# 운영단계의 소프트웨어 신뢰도에 관한 연구

최규식*, 김용경**

# A Study on the S/W Reliability during Operational Stage

Che Gyu Shik*, Kim Yong Kyong**

Abstract

One method to improve S/W quality before releasing after development is to enhance the its reliability, whose direct methodology is to detect and revise the fault through testing. Once the S/W is released because it meets the target reliability, 곧 operational reliability arises. It is obvious the operational reliability different depending on the condition whether it is universal(package) S/W or dedicated S/W. We propose the methodology to calculate operational software reliability of universal and dedicated S/Ws in this paper.

Keywords : 소프트웨어 신뢰성

논문접수일 : 2003년 3월 31일 논문게재확정일 : 2003년 12월 5일

* 건양대학교 정보통신학과

** 건양대학교 경영정보학과

※ 이 논문은 2003학년도 건양대학교 학술연구비 지원에 의하여 이루어진 것임

## 1. 서론

최근 운영시스템, 제어프로그램, 응용프로그램과 같은 여러 가지 소프트웨어 시스템이 더욱 더 복잡화 및 대형화되고 있기 때문에 현재와 같은 정보화 사회에서 신뢰성이 높은 소프트웨어 시스템을 개발하는 일이 매우 중요하며, 따라서 소프트웨어 제품 개발에 있어서 소프트웨어 신뢰도가 핵심사항이라고 할 수 있다. 1970년대 이후 소프트웨어의 신뢰성을 향상시키기 위한 여러 가지 소프트웨어의 신뢰도 모델이 제시되고 검토되었으며, 특히, 소프트웨어 개발 후 테스트단계에 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. 그러한 결함 검출 현상을 설명해주는 소프트웨어 신뢰도 모델을 소프트웨어 신뢰도 성장 모델(SRGM : software reliability growth model)[1]이라 한다.

과거 20여년 동안 많은 SRGM이 제안되었다. 그동안 보편적으로 널리 연구되어 온 SRGM은 NHPP(nonhomogeneous Poisson process) 모델이며, 이러한 모델은 실제 발생 현상과 잘 맞아서 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 확률이다. 테스트 단계에서는 소프트웨어의 결함이 발견되는 대로 제거되어 동일 또는 유사한 결함이 다시 나타나지 않기 때문에 테스트를 반복할수록 고장발생률이 줄어든다. 반면에, 운영 기간 중에는 결함을 제거할 수 있는 경우도 있고, 그렇지 않은 경우도 있다. 보통 패키지 소프트웨어라 불리는 범용소프트웨어의 경우는 개발자가 고객의 요구를 파악하여 기호에 맞는 소프트웨어를 개발하여 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 발행하는 방법을 택하는 것이 일반적이다. 이와는 대조적으로 고객으로부터 주문을 받아서 특수한 용도로 개발하는 전용소프트웨어의 경우에는 운영중에 검출되는 결함을 반복적으로 수정하여, 품질을 유지 관리하기 때문에 신뢰도의 성장이 가능하다. 이러한 두 가지 신뢰도 개념은 다르며, 이들을 구분하는 것이 중요하므로 상황에 따라 적절히 사용해야 한다.

따라서, 본 논문에서는 지금까지 연구된 SRGM, 소프트웨어의 테스트 신뢰도 및 운영 신뢰도를 검토해보고, 특히 운영단계의 신뢰도에 있어서 범용소프트웨어의 신뢰도와 전용소프트웨어의 신뢰도를 비교 연구하고자 한다.

2장에서는 소프트웨어의 신뢰도 개념을 정의하고, 3장에서는 테스트단계 신뢰도와 운영단계 신뢰도를 분석하여 그 차이를 밝히고 실례를 통하여 어느 정도의 차이가 나는가를 검토해보고자 하였다.

**기호설명**

$m(t)$ : $E[N(t)]$, 평균치함수

$N(t)$ : 시각 $t$까지 발견되는 누적결함수

$\lambda(t)$ : 시각 $t$에서의 순간 고장강도, $\lambda(t)=dm(t)/dt$

$\lambda_r$ : 소프트웨어 발행시점의 순간고장강도

$R_o$ : 목표 소프트웨어 신뢰도레벨, $0<R<1$

$a$ : 처음부터 소프트웨어 내에 존재하고 있는 결함의 총 수

$b$, $b_1$, $b_2$ : 결함검출비, 일반적인 경우, 테스트 기간중, 운영기간중, $b_1 \geq b_2$

$T$ : 소프트웨어 발행시각

## 2. 소프트웨어의 신뢰도 개념

소프트웨어 신뢰도는 규정된 환경 하에서 주어진 기간에 소프트웨어를 결함 없이 운영할 수 있는 확률로 정의된다. S/W 고장은 시스템 내에 잔존하는 결함에 의해 프로그램 운전이 원치 않게 중지되는 것으로 정의한다. SRGM 영역에서 아래와 같은 일반적인 가정을 도입한다.

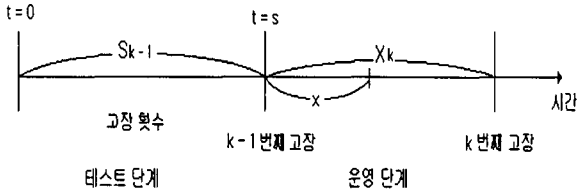
1) S/W 시스템은 S/W 결함에 의해서 무작위적으로 발생되는 S/W 고장에서 자유롭지 못하다. 즉, 언제나 S/W 고장이 발생될 수 있다.

2) S/W 고장이 발생될 때마다 이것을 일으키는 S/W의 결함을 즉시 제거하며, 새로운 결함은 도입되지 않는다.

대부분의 경우에 있어서 소프트웨어의 결함이력이 알려져 있으므로 소프트웨어 신뢰도는 다음과 같이 조건확률로 표현할 수 있다.[2]

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

이는 s 유니트 기간 동안 (k-1)번째까지의 결함을 검출하여 수정한 후 주어진 고장이력에서 다음 고장간격 $x$ 시간동안에 고장없이 소프트웨어가 동작할 확률인 것이다. [그림-1] 참조.




(그림-1) 고장발생 표현

S/W의 테스트에 의해서 검출되는 누적결함의 수의 시간종속적 동태를 기술하는데에 일반적으로 결함의 검출시간단위로서 역일시간이나 장비의 실행시간과 같은 테스트시간을 사용한다. $\{N(t), t\geq 0\}$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.

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

여기서, 평균치 함수는

$$m(t)=a[1-\exp(-bt)] \quad (3)$$

로 표현되고, $m(t)$를 다음과 같이 놓으면

$$m(t)=\int_0^t \lambda(x)dx \text{ 또는 } \lambda(t)=\frac{dm(t)}{dt} \quad (4)$$

$\lambda(t)$는 NHPP의 강도함수라 하고 순간 결함검출비를 의미한다.

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

$$\lim_{t\to\infty} \Pr\{N(t)=n\} = \frac{a^n}{n!}\exp(-a) \quad (5)$$

이는 $N(t)$가 오랜기간동안 테스트를 한 후 평균치 $a$를 가진 Poisson을 따른다는 것을 의미한다. 매우 중요한 S/W 신뢰도 성장지수로서 테스트시각 $t$에서의 단위결함당 단위시간당 결함검출비를 아래와 같이 정의한다.

$$d(t) = \lambda(t)/[a - m(t)] \quad (6)$$

공식(6)에서 $\lambda(t) = \frac{dm(t)}{dt}$ 로 하여 양변을 $t$에 관하여 적분하면,

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

$$= -\ln\frac{a-m(t)}{a},$$

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

$$m(t) = a[1-\exp(-\int_0^t d(u)du)] \quad (7)$$

## 3. 단계별 신뢰도

### 3.1 테스트 신뢰도

테스트 단계 동안에는 소프트웨어의 결함을 계속 발견하여 수정할 수 있기 때문에 테스트를 반복할수록 소프트웨어의 신뢰도가 지속적으로 성장된다. 식(1)의 시간간격 $X_k$가 소프트웨어의 테스트 단계이면 즉, 소프트웨어가 아직도 테스트중이고 테스트공정이 NHPP를 따르면 이의 표준 이론으로부터 평균치 함수를 공식(3)과 같이 정의할 때, 임의의 $t\geq 0$과 $x>0$에서

$$\Pr\{N(t+x)-N(t)=k\} = \frac{[m(t+x)-m(t)]^k}{k!} e^{\{-[m(t+x)-m(t)]\}} \quad (8)$$

이므로, 식(1)의 신뢰도는 다음과 같이 표현할 수 있다.

$$R_{te}(x|s) = \Pr\{N(s+x)-N(s)=0\}$$

$$= \exp\{-[m(s+x)-m(s)]\}$$

$$= \exp[-m(x)e^{-b_1 s}] \quad (9)$$

$R_{te}(x|s)$가 테스트 공정기간의 소프트웨어 신뢰도만을 측정할 수 있으며, 운영단계까지 연장되지 않으므로 식(9)에서 정의된 소프트웨어 신뢰도를 테스트 신뢰도로 부를 수 있다.

$R_{te}(x|s)$는 시각 $s$에서 최종적으로 결함을 발견하여 수정한 후 시각 $(s+x)$ 기간동안 새로운 결함이 발견되지 않을 확률이다. 소프트웨어를 개발하여 결함테스트를 하면 할수록 결함을 발견하여 수정하는 빈도가 커지므로 신뢰도가 성장되나, 결함 수정 후 경과시간이 길어지면 길수록 결함발견확률이 높아지기 때문에 소프트웨어의 신뢰도는 낮아진다. 즉, 결함발견 확률이 높아진다. 테스트 단계에서는 얼마나 오랜 기간동안 결함이 발견되지 않느냐가 중요한 것이 아니라, 현 단계에서 소프트웨어 내에서 발견되지 않고 잔존하는 결함의 수가 얼마나 되는가가 더 중요하다. 이 결함의 수를 예측할 수 있어야 발행시기를 결정할 수 있기 때문이다. 즉, 식(3)에 의한 잔여결함을 줄이는 것이 더 중요하다.

### 3.2 운영 신뢰도

운영 신뢰도는 운영 단계에서 결함이 발생되지 않을 확률이다. 테스트 단계에서는

소프트웨어의 결함이 발견되는 대로 제거되어 동일 또는 유사한 결함이 다시 나타나지 않기 때문에 테스트를 반복할수록 고장발생률이 줄어든다. 반면에, 운영 기간 중에는 결함을 제거할 수 있는 경우도 있고, 그렇지 않은 경우도 있다. 보통 패키지 소프트웨어라 불리는 범용소프트웨어의 경우는 소프트웨어를 개발하기 전에 개발자가 일반대중고객의 요구를 파악하여 기호에 맞는 소프트웨어를 개발하여 불특정 다수인을 상대로 출시하는 경우이므로, 일단 발행되면 운영중에 소프트웨어의 품질에 문제가 있거나 다른 요건이 생겨도 이를 수정하기는 현실적으로 어렵다. 운영중에 발견되는 결함은 이를 수집하여 계속 데이터베이스화 했다가 그 다음 버전 발행시 반영될 수 있으나, 이미 발행된 버전에는 적용되기 어려우므로 운영중에는 고장률이 일정한 것으로 한다. 이와는 대조적으로 고객으로부터 주문을 받아서 특수한 용도로 개발하는 전용소프트웨어의 경우에는 운영중에 검출되는 결함에 대해서 고객의 요구에 의해 계속 수정을 하며, 품질을 유지 관리하기 때문에 고장률을 낮출 수 있으며, 신뢰도의 성장도 가능하다.

### 3.2.1 범용소프트웨어

시간간격 $X_k$가 운영단계에 있다면 다음 고장시각은 파라미터 $\lambda_r$을 가진 지수분포를 따르며, 여기서 $\lambda_r = \lambda(s)$는 시각 s=T에서 계산된 본래 NHPP의 고장강도 함수이다. 범용소프트웨어의 신뢰도를 $R_u(x|T)$로 표현하면 그 함수는 아래와 같다.[3]

$$R_u(x|T) = \exp(-\int_0^x \lambda_r dt) = \exp[-\lambda(T)x] \tag{10}$$

즉, 운영기간 동안에는 고장강도가 일정하여 신뢰도 함수의 지수 부분이 평균치 함수의 차이가 아니라 고장강도함수에 경과시간을 곱한 값으로 된다.

### 3.2.2 전용소프트웨어

고객으로부터 개발을 의뢰받은 주문형 소프트웨어의 경우는 운영중 발견되는 결함에 대한 A/S를 협약할 수 있으므로 이 때는 신뢰도식이 식(9)와 유사하게 되는 것이 타당하다. 이 경우 평균치 함수는 다음과 같이 된다.

발행시각 $T$에서의 누적 결함수는

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

$$\begin{aligned} m(t) &= m_2(t) = \bar{a}(1-e^{-b_2(t-T)}) + m_1(T) \\ &= ae^{-b_1 T}(1-e^{-b_2(t-T)}) + a(1-e^{-b_1 T}) \end{aligned} \tag{11}$$

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 테스트기간중에 검출되는 총 결함의 수 $m_1(T) = a(1-e^{-b_1 T})$를 제외한 것이 순수하게 운전중에 검출되는 총 결함의 수이므로,

$$\begin{aligned} m_2(T_{LC}) - m_1(T) &= ae^{-b_1 T}(1-e^{-b_2(T_{LC}-T)}) \\ &\quad + a(1-e^{-b_1 T}) - a(1-e^{-b_1 T}) \\ &= ae^{-b_1 T}(1-e^{-b_2(T_{LC}-T)}) \end{aligned} \tag{12}$$

전용소프트웨어의 신뢰도를 $R_d(x|T)$로 표현하면 이는 테스트 단계의 신뢰도 형상과 유사하며, 그 함수는 아래와 같다.

$$R_d(x|s) \equiv \Pr\{N(T+x) - N(T) = 0\}$$
$$= \exp\{-[m(T+x) - m(T)]\}$$
$$= \exp[-ae^{-b_1 T}(1-e^{-b_2 x})] \quad (13)$$

식(10)으로 표현된 소프트웨어 신뢰도를 범용소프트웨어의 운영 신뢰도라 하며, 식(13)으로 표현된 소프트웨어 신뢰도를 전용 소프트웨어의 운영신뢰도라 한다. 식(10)와 식(13)은 동일하지 않으며, 어느 것을 사용하느냐의 여부에 따라서 상이한 추정을 하게 된다는 사실에 유의해야 한다.

운영기간 중에는 대부분의 경우, 그것이 주문형 소프트웨어라 할지라도 테스트기간에 비하여 소프트웨어의 결함수정이 어렵기 때문에 신뢰도 성장이 어렵다. 또한, 테스트기간 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, 일반적으로 테스트기간에 비하여 운영기간 중에는 경과시간에 대하여 신뢰도가 크게 저하되는 것으로 보아야 한다.

## 4 신뢰도 개념의 그래프 해석

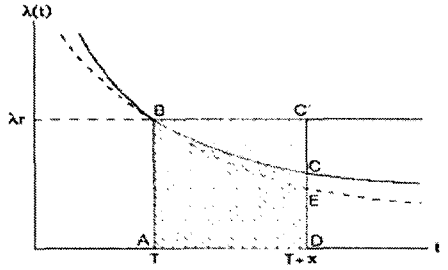
일반적으로 소프트웨어를 테스트하여 결함을 제거하게 되면 결함발견 빈도수가 줄어들기 때문에 고장강도함수가 지수함수적으로 감소하는 것으로 되어있다. 그러나, 개발 초기에는 소프트웨어에 익숙하지 못한 단계로서 불안정하여 테스트단계 동안에는 그 결함들이 감소하기 전에 결함발생빈도가 오히려 증가한 후 어느 시점부터 다시 지수함수적으로 감소되는 경우도 있다. 본 논문에서는 고장강도가 지수함수적으로 감소되는 경우가 더 일반적이므로 이에 대해서 먼저 고찰한 후, 결함발생빈도가 증가하다가 지수함수적으로 감소하는 경우에 대해서도 검토한다. 먼저 지수함수적으로 감소하는 경우를 해석한다. 이 경우의 고장강도와 시간과의 관계를 [그림-2]에 보였다. [그림-2]의 고장강도 곡선에서 처음 감소부분은 테스트기간 중의 고장강도 감소를 표현하고 $\lambda = \lambda_\gamma$인 시점부터는 발행시점에서의 고장강도이므로 범용소프트웨어인 경우 이것이 시간에 대하여 일정하게 되어 직선(실선)의 형태를 취한다. 식(10)을 다음과 같이 다시 쓸 수 있다.

$$R_u(x|T) = \exp(-\int_0^x \lambda_\gamma dt) = \exp(-\lambda_\gamma x)$$
$$= \exp(-S_{ABC'D}) \quad (14)$$

여기서, $S_{ABC'D}$는 [그림-2]의 사각형 ABC'D의 면적이다. 이와 마찬가지로 전용 소프트웨어에 대해서도 식(13)을 다음과 같이 나타낼 수 있다.

$$R_d(x|T) = \exp\{-[m(T+x) - m(T)]\}$$
$$= \exp[-ae^{-b_1 T}(1-e^{-b_2 x})]$$
$$= \exp(-S_{ABCD}) \quad (15)$$

여기서, $S_{ABCD}$는 [그림-2]의 (편의상) 곡선 사다리꼴 ABCD의 넓이를 나타낸다. 그리고, 고장강도가 테스트단계보다는 완만하게 감소할 것이므로 곡선 B-E(점선)가 아니라 곡선 B-C(실선)와 같은 형태를 취한다.




(그림-2) $\lambda(t)$가 $t \geq 0$에서 단조감소하는 경우

식(14)와 (15)로부터 두 신뢰도 개념간의 차이를 명확하게 알 수 있다. 두 가지 신뢰도의 경우를 일반적으로 다음과 같이 나타낼 수 있다.

$$R(x|s) = \exp(-S) \qquad (16)$$

여기서, $S$는 시간간격 $x$와 고장강도 곡선으로 둘러싸인 면적을 표시한다.

위의 두 가지로 정의한 신뢰도를 비교해보면, $T \geq 0$와 $x > 0$에서 주어진 어떠한 값에 대해서도 $\lambda(t)$가 $t \geq 0$에서 단조 감소하는 함수이면, $R_u(x|T) < R_d(x|T)$이다.

실제 계산 예로서 [그림-2]를 고려하여 T=800, x=100, a=33.99, $b_1=b_2=b=0.006$인 경우에 대해서 계산해보기로 한다.

$\lambda(t)$ = 33.99×0.006 exp(-0.006t) = 0.2039 exp(-0.006t), $S_{ABCD}$ = 33.99 $e^{-0.006\times800}(1-e^{-0.006\times100})$ = 0.1262,

$S_{ABC'D}$ = 0.2039 $e^{-0.006\times800}$ × 100 = 0.1678,

∴ $R_d(100|800)$ = exp(-0.1262) = 0.8814, $R_u(100|800)$ = exp(-0.1678) = 0.8455로서

$R_d(100|800) > R_u(100|800)$이다.

위의 예는 범용소프트웨어와 전용소프트웨어의 신뢰도 차이를 비교해보기 위한 단순한 예에 불과하지만 이는 어느 경우에나 유사한 결과를 얻게 된다.

따라서, 일반적으로 운영단계에서는 순간 고장강도가 범용소프트웨어인 경우, 일정하여 운영신뢰도를 성장시키지 못하며, 시간경과에 따라 신뢰도가 감소한다. 전용소프트웨어의 경우는 범용소프트웨어의 경에 비하여 신뢰도 저하가 사소한 편이다.

두 가지 신뢰도 측정에 대한 상기 고찰에서 살펴본 바와 같이 고객의 입장에서 보면 범용소프트웨어를 사용하는 고객이 전용소프트웨어를 사용하는 고객보다 훨씬 많아서 운영 단계의 범용소프트웨어신뢰도가 더 중요하다고 할 수 있다. 고객은 소프트웨어의 전 수명기간 중 운영단계에서만 소프트웨어를 사용하고, 고장공정도 테스트 단계와 달리 NHPP를 따르지 않기 때문에 운영신뢰도 정의가 고객에게 더욱 더 중요하다는 것이 명백하다. 그러나, 이러한 개념은 아직까지 많은 관심을 끌지 못하고 있다.

NHPP SRGM과 관련된 대부분의 기존 문헌에서 채택된 소프트웨어 신뢰도 측정은 테스트신뢰도이다. [4-10] 참조. 운영신뢰도 측정은 신뢰도 분석에서 고객이 필히 경험하게 되는 신뢰도와 관련되어 있음에도 불구하고 참고문헌 [11-13]과 같은 문헌에서만 일부 연구되고 있을 뿐이다.

## 5. 고장강도함수에 따른 신뢰도 비교

테스트단계 기간동안에는 평균치함수 $m(t)$가 지수함수형[4], 즉 $\lambda(t)$가 단조감소형이거나 S-형[5], 즉 $\lambda(t)$가 처음에는 증가하다가 그 다음에 감소하는 형이다. 아래에서 두 가지 경우의 결과를 요약한다.

**가.** $T \geq 0$과 $x > 0$에서 주어진 어떠한 값에 대해서도 $\lambda(t)$가 $t \geq 0$에서 단조 감소하는 함수이면, $R_u(x|T) < R_d(x|T)$이다.

이것은 제4장의 그래프적 설명으로서도 잘 알 수 있다. 사실상, $S_{ABC'D} > S_{ABCD}$이므로, 공식(14)와 (15)로부터 $R_u(x|T) < R_d(x|T)$임이 명백하다.

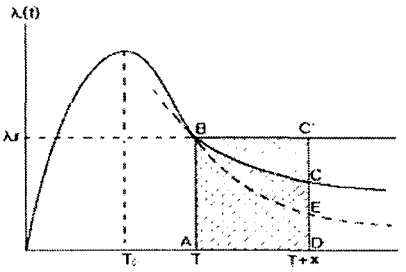
일반적으로 소프트웨어를 테스트하여 결함을 제거하게 되면 고장강도함수가 절대적으로 감소하는 것을 예상하게 된다. 반대로, 개발 초기에는 소프트웨어를 배우는 단계이고, 그래서 불안정하므로, 참고문헌[5]에 의하면 테스트 단계에서는 그 결함들이 감소하기 전에 결함 발생 빈도가 증가되는 것으로 관찰되었다. 이와 관련하여 이러한 상황 결과를 아래와 같이 정리한다.

**나.** $\lambda(t)$가 처음에 증가하다가 그 다음에 감소하는 경우, $m(t)$의 변곡점을 $T_o$로 표시하기로 하면(그림 3 참조), 여기서, $R_{op}(x|T)$는 운영단계의 신뢰도를, $R_{te}(x|T)$는 테스트 단계의 신뢰도를 의미한다.

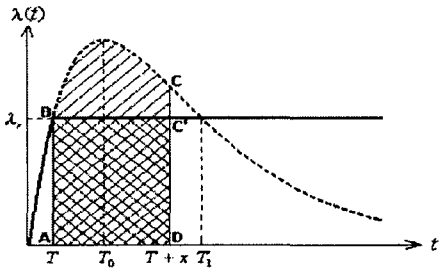
- $T_o \leq T$이면 $R_{op}(x|T) < R_{te}(x|T)$이다.(그림 3(a))
- $T_o > T$이면 $T_1$을 $\lambda(T) = \lambda(T_1)$, $T_1 > T$의 해라고 할 때 다음과 같다.

(*i*) $T_1 \geq T + x$인 경우(그림 3(b))와

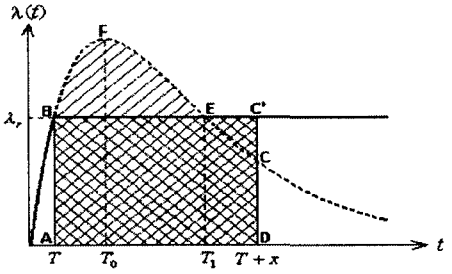
(*ii*) $T_1 < T + x$인 경우(그림 3(c))를 생각해 볼 수 있으나, 이는 현실적으로 존재할 수 없으므로 검토에서 제외한다. 이 때도 마찬가지 이유로 곡선 B-E(점선) 대신 곡선 B-C(실선)의 형태가 된다.




(a) $T \geq T_o$일 때




(b) $T < T_o$, $T + x \leq T_1$일 때




(c) $T < T_o$, $T + x > T_1$일 때

(그림 3) $\lambda(t)$가 처음에 증가하다가 감소하는 경우

대부분의 실제적인 케이스는 나의 경우이다. 고장강도 $\lambda(t)$가 나의 케이스 2에서처럼 계속 증가할 때에 소프트웨어가 발행되는 경우는 없다고 볼 수 있다. 이 케이스는 단지 완벽을 기하기 위해서 포함시킨 것이다.

## 6. 결론

하드웨어의 신뢰도와 달리 소프트웨어의 신뢰도는 결함 없이 운영되는 어느 시점부터 다음 어느 시점까지 소프트웨어가 결함없이 운영될 확률로 정의한다. 소프트웨어의 신뢰도를 평가할 때 지금까지는 테스트 단계의 신뢰도 성장과 운영단계의 신뢰도 성장이 동일한 것으로 간주하여 연구를 해 왔다. 그러나, 일반적으로 이 두 단계의 신뢰도 개념은 다르다.

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. 테스트 단계에서는 결함을 발견하기 위해 테스트를 하고 결함 발견 즉시 수정하여 동일한 결함이 다시 발생되지 않으므로 신뢰도가 크게 성장된다. 반면, 운영단계에서는 일반적으로 불특정 다수의 고객이 결함테스트 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. 범용소프트웨어인 경우, 운영단계의 신뢰도는 신뢰도 함수의 지수부분이 평균치 함수의 시간적 차이가 아니라 일정한 고장강도에 경과시간을 곱한 형태를 취한다. 한편, 전용소프트웨어인 경우는 특수목적을 가진 소프트웨어로서 고객의 요건에 맞는 소프트웨어를 개발하는 경우이므로, 운영중의 결함에 대해서 A/S 계약이 이루어지는 경우가 일반적이므로 결함검출시 이를 보고하여 수정할 수 있다. 그러나, 이러한 경우에도 결함수정은 그리 쉽지 않다.

범용소프트웨어와 전용소프트웨어의 운영단계 신뢰도를 실제 예를 가지고 분석해본 결과 전용소프트웨어의 신뢰도가 범용소프트웨어의 신뢰도보다 높은 것으로 계산되었다. 이는 어느 시점에서도 동일하게 적용된다.

## 참고문헌

[1] Shigeru Yamada, Shunji Osaki, "Software Reliability Growth Modeling : Models and Applications", IEEE Trans. on Software Eng., vol.12, 1985,12. pp1431-1437

[2] Hiroshi Ohtera, Shigeru Yamada, "Optimal Software - Release Time Considering an Error-Detection Phenomenon during Operation", IEEE Trans. on Reliability, vol.39, no.5, 1990,12. pp596-599

[3] Shigeru Yamada, Shunji Osaki, Cost-Reliability Optimal Release Policies for Software Systems", IEEE Trans. on Reliability, vol.R-34, no.5, 1985,12. pp422-424

[4] Goel AL, Okumoto K, "Time

dependent error-detection rate model for software reliability and other performance measures", IEEE trans. on reliability, 1979, R-28, pp206-211

[5] Yamada S, Ohba M, Osaki S, "S-shaped reliability growth modeling for software error detection", IEEE trans. on reliability, 1983, R-32, pp475-478

[6] Yamada S, Osaki S, "Optimal software release policy for a nonhomogeneous software error detection rate model", Microelectronics and reliabilitu, 1986,R26, pp691-702

[7] Hossain SA, Dahiya RC, "Estimating the parameters of a nonhomogeneous Poisson process model for software reliability", IEEE trans. on reliability, 1993, R-42, pp604-612

[8] Kapur PK, Bhushan S, Younes S, "An exponential SRGM with a bound on the number of failures", Microelectronics and reliability, 1993, R-33, pp1245-1249

[9] Zeephongsekul P, "Reliability growthB. Yang, M. Xie", A study of operational and testing reliability in software reliability analysis", Reliability Engineering & System Safety, 70, 2000,12. pp323-329

[10] Pham H, Zhang XM, "A software cost model with warrentcosts", IEEE trans. on computers, 1999, 48, pp71-75

[11] Musa JD,"A theory of software reliability and its application", IEEE trans. on software engineering, 1975, SE-1, pp312-327

[12] Govil KK, "A new model of software reliability prediction", Microelectrnics and reliability, 1983, 23, pp1009-1010

[13] Suresh N, Babu AJG, "Software reliability estimation and optimization, a nonhomogeneous Poisson process approach", International journal of quality and reliability management", 1997, 14, pp287-300.

## ■ 저자소개

최규식

서울대학교 공과대학 전기과를 졸업하고, 뉴욕공과대학에서 석사학위를 받았으며, 명지대학교 전기과에서 박사학위를 받았다. OPC 중앙연구소와 KOPEC 중앙연구소에서 근무하였다. 현재는 건양대학교 정보전자통신공학부 교수이며, 관심분야는 무선통신 및 소프트웨어 신뢰도이다.

김용경

고려대학교를 졸업하고, 숭실대학교에서 이학석사, 명지대학교에서 경영학 박사학위을 취득했다. 현재 건양대학교 경영정보학과 교수이며, 한국정 보기술응용학회 부회장이다.

관심분야는 소프트웨어공학, 소프트웨어 품질관리, 정보시스템 감리 및 감사 등이다.

◈ 이 논문은 2003년 3월 31일 접수하여 3차 수정을 거쳐 2003년 12월 5일 게재확정 되었습니다.