주문형 소프트웨어의 운영단계 신뢰도에 관한 연구

최 규 식†

요 약

소프트웨어를 개발하여 출시하기 전에 품질을 향상시키는 중요한 방법으로서 신뢰도를 향상시켜야 하며, 이의 직접적인 방법으로서는 테스트를 통하여 결함을 검출하고 수정해가는 것이다. 신뢰도가 목표치에 도달하면 이를 출시(release)하게 되며, 그 후로는 운영중의 신뢰도 문제가 발생하게 된다. 개발 출시된 소프트웨어가 패키지 소프트웨어라 불리는 범용 소프트웨어냐, 아니면 특수목적을 가진 전용 소프트웨어냐에 따라 운영 소프트웨어의 신뢰도 추이가 달라진다. 또한, 개발 테스트 단계의 테스트 노력이 일정하냐, 아니면 웨이불 곡선을 따르냐에 따라서도 신뢰도 평가가 다르다. 본 논문에서는 주문형 소프트웨어에 대해서 일정 테스트 노력을 적용하는 경우와 웨이불 테스트 노력을 적용하는 경우의 운영단계 신뢰도 추이에 대해서 연구한다.

A Study on the Operational Stage Reliability of Dedicated Software

Gyu Shik Che†

ABSTRACT

The software reliability have to be improved as one major method to enhance the quality of developed software, and its defect is detected and modified through testing as a direct way to meet the purpose before releasing. Once its reliability grows up to the target and it is released to public. there may be operational reliability problem. The operational software reliability trend may be different depending on the condition whether it is universal or dedicated. And its reliability estimation is changed based on the condition if it follows uniform testing efforts or Weibull testing efforts. I study the operational reliability trend of dedicated software, applying two case testing efforts for the released item.

키워드 : SRGM, NHPP, 평균치 함수(Mean Value Function), 전용 소프트웨어(Dedicated Software), 목표 신뢰도(Target Reliability), 일정테스트 노력 함수(Uniform Testing Efforts Function), 웨이불 테스트 노력 함수(Weibull Testing Efforts Function)

1. 서 론

소프트웨어의 제품개발에 있어서 소프트웨어의 신뢰도가 핵심사항이라고 할 수 있다. 1970년대 이후 소프트웨어의 신뢰성을 향상시키기 위한 여러 가지 소프트웨어의 신뢰도 모델이 제시되고 검토되었다. 특히, 소프트웨어 개발 후 테스트 단계에 적용하는 신뢰도를 추정하고 예측하는 모델이 많이 개발되었다. 소프트웨어가 주어진 시간 간격 동안 고장이 발생하지 않을 확률 즉, 신뢰도는 소프트웨어의 테스트과정을 계속해서 반복 및 수정하면 더욱 더 향상될 것이다.

그동안 테스트 기간중의 소프트웨어 결함 검출현상을 설명하기 위한 여러 가지 소프트웨어 신뢰도 모델이 개발되었다. 소프트웨어 테스트에 의해서 발견되는 누적결함(또는 소프트웨어 고장간의 시간간격)과 소프트웨어 테스트 시간 간격 사이의 관계를 짓는 모델들을 소프트웨어 신뢰도 성장모델(SRGM ; software reliability growth model)[1]이라 한다. 이러한 모델들을 이용하여 평균 초기결함의 수, 평균 고장간 시간 간격, 임의의 테스트 시간에 소프트웨어 내의 평균 잔여결함수, 소프트웨어 신뢰도 함수와 같은 소프트웨어 신뢰도 척도를 추정할 수 있다.

또한, 소프트웨어 개발에는 많은 개발자원들이 소요된다. 소프트웨어 테스트 단계 기간 동안에는 소프트웨어의 신뢰도가 내재 결함을 검출 및 수정하는데 소요되는 개발자원의 양에 크게 의존한다. Musa 등[2]은 기존 소프트웨어 신뢰도 성장모델을 분류하는 하나의 안을 개발하였다. Yamada 등[3]은 역일 테스트 시간, 테스트 노력량, 테스트 노력에 의해서 검출되는 소프트웨어 결함의 수 사이의 관계를 명시적으로 설명할 수 있는 간단하고도 새로운 모델을 제시하였다. 테스트 노력은 테스트 단계에서 소요되는 인력, CPU 시간, 실행테스트 케이스 등등에 의해서 측정된다.

그동안 많은 연구가와 참여자들에 의해서 보편적으로 널리 연구되고 사용되는 SRGM의 부류는 NHPP(nonhomogeneous Poisson process) 모델이며, 이러한 부류의 모델은 실제로 많은 장점을 가지고 있어서 그간 많은 관심을 끌어왔다.

소프트웨어 신뢰도는 관련 SRGM으로부터 유도되어 왔다. 여기에 두 가지의 상이한 신뢰도 개념이 있는데 즉, 테스트 신뢰도와 운영 신뢰도가 바로 그것이다. 테스트 신뢰도는 테스트 단계에서 결함이 발생하지 않을 확률이며, 운영 신뢰도는 운영 단계에서 결함이 발생하지 않을 확률이

※ 본 연구는 한국과학재단 목적기초연구(R01-2000-000-00273-0) 지원으로 수행되었음.
† 정 회 원 : 건양대학교 IT학부 교수
논문접수 : 2003년 5월 20일, 심사완료 : 2003년 6월 16일

다. 테스트 단계에서는 소프트웨어의 결함이 발견되는 대로 제거되어 동일 또는 유사한 결함이 다시 나타나지 않기 때문에 테스트를 계속 하면 할수록 고장발생률이 줄어든다.

운영 기간 중에는 결함을 제거할 수 있는 경우도 있고, 그렇지 않은 경우도 있다. 보통 패키지 소프트웨어라 불리는 범용 소프트웨어의 경우는 개발자가 고객의 요구를 파악하여 기호에 맞는 소프트웨어를 개발, 불특정 다수인을 상대로 출시하는 경우이므로, 운영중에 소프트웨어의 품질에 문제가 있거나 다른 요건이 생겨도 이를 수정하기는 현실적으로 어렵다. 따라서, 사용자가 사용하는 전 기간에 걸쳐서 일정한 고장발생률을 겪게 되는 경우가 대부분이다. 이 경우, 이를 개발에 반영하여 그 다음 버전을 출시하는 방법을 택하는 것이 일반적이다. 이와는 대조적으로 고객으로부터 주문을 받아서 특수한 용도로 개발하는 전용 소프트웨어 또는 주문형 소프트웨어의 경우에는 운영 중에 검출되는 결함에 대해서 계속 수정을 하며, 품질을 유지 관리하기 때문에 신뢰도의 성장이 가능하다. 이러한 두 가지 신뢰도 개념은 다르며, 따라서 이들이 운영단계의 신뢰도에 어떠한 영향을 미치는지도 연구해볼 만한 가치가 크다.

본 논문에서는 테스트 노력이 일정하다고 가정하여 신뢰도를 향상시키는 방법과, 테스트 노력이 웨이불곡선을 따른다고 가정하여 신뢰도를 향상시키는 방법을 적용하여 운영단계의신뢰도 추이 및 성장법을 연구하고자 한다.

2장에서는 소프트웨어의 신뢰도에 관한 일반적인 개념 및 테스트 후 및 출시시각, 출시 후 경과시간에 대한 신뢰도의 추이를 연구하고, 3장에서는 소프트웨어의 목표 신뢰도를 고려한 출시시각 문제를 고려한 연구를 하였으며, 4장에서는 일정 테스트 노력을 적용한 경우와 웨이불 테스트 노력을 적용한 경우의 운영 신뢰도에 관하여 연구하였다.

기호설명

$N(t)$: 시각 t까지 검출되는 소프트웨어의 누적결함 갯수
a : 초기부터 소프트웨어 내에 존재하고 있는 결함의 갯수
b, b_1, b_2 : 일반적인 경우, 테스트 시간중, 출시후 운영 시간중의 결함 검출비, $b_1 \geq b_2$
$m(t)$: $E[N(t)]$, 평균치 함수
$R(x|t)$: 소프트웨어의 신뢰도, 시각 $t(x \geq 0)$에서 결함이 검출된 후 $(t, t+x)$에서 고장이 일어나지 않을 확률
R_o : 목표 신뢰도, $0 < R_o < 1$
T_{LC} : 소프트웨어의 수명주기
T : 전체 테스트 시간
T^* : 최적 소프트웨어 테스트 시간
T_1 : 일정 테스트 노력에서 목표 신뢰도를 만족시키는 유일 해 T
T_2 : 웨이불 테스트 노력에서 목표 신뢰도를 만족시키는 유일 해 T
d : 상수, $e^{bT_{LC}}$, $d > 1$
g : 상수, $\exp[-\alpha\gamma(1-e^{-\beta T_{LC}^{m}})]$, $0 < d < 1$
α : 소프트웨어 테스트에서 필요로 하는 테스트 노력 소요 총량
β, m : 척도모수, 형상모수
s : 결함이 발견되는 시각
$w(t)$: 웨이불 곡선

2. 테스트 노력을 고려한 소프트웨어의 신뢰도

소프트웨어 신뢰도는 규정된 환경 하에서 주어진 기간에 소프트웨어를 결함 없이 운영할 수 있는 확률인 것으로 정의한다. S/W 고장은 시스템 내에 잔존하는 결함에 의해 프로그램 운전이 강제적으로 중지되는 것으로 정의한다. SRGM 영역에서 아래와 같은 일반적인 가정을 도입한다[4-5].

① S/W 시스템은 S/W 결함에 의해서 무작위 시간으로 발생되는 S/W 고장에서 자유롭지 못하다. 즉, 언제나 S/W 고장이 발생될 수 있다.
② S/W 고장이 발생될 때마다 이것을 일으키는 S/W의 결함을 즉시 제거하며, 새로운 결함은 도입되지 않는다.
③ 결함제거 공정은 NHPP이다.
④ 소프트웨어 시스템이 시스템의 잔여 결함을 명시함으로써 생기는 무작위 시간에서의 고장에 의존한다.
⑤ 오류(error)를 교정하는데 드는 시간은 극히 미약하고 검출된 오류는 확실히 제거된다.

$\{N(t), t \geq 0\}$를 시간간격$(0, t)$에서 검출되는 누적 결함(또는 고장)의 수를 나타내는 계수공정이라 하면, NHPP의 평균치함수로 불리는 $N(t)$의 기대치는 $m(t)$로 정의한다. NHPP에 근거한 SRGM은 아래와 같이 쓴다.

$$\Pr\{N(t)=n\} = \frac{\{m(t)\}^n}{n!}\exp[-m(t)], \quad t \geq 0,\ n = 0, 1, 2, \cdots \tag{1}$$

$a(=m(\infty))$를 최종적으로 검출될 기대 누적결함의 수 즉, 산출할 최초의 기대결함이라고 정의하면 다음과 같은 방정식을 유도할 수 있다.

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

이는 $N(t)$가 오랜 기간 동안 테스트를 한 후 평균치 a를 가진 Poisson을 따른다는 것을 의미한다. 매우 중요한 S/W 신뢰도 성장지수로서 테스트시각 t에서의 단위 결함당 단위 시간당 결함 검출비를 아래와 같이 정의한다[6].

$$d(t) = \lambda(t)/[a - m(t)] \tag{3}$$

식 (3)에서 $\lambda(t) = \frac{dm(t)}{dt}$로하여 양변을 t에 관하여 적분

하면,

$$\int_0^t d(u)\,du = -\ln(a-m(t))|_0^t = -\ln\frac{a-m(t)}{a},$$

$$a-m(t) = a\cdot\exp\left(-\int_0^t d(u)\,du\right)$$

이므로, $d(t)$와 $m(t)$ 사이는 다음과 같은 관계가 있다.

$$m(t)=a[1-\exp(-\int_0^t d(u)\,du)] \tag{4}$$

2.1 일정테스트 노력 곡선

NHPP의 표준 이론으로부터 임의의 $t\geq 0$과 $x>0$에서

$$\Pr\{N(t+x)-N(t)=k\} = \frac{[m(t+x)-m(t)]^k}{k!}\exp\{-[m(t+x)-m(t)]\} \tag{5}$$

이므로, 소프트웨어의 신뢰도는 다음과 같이 표현할 수 있다.

$$\begin{aligned} R(x|t) &\equiv \Pr\{N(t+x)-N(t)=0\} \\ &= \exp\{-[m(t+x)-m(t)]\} \\ &= \exp[-a(1-e^{-bx})e^{-bt}] = \exp[-m(x)e^{-bt}] \end{aligned} \tag{6}$$

여기서, 평균치 함수는

$$m(t)=a[1-\exp(-bt)] \tag{7}$$

로 표현되고, $m(t)$를 다음과 같이 놓으면

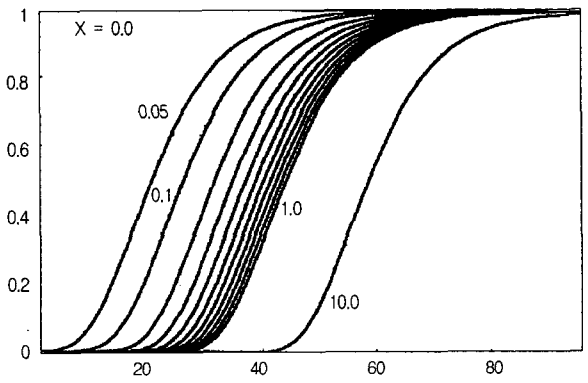
$$m(t)=\int_0^t \lambda(x)dx \text{ 또는 } \lambda(t)=\frac{dm(t)}{dt} \tag{8}$$

와 같이 나타낸다.

$\lambda(t)$는 NHPP의 강도함수라하고 순간 결함 검출비를 의미한다.

즉, 어느 시각 t부터 $(t+x)$ 시각까지 새로운 결함이 발견되지 않을 확률을 신뢰도로 정의하며, 이는 평균치 함수의 차이를 지수함수의 지수로 취한 형태를 하고 있다.

식 (6)으로 표시된 신뢰도의 특성을 이해하기 위한 테스트 시간과 신뢰도의 관계는 (그림 1)과 같다.



(그림 1) 출시시각과 신뢰도 관계-일정 테스트 노력

이 그림에서 최종 결함 수정 후 경과되는 각각의 시간에 대해서 테스트 시간과 신뢰도 성장과의 관계를 보여주고 있다. 일정한 경과시간 x에 대해서 테스트 시간 및 출시시기를 늦추면 늦출수록 신뢰도가 s형 곡선으로 성장함을 알 수 있다. 그림에서 x = 0.0일 때는 테스트 시간에 관계없이 신뢰도가 1이나, x의 값이 커지면 커질수록 곡선이 횡축의 우측으로 이동하여 신뢰도가 저하됨을 알 수 있다. 결함 수정 후 경과시간을 어떤 범위로 하여 신뢰도 성장 기준을 잡는가도 중요한 문제이다.

우리가 원하는 목표 신뢰도를 R_o과 하면 식 (6)으로부터

$$\exp[-m(x)e^{-bt}]=R_o \tag{9}$$

가 된다.

주어진 기간에 맞추어 소프트웨어 시스템을 개발할 때 시간, 자금, 인력과 같은 자원들이 소요된다. 특히, 소프트웨어 테스트에까지 이르는 자원들이 소프트웨어 신뢰도에 상당한 영향을 미친다. 소프트웨어 개발 자원 전체의 약 40~50%가 테스트단계에서 소요된다.

2.2 웨이블 테스트 노력 곡선

주어진 기간에 맞추어 소프트웨어 시스템을 개발할 때 시간, 자금, 인력과 같은 자원들이 소모된다. 일반적으로, 이러한 소요자원의 시간종속 동태를 예측할 때 레일레이 곡선을 사용해왔다. 특히, 소프트웨어 테스트에까지 이르는 자원들이 소프트웨어 신뢰도에 상당한 영향을 미친다. 소프트웨어 개발 자원 전체의 약 40~50%가 테스트 단계에서 소요된다. Yamada 등[3]은 테스트 기간중의 테스트 노력과 소프트웨어 개발 노력 모두가 레일레이 곡선으로 설명될 수 있다는 것을 가정하여 소프트웨어 테스트에 쓰이는 테스트 노력의 양을 고려한 소프트웨어 신뢰도 성장 모델을 제시하였다. 그들은 레일레이 곡선의 대안으로서 지수곡선도 제안하였다. 그러나, 많은 경우의 소프트웨어 테스트에 있어서 테스트 노력을 지수곡선이나 레일레이 곡선으로만 설명하는 것이 무리가 따른다. 이는 실제 테스트 노력 데이터가 다양한 형상을 나타내기 때문이다. 우리는 시각 t에서의 테스트 노력 형상을 기술하기 위해 테스트 노력함수로서 웨이블곡선을 이용한다. 따라서, 이것을 다음과 같이 표시한다[7].

$$w(t)=\alpha\cdot\beta\cdot m\cdot t^{m-1}\cdot\exp[-\beta t^m] \tag{10}$$

α : 소프트웨어 테스트에서 필요로 하는 테스트 노력 소요 총량

β, m : 척도모수, 형상모수

m = 1, m = 2일 경우 각각 지수 및 레일레이 곡선 테스트 노력함수를 얻을 수 있다. m > 1이면 척도모수는

$$\beta=1/[(\frac{m}{m-1})t_{w\max}^m] \tag{11}$$

$t_{w\ max}$: 테스트 노력 $w(t)$의 양이 최대로 되는 시각이고, 식 (11)의 적분형태

$$W(t)=\alpha(1-\exp[-\beta t^{m}]) \tag{12}$$

는 시각 (0, t]에서의 누적 테스트 노력량을 나타낸다. 방정식 (12)로부터 $t=t_q$이면

$$W(t_q)\simeq 0.63\alpha \tag{13}$$

가 된다.

웨이블 테스트 노력 함수 식 (11)이나 식 (13)에서의 α, β, m은 최소자승법으로 구한다. α, β, m는 (t_k, w_k)형태의 n개 관찰 데이터쌍에 대해서 결정한다. 관찰된 테스트 노력 데이터에 최소자승법을 적용하기 위해 식 (10)을 아래와 같이 변환한다.

$$\ln w(t)=\ln\alpha+\ln\beta+\ln m+(m-1)\ln t-\beta t^{m} \tag{14}$$

식 (14)로부터

$$S(\alpha,\beta,m)=\sum_{k=1}^{n}\{\ln w_k-\ln\alpha a,\ln\beta a-\ln m-(m-1)\ln t_k+\beta t_k^{m}\}^2 \tag{15}$$

를 최소화시켜 최소자승 평가자 $\hat{\alpha}$, $\hat{\beta}$, $\hat{m}$을 구할 수 있다. 또, α, β, m은 (t_k, w_k)형태의 n개 관찰 데이터쌍으로부터 결정한다.

소프트웨어 고장은 시스템 내에 잔존하고 있는 소프트웨어의 결함에 의해서 프로그램 동작이 제대로 되지 않는 것을 말한다. 웨이블 테스트 노력을 적용한 평균치 함수를 다음과 같이 정의한다.

$$m(t)=a(1-\exp[-rW(t)]) \tag{16}$$

소프트웨어 결함검출 현상을 통계적으로 모델링할 때에 식 (16)의 $m(t)$에 의한 NHPP에 근거하여 $N(t)$의 평균치 함수를 정의하면 웨이블 테스트 노력함수를 고려한 소프트웨어 신뢰도 성장 모델을 만들 수 있다.

$$\Pr\{N(t)=n\}=poim(n\,;\,m(t)) \tag{17}$$

NHPP 고장강도함수는 평균치 함수의 미분 형태이다.

$$\lambda(t)=dm(t)/dt=a\cdot r\cdot w(t)\cdot\exp[-rW(t)] \tag{18}$$

식 (17)로부터 $N(t)$의 제한적인 분포가 평균치 함수

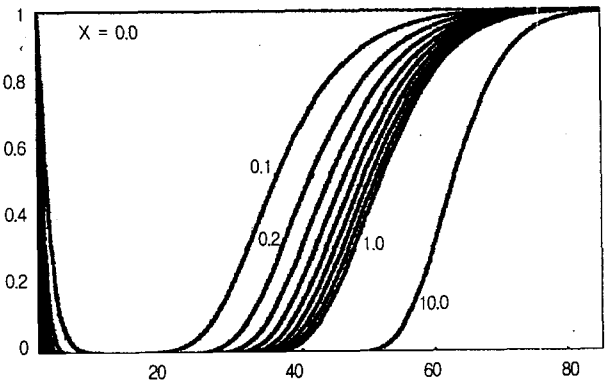
$$m(\infty)=a(1-\exp[-r\alpha]) \tag{19}$$

을 가진 포아송 분포라는 것을 보여주고 있다. 즉, 어느 소프트웨어에 평균적으로 존재하는 결함의 수는 테스트를 무한정 계속할 때 발견되는 결함의 수와 같다는 것을 의미한다.

식 (16)의 $m(t)$를 가진 NHPP 모델에 의해서 소프트웨어 신뢰도 평가에 대한 두 가지 정량적 평가 척도를 얻을 수 있다. 시각 t에서의 신뢰도는 다음과 같다.

$$R(x\mid t)=\exp\{-a(\exp[-r\cdot W(t)]-\exp[-r\cdot W(t+x)])\} \tag{20}$$

식 (20)으로 표시된 신뢰도의 특성을 이해하기 위한 테스트시간과 신뢰도의 관계는 (그림 2)와 같다.



(그림 2) 출시시각과 신뢰도 관계-웨이블 테스트 노력

이 그림에서는 최종 결함 수정 후 경과되는 각각의 시간에 대해서 출시시각과 신뢰도 성장과의 관계를 보여주고 있다. 일정한 경과시간 x에 대해서 테스트 시간 및 출시시기를 늦추면 늦출수록 신뢰도가 성장함을 알 수 있다. 또한, 비록 신뢰도가 성장하여 목표신뢰도 이상이 될 수 있으나, 결함 수정 후 경과시간이 길어지면 길어질수록 결함 발견 확률이 높아 신뢰도가 저하된다는 것도 알 수 있다. 이것은 일정 테스트 노력일 경우와 유사하다.

상기 식 (20)에서 정의한 테스트 단계의 신뢰도 의미를 고찰해 보기로 한다.

$R(x|t)$는 시각 t에서 최종적으로 결함을 발견하여 수정한 후, x 유니트의 경과시간동안 새로운 결함이 발견되지 않을 확률이다. 소프트웨어를 개발하여 결함 테스트를 하면 할수록 결함을 발견하여 수정하는 빈도가 작아지면서 신뢰도가 성장되며, 결함 수정 후 경과시간이 길어지면 길수록 결함 발견 확률이 높아지기 때문에 소프트웨어의 신뢰도는 낮아진다. 한편, 테스트 단계에서는 얼마나 오랜 시간동안 결함이 발견되지 않느냐가 중요한 것이 아니라, 현 단계에서 소프트웨어 내에서 발견되지 않고 잔존하는 결함의 수가 얼마나 되는가가 더 중요하다.

소프트웨어를 $t=T$에서 출시하는 경우, 발견되는 누적 분포는 $m(T)=a(1-e^{-rW(T)})$이므로, 잔여 분포는 $\bar{a}=a-a(1-e^{-rW(T)})=a\cdot e^{-rW(T)}$이다. 이것이 소프트웨어를 출시해서 운전하는 경우의 초기 결함수이므로

$$m(T+x)=a\cdot e^{-rW(T)}\cdot(1-e^{-rW(x)})+m(T)=a(1-e^{-rW(T)-rW(x)}) \tag{21}$$

이다. 그러므로,

$$R(x|T) = \exp[-m(T+x)+m(T)]$$
$$= \exp[-ae^{-\gamma W(T)}(1-e^{-\gamma W(x)})]$$
$$= \exp\{-ae^{-\alpha\gamma(1-e^{-\beta T^*})}[1-e^{-\alpha\gamma(1-e^{-\beta x^*})}]\} \quad (22)$$

이고, 특히 T = 0인 경우는

$$R(x|0) = \exp\{-a[1-e^{-\alpha\gamma(1-e^{-\beta x^*})}]\} \quad (23)$$

이다.

따라서, 시각 T에서 출시된 소프트웨어의 신뢰도는 경과시간 x에 대해서 지수함수적으로 감소한다. 감소비를 줄이기 위해서는 출시시각을 늦추거나 결함 검출비를 높여야 한다.

3. 출시시각 결정

3.1 일정테스트 노력

식 (9)로부터

$$T_1 = \frac{1}{b}\ln\frac{m(x)}{\ln\frac{1}{R_o}}$$

이므로

$$0 < \frac{1}{b}\ln\frac{m(x)}{\ln\frac{1}{R_o}} < T_{LC}$$

인 범위 즉,

$$1 < \frac{m(x)}{\ln\frac{1}{R_o}} < d$$

$$\ln\frac{1}{R_o} < m(x) < d\cdot\ln\frac{1}{R_o}$$

$$\ln R_o > -m(x) > d\cdot\ln R_o$$

$$R_o > e^{-m(x)} > R_o^d$$

이므로,

① $R_o > R(x|0) > R_o^d$에서 양의 유일 해 $T^* = T_1$가 존재한다.

② $R(x|0) > R_o$이면 $T_1 < 0$이므로 $T^* = T_1 = 0$이다.

③ $R(x|0) < R_o^d$이면 $T_1 > T_{LC}$이므로 $T^* = T_1 = T_{LC}$이다

①의 경우에 대해서 좀더 검토해 보기로 한다.

$$\exp(-m(x)) = R(x|0) \quad (24)$$

이므로,

$$m(x) = -\ln R(x|0) = \ln\frac{1}{R(x|0)} \quad (25)$$

으로 되어 식 (6)으로부터 T_1을 다음과 같이 구할 수 있다.

$$T^* = T_1 = \frac{1}{b}\ln\left\{\frac{\ln\frac{1}{R(x|0)}}{\ln\frac{1}{R_o}}\right\} \quad (26)$$

여기서, 편의상 $R(x|0) = t$라고 가정하고 식 (26)을 t에 관하여 두 번 미분하면

$$\frac{dT_1^2}{dt^2} = \frac{1}{b}\left\{\frac{\ln\frac{1}{t}-1}{(t\cdot\ln\frac{1}{t})^2}\right\} \quad (27)$$

이 되어 함수의 변곡점은 목표신뢰도 R_o의 값에 관계 없이 항상

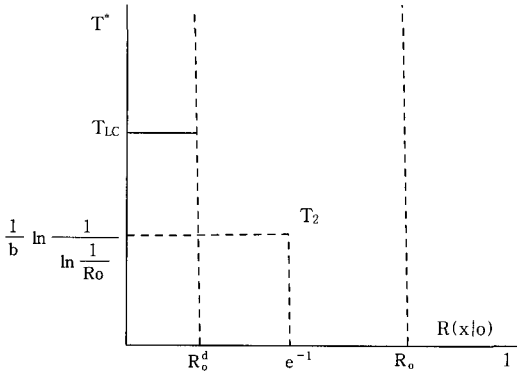
$$t = R(x|0) = \frac{1}{e} \quad (28)$$

인 값으로 결정된다.

이 때

$$T_1 = \frac{1}{b}\left\{\ln\frac{1}{\ln\frac{1}{R_o}}\right\} \quad (29)$$

이다. 이러한 관계를 (그림 3) 표시하였다.



(그림 3) 목표 신뢰도와 출시시각 관계-일정 테스트 노력

4.2 웨이블 테스트 노력

마찬가지로, 신뢰도를 요건에 최근접시키는 유일한 시각이 존재한다. 최적 소프트웨어 출시시각은 미리 규정된 소프트웨어 목표 신뢰도를 만족시키는 최근접이 되는 시각이다.

출시시각 T에서의 신뢰도는 식 (22)와 같으므로 여기서 목표 신뢰도를 만족시키는 출시시각을 구하면 다음과 같다.

목표 신뢰도를 R_o라 하면

$$\exp\{-ae^{-\alpha\gamma(1-e^{-\beta T^*})}[1-e^{-\alpha\gamma(1-e^{-\beta x^*})}]\} = R_o \quad (30)$$

이다. 식 (30)을 만족하는 출시시각을 T_2라 하면

$$T_2 = \left\{-\frac{1}{\beta}\ln\left[1+\frac{1}{\alpha\gamma}\ln\frac{\ln R_o}{\ln R(x|0)}\right]\right\}^{\frac{1}{m}} \quad (31)$$

이므로

$$0 < \left\{ -\frac{1}{\beta} \ln \left[1 + \frac{1}{\alpha\gamma} \ln \frac{\ln R_o}{\ln R(x|0)} \right] \right\}^{\frac{1}{m}} < T_{LC}$$

인 범위 즉,

① $R_o > R(x|0) > R_o^{\frac{1}{g}}$

에서 양의 유일 해 $T^* = T_2$이 존재한다.

이 값은 식 (31)로 표현된다.

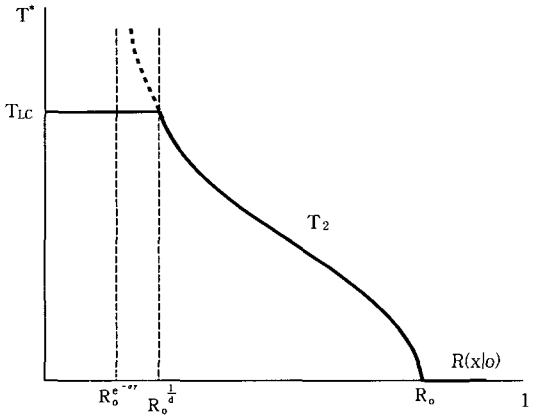
② $R(x|0) \geq R_o$

이면 $T_1 \leq 0$이므로 $T^* = T_2 = 0$이다.

③ $R(x|0) < R_o^{\frac{1}{g}}$

이면 $T_1 \geq T_{LC}$이므로 $T^* = T_2 = T_{LC}$이다.

이와 같은 내용을 (그림 4) 표시하였다.



(그림 4) 목표 신뢰도와 출시시각 관계-웨이불 테스트 노력

4. 운영 신뢰도

운영 신뢰도는 운영 단계에서 결함이 발생되지 않을 확률이다. 테스트 단계에서는 소프트웨어의 결함이 발견되는 대로 제거되어 동일 또는 유사한 결함이 다시 나타나지 않으며, 소프트웨어는 노화되지 않으므로 테스트를 계속 하면 할수록 고장발생률이 줄어든다. 반면에, 운영 기간 중에는 결함을 제거할 수 있는 경우도 있고, 그렇지 않은 경우도 있다. 보통 패키지 소프트웨어라 불리는 범용 소프트웨어의 경우는 소프트웨어를 개발하기 전에 개발자가 일반 고객의 요구를 파악하여 기호에 맞는 소프트웨어를 개발하여 불특정 다수인을 상대로 출시하는 경우이므로, 일단 출시되면 운영중에 소프트웨어의 품질에 문제가 있거나 다른 요건이 생겨도 고객의 요청을 받아들여 이를 수정하기는 현실적으로 어렵다. 운영중에 발견되는 결함은 이를 수집하여 계속 데이터베이스화 했다가 그 다음 버전 출시시 반영될 수 있으나, 이미 출시된 버전에는 적용되기 어려우므로 운영중에는 고장률이 일정한 것으로 한다.

이와는 대조적으로 고객으로부터 주문을 받아서 특수한 용도로 개발하는 전용소프트웨어 즉 주문형 소프트웨어의 경우에는 운영 중에 검출되는 결함에 대해서 고객의 요구에 의해 계속 수정을 하면서 품질을 유지 관리하기 때문에 고장률을 낮출 수 있으며, 신뢰도의 성장도 가능하다. 고객으로부터 개발을 의뢰받은 주문형 소프트웨어의 경우는 운영 중 발견되는 결함에 대한 A/S를 협약할 수 있으므로 이 때는 신뢰도식이 식 (6)이나 식 (12)와 유사하게 되는 것이 타당하다.

4.1 일정테스트 노력

평균치 함수는 다음과 같이 된다.

출시시각 T에서의 누적발생결함수는 $m_1(T) = a(1-e^{-b_1 T})$이므로, 잔여결함의 수는 $\bar{a} = a - a(1-e^{-b_1 T}) = ae^{-b_1 T}$이다. 이것이 출시후 운영중의 초기 결함수이므로, 다음과 같은 방정식을 쓸 수 있다.

$$\begin{aligned} m(t) = m_2(t) &= \bar{a}(1-e^{-b_2(t-T)}) + m_1(T) \\ &= ae^{-b_1 T}(1-e^{-b_2(t-T)}) + a(1-e^{-b_1 T}) \end{aligned} \quad (32)$$

따라서, 수명주기 기간동안 검출되는 결함의 총 수는 $m_2(T_{LC}) = ae^{-b_1 T}(1-e^{-b_2(T_{LC}-T)}) + a(1-e^{-b_1 T})$이고, 이 값에서 테스트 기간 중에 검출되는 총 결함의 수 $m_1(T) = a(1-e^{-b_1 T})$를 제외한 것이 순수하게 운전중에 검출되는 총 결함의 수이므로,

$$\begin{aligned} m_2(T_{LC}) - m_1(T) &= ae^{-b_1 T}(1-e^{-b_2(T_{LC}-T)}) \\ &\quad + a(1-e^{-b_1 T}) - a(1-e^{-b_1 T}) \\ &= ae^{-b_1 T}(1-e^{-b_2(T_{LC}-T)}) \end{aligned} \quad (33)$$

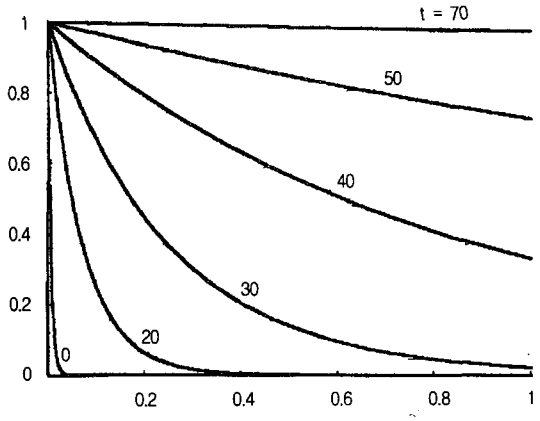
일정 테스트 노력을 적용한 전용 소프트웨어의 신뢰도를 Rc(x|T)로 표현하면 이는 테스트 단계의 신뢰도 형상과 유사하며, 그 함수는 아래와 같다.

$$\begin{aligned} R_c(x|T) &\equiv \Pr\{N(T+x) - N(T) = 0\} \\ &= \exp\{-[m(T+x) - m(T)]\} \\ &= \exp[-ae^{-b_1 T}(1-e^{-b_2 x})] \end{aligned} \quad (34)$$

식 (34)로 표현된 소프트웨어 신뢰도를 전용 소프트웨어의 운영신뢰도라 한다.

운영기간 중에는 대부분의 경우, 그것이 주문형 소프트웨어라 할지라도 테스트 기간에 비하여 소프트웨어의 결함수정이 어렵기 때문에 신뢰도 성장이 어렵다. 또한, 테스트 기간 중에는 결함을 발견하여 수정할 목적으로 테스트를 수행하지만, 운영기간 중에는 결함 없이 안정되게 소프트웨어를 사용하는 것이 목적이므로, 결함을 의도적으로 발견하려 노력하지 않는 이상 결함 발견의 기회가 많지 않다. 따라서, 결함을 수정하기란 더욱 더 어렵다. 그러므로, 일반적

으로 테스트 기간에 비하여 운영기간 중에는 경과시간에 대하여 신뢰도가 크게 저하되는 것으로 보아야 한다.



(그림 5) 경과시간과 신뢰도 관계-일정 테스트

(그림 5)에서 주어진 각각의 테스트 시간 t에 대해서 최종 결함 수정 후 경과시간 x와 신뢰도 성장과의 관계를 보여주고 있다. 일정한 테스트시간 t에 대해서 경과시간 x가 작으면 작을수록 신뢰도가 높으며, 경과시간 x가 증가함에 따라 신뢰도가 급격히 감소된다. 이 그림에서 보듯 테스트 시간이 길면 길수록 경과시간에 대한 신뢰도의 저하가 작아짐을 알 수 있다.

$R(x|t)$는 시각 t에서 최종적으로 결함을 발견하여 수정한 후 다음 경과시간 x 시간 동안 새로운 결함이 발견되지 않을 확률이다. 소프트웨어를 개발하여 결함 테스트를 하면 할수록 결함을 발견하여 수정하는 빈도가 작아지므로 신뢰도가 성장되며, 결함 수정 후 경과시간이 길어지면 질수록 결함 발견 확률이 높아지기 때문에 소프트웨어의 신뢰도는 낮아진다.

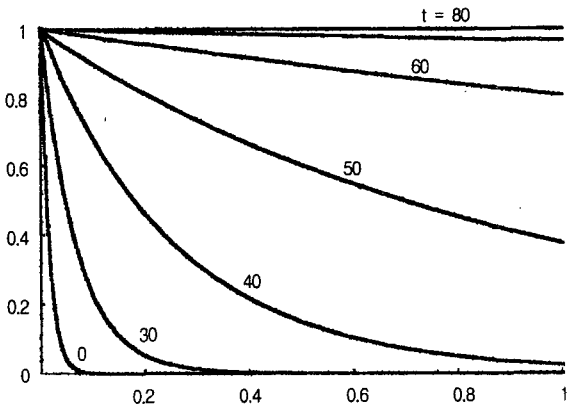
4.2 웨이불 테스트 노력

이 경우의 신뢰도식은 식 (22)와 같이 표현되므로 웨이불 테스트 노력 신뢰도를 $R_w(x|T)$라 하면 이것은

$$R_w(x|T)=\exp[-m(T+x)+m(T)]$$
$$=\exp\{-ae^{-\alpha\gamma(1-e^{-\beta T^{\gamma}})}[1-e^{-\alpha\gamma(1-e^{-\beta x^{\gamma}})}]\} \quad (35)$$

와 같이 나타낼 수 있으며, 시각 T에서 소프트웨어를 출시했을 때 경과시간에 대한 신뢰도의 변화를 (그림 6)에 나타내었다.

이 그림에서는 주어진 각각의 시험 시간 $t=T$에 대해서 최종 결함 수정 후 경과시간 x와 신뢰도 성장과의 관계를 보여주고 있다. 일정한 시험시간 t에 대해서 경과시간 x가 작으면 작을수록 신뢰도가 높으며, 경과시간 x가 증가함에 따라 신뢰도가 급격히 감소된다. 이 그림에서 보듯 시험 시간이 길면 길수록 경과시간에 대한 신뢰도의 저하가 작아짐을 알 수 있다.



(그림 6) 결함 수정후 경과시간과 신뢰도 관계

$R(x|t)$는 시각 t에서 최종적으로 결함을 발견하여 수정한 후 x 단위시간 동안 새로운 결함이 발견되지 않을 확률이다. 소프트웨어를 개발하여 결함시험을 하면 할수록 결함을 발견하여 수정하는 빈도가 작아지므로 신뢰도가 성장되며, 결함 수정 후 경과시간이 길어지면 질수록 결함 발견 확률이 높아지기 때문에 소프트웨어의 신뢰도는 낮아진다.

(그림 5)와 (그림 6)을 비교해 보면 동일한 테스트 시간 후 경과되는 시간 x에 대한 신뢰도 변화에 있어서 웨이불 곡선을 적용한 경우의 신뢰도가 더 크게 저하되는 것을 알 수 있다.

5. 결 론

하드웨어의 신뢰도와 달리 소프트웨어의 신뢰도는 결함 없이 운영되는 어느 시점부터 다음 어느 시점까지 소프트웨어가 결함 없이 운영될 확률로 정의한다.

테스트 단계에서는 테스트 공정이 NHPP를 따르는 것으로 연구되어 왔으므로 이 공정과 평균치 함수를 이용하여 어느 기간 동안 결함이 발견되지 않을 확률로 신뢰도를 정의한다. 즉, 어느 s 시각부터 $s+x$ 시각까지 새로운 결함이 발견되지 않을 확률을 신뢰도로 정의하며, 이는 평균치 함수의 시간적 차이를 지수함수의 지수로 취한 형태를 하고 있다.

운영단계에서는 일반적으로 불특정 다수의 고객이 결함 테스트 목적이 아닌 운영 목적으로 사용하다가 발견하는 결함에 의해 신뢰도가 영향을 받으므로, 결함 발견이 어려울 뿐더러, 소프트웨어의 결함을 발견하여 이를 개발자에게 알려서 수정하도록 하고, 이 수정된 것을 불특정 다수인에게 다시 반영하는 것이 현실적으로 어렵다. 따라서, 운영단계에서는 신뢰도의 성장이 쉽지 않으므로 테스트 단계의 신뢰도와 다르며, 그 신뢰도가 훨씬 저하된다. 범용 소프트웨어인 경우, 운영단계의 신뢰도는 신뢰도 함수의 지수부분이 평균치 함수의 시간적 차이가 아니라 일정한 고장강도에 경과시간을 곱한 형태를 취한다. 한편, 전용 소프트웨어인 경우는 특수목적을 가진 소프트웨어로서 고객의 요건에

맞는 소프트웨어를 개발하는 경우이므로, 운영중의 결함에 대해서 A/S 계약이 이루어지는 경우가 일반적이므로 결함 검출시 이를 보고하여 수정할 수 있다. 그러나, 이러한 경우에도 결함수정은 그리 쉽지 않다.

범용 소프트웨어와 전용 소프트웨어의 운영 단계 신뢰도를 실제 예를 가지고 분석해본 결과 전용 소프트웨어의 신뢰도가 범용 소프트웨어의 신뢰도보다 높은 것으로 계산되었다. 이는 어느 시점에서도 동일하게 적용된다.

출시시각을 결정함에 있어서 개발 후 테스트를 시작하기 전의 신뢰도 $R(x|0)$가 어떠한 조건에 있는가를 검토하여 각 조건에 따른 최적 출시시각을 결정하였다.

일정 테스트 노력 곡선인 경우, 그 조건은 $R(x|0) > R_o$, $R_o > R(x|0) > R_o^d$, $R(x|0) < R_o^d$이다. 이 중에서 이상적인 경우는 $R_o > R(x|0) > R_o^d$인 경우이다. 이 범위의 조건일 때에 최적 출시시각을 결정하고자 하는 본 연구의 목적에 부합된다. 이 경우에 대해서 최적 출시 시각이 목표 신뢰도 면에서 어떠한 경향을 보이는지 그림으로 표시하였다. 그 외의 범위에서는 비용이나 목표 신뢰도 어느 한 쪽 또는 양쪽 모두가 적절한 해법이 없거나 제시하기 어려운 경우에 속하여 최적 출시시각을 결정하기 어려우므로, 어느 한 쪽의 의도에 의해서 결정되어야만 한다.

웨이불 곡선인 경우를 고찰해 보면 $R(x|0) \geq R_o$, $R_o > R(x|0) > R_o^{\frac{1}{g}}$, $R(x|0) < R_o^{\frac{1}{g}}$이다. 이 중에서 이상적인 경우는 $R_o > R(x|0) > R_o^{\frac{1}{g}}$인 경우이다. 이 범위의 조건일 때에 최적 출시시각을 결정하고자 하는 본 연구의 목적에 부합된다. 그 외의 범위에서는 일정 테스트 노력을 적용한 경우와 마찬가지로 비용이나 목표 신뢰도 어느 한 쪽 또는 양쪽 모두가 적절한 해법이 없거나 제시하기 어려운 경우에 속하여 최적 출시시각을 결정하기 어려우므로, 어느 한 쪽의 의도에 의해서 결정되어야만 한다.

분석 결과에 의하면 일정 테스트 노력을 적용한 경우에 비하여 웨이불 테스트 노력을 적용한 경우의 운영 기간중의 신뢰도 저하가 약간 심한 편이다.

참 고 문 헌

[1] C. V. Ramamoorthy, F. B. Bastani, "Software reliability-Status and perspectives," IEEE Trans. on Software Eng., Vol.SE-8, pp.354-371, August, 1982.

[2] J. D. Musa, A. Iannino, K. Okumoto, "Software Reliability : Measurement, Prediction, Application," pp.230-238, March, 1987.

[3] S. Yamada, H. Ohtera, H. Narihisa, "Software reliability growth models with testing-efforts," IEEE Trans. on Reliability, Vol.R-35, pp.19-23, April, 1986.

[4] Sy-Yen Kuo, Chin-Yu Huang, Michael, R. Lyu, "Framework for Modeling Software Reliability, Using Various Testing Efforts and Fault-Detection Rates," IEEE Trans on Reliability, Vol.50, No.3, pp.310-320, 2001.

[5] Xiaolin Teng, Hoang Pham, "A Software-Reliability Growth Model for N-Version Programming Systems," IEEE Trans on Reliability, Vol.51, No31, pp.311-321, 2002.

[6] Shigeru Yamada, Shunji Osaki, "Software Reliability Growth Modeling : Models and Applications," IEEE Trans. on Software Eng., Vol.12, pp.1431-1437, 1985.

[7] Shigeru Yamada, Shunji Osaki, "Software Reliability Growth with a Weibull Testing-Efforts : A Models and Applications," IEEE Trans. on Reliability Vol.42, No1, pp.100-106, 1993.

[8] Shigeru Yamada, Jun Hishitani, Shunji Osaki, "Cost-Reliability Optimal Release Policies for Software Systems," IEEE Trans. on Reliability, Vol.R-34, No.5, pp.422-424, 1985.

[9] Hiroshi Ohtera, Shigeru Yamada, "Optimal Software-Release Time Considering an Error-Detection Phenominon during Operation," IEEE Trans. on Reliability, Vol.39, No.5, pp.596-599, 1990.

[10] Shigeru Yamada, Shunji Osaki, "Cost-Reliability Optimal Release Policies for Software Systems," IEEE Trans. on Reliability, Vol.R-34, No.5, pp.422-424, 1985.

[11] Goel AL, Okumoto K., "Time dependent error-detection rate model for software reliability and other performance measures," IEEE trans. on reliability, Vol.R-28, pp.206-211, 1979.

[12] Yamada, S., Ohba, M., Osaki, S., "S-shaped reliability growth modeling for software error detection," IEEE trans. on reliability, Vol.R-32, pp.475-478, 1983.

최 규 식

e-mail : che@konyang.ac.kr

1976년 서울대학교 공과대학(공학사)

1981년 뉴욕공과대학 졸업(공학석사)

1993년 명지대학교 졸업(공학박사)

1975년~1977년 동양정밀 중앙연구소

1977년~1993년 한국전력기술 중앙연구소 책임연구원

1993년~현재 건양대학교 IT학부 교수

관심분야 : 소프트웨어 공학, 초고속 무선통신