 있을 수 있으므로 모든 경우에 대해서 로지스틱형만이 우수하다고 단정할 수는 없다.

이러한 각종 파라미터에 의한 테스트노력곡선을 이용하여 각 경우의 신뢰도 성장 곡선을

그리면 <그림 5.2>와 같다. 단, 여기에서 목표 신뢰도 R_0=0.95, 경과시간 x=1로 가정하였다.



<그림 5.2> SRGM 비교

마찬가지로 그림에서 지수함수형은 실선으로, 레일레이형은 다이어먼드형으로, 웨이불형은 별표형으로, 로지스틱형은 중간이 일정한 간격으로 끊어진 실선으로 표시하였다. 그림을 검토해볼 때 지수함수형인 경우, 초창기에 신뢰도 성장이 저조하다가 T=3 부근에서 급격히 성장되어 T=10 부근에서 안정된다. 레일레이형인 경우, 지수함수형인 경우와 마찬가지로 초창기에 성장이 극히 저조하다가 T=4 부근에서 서서히 성장되어 T=10 근처에서 안정된다. 웨이불형인 경우는 테스트 초기부터 10% 정도의 신뢰도를 가지며, 테스트 시간 경과에 따라 원활하고도 완만한 성장을 한다. 이의 특징은 앞의 두 가지경우에 비하여 처음 신뢰도가 10% 정도 된다는 것이다. 로지스틱형인 경우는 테스트 초창기에 14% 정도의 신뢰도를 가지고 있으며, 타 형태의 테스트 노력에 비하여 신뢰도 성장이 극히 저조하다가 T=5 부근에서부터 서서히 성장하기 시작하여 T=7 부근에서 갑자기 성장, T=10 부근에서 안정되는 지연 S 형태를 취한다. 한편, 목표신뢰도 R_0=0.95를 만족시키는 인도시각은 지수함수형인 경우, T=10.8, 레일레이함수형인 경우, T=10.3, 웨이불형인 경우, T=9.9, 로지스틱형인 경우, T=10.4로서 모두가 대동소이하다. 그러나, <그림 5.1>의 추이에서 보듯 본 사례의 경우, 로지스틱인 경우가 가장 정당성을 갖는다고 할 수 있으므로, T=10.4라고 하는 것이 타당하다.

6. 결 론

소프트웨어를 개발하여 발행 전에 디버깅을 효과적으로 수행하여 소프트웨어의 신뢰도를 향상시키는 것이 중요하다. 소프트웨어 신뢰도를 측정하기 위한 공통적인 접근법은 소프트웨어 고장으로부터 구한 가용한 데이터로부터 산출된 파라미터를 가진 분석모델을 이용하는 것이다. 한편, 소프트웨어 개발에 있어서 상당한 양의 자원이 소프트웨어의 테스트에 쓰인다. 소프트웨어의 신뢰도를 산정하고 신뢰도를 성장시키고자 할 때 이러한 테스트 노력이 중요한 요소라 하는 데에 소요되는 개발자원의 양에 전적으로 의존한다. 연구 결과에 의하면 테스트 기간에 대한 테스트 자원의 소요 거동은 테스트 노력곡선으로 생각할 수 있다. 그간 여러 문헌에서 제시된 테스트 노력 함수로서는 지수함수형, 레일레이형, 웨이불형이며, 최근에는 로지스틱형도 제시되고 있다.

본 논문에서는 상기 네 가지 경우에 대한 함수의 파라미터 산출법을 연구하고, 이러한 파라미터 산출법과 함수들이 실제와 얼마나 부합되는가를 검토하기 위하여, 실제 개발 과정을 통해 수집된 경험 데이터를 가지고 각 경우에 대한 파라미터를 산출하고 이를 비교하였다. 테스트노력과 오류검출 현상이 소프트웨어 개발환경에 따라 불규칙성을 나타내는 것이 일반적이라고 할 때, 지수함수형이나 레일레이형은 이러한 현상에 융통성이 부족한 것으로 사료된다. 웨이불형은 앞의 두 가지 형에 대한 보완형으로 제안되었다고 할 만큼 융통성이 있어서 실제 현상과 유사한 특성을 나타내지만, 시간 지수가 3 이상이면 문제가 있는 것으로 검토되었으므로 이를 유의할 필요가 있다. 최근에 로지스틱형이 제안되고 있는데, 본 연구에서 비교 연구한 바에 의하면 사례의 현상에 부합하는 것으로 판명되었다. 그리고, 목표신뢰도를 정하고 산출된 파라미터에 의해서 각 경우의 발행시각을 계산해 본 결과, 어느 정도 합리적인 결과를 얻었으나, 테스트 초기에는 불규칙하고 불확실한 테스트 노력 현상 때문에 산출된 파라미터의 신뢰성이 저하된다는 것을 발견하였다. 본 예의 경우 로

지스틱형은 비교적 정확한 분석 결과를 나타낼 수 있는 안이라 할 수 있다.

본 논문에서는 목표 신뢰도만을 연구하였으나, 추후 소프트웨어를 개발하여 테스트를 거쳐서 고객에게 인도한 후 운영하는 전 기간에 걸쳐서 소요되는 소프트웨어의 비용을 고려한 발행시각을 결정하는 연구가 계속되어야 할 것이다.

참 고 문 헌

[1] C. V. Ramamoorthy, F. B. Bastani, "Software reliability - Status and perspectives", IEEE Trans. on Software Eng., vol. SE-8, pp354-371, 1982 Aug.

[2] S. Yamada, H. Ohtera, H. Narihisa, "Software reliability growth models with testing- efforts", IEEE Trans. Reliability, vol. R-35, pp19-23, 1986 Apr.

[3] Michael A. Friedman, Jeffrey M. Voas, " Software Assessment : Reliability, Safety, Testability", John Wiley & Sons, Inc., pp199-204, 1995

[4] Syed A. Hossain, Ram C. Dahiya, " Estimating the Parameters of a Non-homogeneous Poisson-Process Model for Software Reliability", IEEE Trans. Reliability, vol. 42, no.4, pp604-612, 1993 Dec.

[5] Peter Spreij, " Parameter Estimation for a Specific Software Reliability Model", IEEE Trans. on Reliability, vol. R-34, no. 4, pp323-332, 1985.Oct

[6] Tapan Kumar Nayak, "Software Reliability: Statiscal Modeling & Estimation", IEEE Trans. on Reliability, vol. R-35, no.5, pp566-570, 1986 Dec.

[7] S. Yamada, J. Hishitani, S. Osaki, "Software - Reliability Growth with a Weibull Test-Effort : A Model & Application", IEEE Trans. Reliability, vol. 42, no.1, pp100-106, 1993 March

[8] Min Xie, Bo Yang, " A study of the effect of imperfect debugging on software development cost", IEEE Trans. on Software Eng., vol.29, no.5, pp471-473, 2003.5

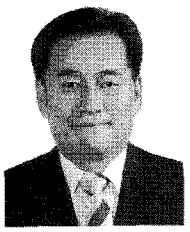
[9] X. Zhang, X. Teng, H. Pham, "considering fault removal efficiency in software reliability assessment", IEEE Trans. on Systems, man, and cybernetics, vol.33, no.1, pp114-120, 2003.1

[10] S. Yamada, S. Osaki, "Cost-reliability optimal release policies for software systems", IEEE Trans. on Reliability, vol. R-34, 1985 Dec., pp422-424

[11] Amrit L. Goel, Kazu Okumoto, "Time - Dependent Error - Detection Rate Model for Software Reliability and Other Performance Measure", IEEE Trans. on Reliability, vol R-28, No.3, 1979.8. pp206-211

[12] Chin-Yu Huang, Sy-Yen Kuo, "Analysis of Incorporating Logistic Testing-Effort Function into Software Reliability Modeling", IEEE Trans. on Reliability, vol.51, pp261-270, 2002, Sep.

[13] Xuemei Zhang, Hoang Pham, "An analysis of factors affecting software reliability", The Journal of Systems and Software, 2000. pp43-56



양 계 탁 (Gye-Tak Yang)

- 1984년 2월: 연세대학교 수학과 (이학사)
- 1986년 2월 : 연세대학교 수학과 (이학석사)
- 1992년 2월 : 연세대학교 수학과 (이학박사)
- 1993년 3월~현재 : 건양대학교 정보보호학과 교수
- 관심분야 : 어플리케이션 보안, 보안 솔루션, 암호학, 인증