소프트웨어의 운영 및 테스트 신뢰도

최규식*
*건양대학교 정보전자통신공학부
e-mail : che@konyang.ac.kr

Operational and Testing Reliabilities of S/W

Che Gyu Shik*
*Dept of Information and Communication, Konyang University

요약

오늘날 어떠한 복잡한 기기에 있어서도 S/W 의 신뢰도는 중요한 요소가 되고 있다. S/W 신뢰도는 보통 NHPP와 같은 신뢰도모델에 의해서 평가한다. S/W 시스템은 테스트 단계에서 품질이 향상되고 운영단계에서는 거의 변화가 없다. 신뢰도를 테스트 단계에서 추정할 것인가 아니면 운영단계에서 추정할 것인가의 여부에 따라서 그 추정방법이 달라져야 한다. 본 논문에서는 두 가지의 상이한 신뢰도개념 즉 운영신뢰도와 테스트 신뢰도를 명확히 구분하여 자세히 고찰한다. 이러한 개념은 기존의 여러 문헌에서 혼용해 사용하거나 잘못사용되어 왔다. 상이한 신뢰도개념을 사용하면 상이한 신뢰도를 얻게 되며, 추후 상이한 신뢰도기준 결정기법에 이르게 된다.

1. 서론

S/W 신뢰도는 관련 SRGM으로부터 유도되어 왔다. 그러나, 여기에 두 가지의 상이한 신뢰도 개념이 있는데 즉, 테스트 신뢰도와 운영 신뢰도가 바로 그것이다. 테스트 신뢰도는 테스트 단계에서 에러가 발생하지 않을 확률이다. 운영 신뢰도는 운영 단계에서 에러가 발생하지 않을 확률이다. 테스트 단계에서는 소프트웨어결함이 발견되는 대로 제거되어 유사한 에러가 다시 나타나지 않기 때문에 고장발생율은 테스트 시간 여부에 달려있다. 반면에, 운영테스트기간 중에는 결함을 제거하는 것이 테스트 단계와 달리 결함을 제거할 수 없으므로, 사용자가 사용하는 전 기간에 걸쳐서 일정한 고장발생율을 겪게 될 수도 있다. 이러한 두가지 신뢰도개념은 명백히 다르며, 이들을 구분 짓는데 중요한 사항이 되어 상황에 따라 적절히 사용해야 한다.

2항에서는 테스트신뢰도와 운영신뢰도를 구분하고 3항에서는 이 두 신뢰도 개념을 분석적으로 비교하였다. 4항에서는 최적발행시각을 결정함에 있어서 상이한 신뢰도개념에 대한 영향을 고찰하였다. 그 결과를 예시하기 위해서 5항에서 수치예제를 제안하였다.

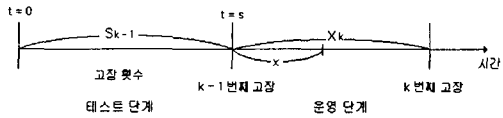


그림 1 고장횟수와 신뢰도와의 관계

2. 테스트 및 운영신뢰도 개념

S/W 신뢰도는 규정된 환경 하에서 주어진 기간에 대한 무결함 소프트웨어운영확률인 것으로 정의한다.[5] 대부분의 경우에 있어서 소프트웨어의 결함이력이 알려져 있으므로 S/W 신뢰도는 다음과 같이 조건확률로 표현할 수 있다.

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

이는 s 유니트기간동안 주어진 고장이력의 다음 고장간격 기간 동안의 x기간에 대한 신뢰도를 표현한다. 시각 t까지의 누적고장수를 $N(t)$로 표현하고 $\{N(t);t\geq 0\}$ 평균치함수 $m(t)$를 가진 NHPP로 모델화할 수 있다. 본 항에서는 두 개의 신뢰도 개념을 정의하고 이들의 차이를 추정신뢰도로 고찰하고자 한다.

2.1 테스트 신뢰도

테스트 단계 동안에는 소프트웨어가 개선된다. 공식 (1)의 시간간격 X_k가 소프트웨어의 테스트단계이면 즉, 소프트웨어가 아직도 테스트중이고 테스트공정이 NHPP를 따른다면 NHPP의 표준이론으로부터 평균치 함수를 $m(t)=a(1-e^{-bt})$로 정의할 때 임의의 $t\geq 0$과 $x>0$에서

$$R_{te}(x|s)\equiv \Pr\{N(s+x)-N(s)=0\} = \exp\{-[m(s+x)-m(s)]\} \quad (2)$$

그런데, 이 신뢰도는 운영신뢰도에 비하여 그렇게 중요한 의미를 가지고 있지 못하다. 테스트 단계에서는 소프트웨어를 실제 사용할 때의 결함에 의한 소프트웨어 기능 이상을 우려하는 것이 아니라, 결함을 될수록 많이 검출하여 잔여결함의 수를 줄임으로써 소프트웨어가 운용될 경우에 오류발생을 줄이기 위한 것이기 때문이다. 그러므로, 테스트 단계에서는 어떤 x 기간동안 결함 없이 잘 운용할 수 있느냐가 문제가 아니라, 운용시에 신뢰성 있게 잘 사용될 수 있도록 잔여 결함의 수를 줄이는 것이 중요하다.

2.2 운영신뢰도

소프트웨어를 s단위의 시간동안 계속 테스트하여 고객에게 발행되었다면, 신뢰도함수는 아래와 같다.

$$R_{op}(x|s)=\exp\left(-\int_0^x \lambda_f dt\right)=\exp[-\lambda(s)x] \quad (3)$$

공식(3)에서 정의된 소프트웨어 신뢰도를 운영신뢰도라 부르며, 이는 운영단계기간의 소프트웨어 신뢰도를 측정한다. 즉, x는 운영단계에서의 경과시간을 나타낸다. 공식(2)와 (3)은 동일하지 않으며, 어느 것을 사용하느냐의 여부에 따라서 상이한 추정을 하게 된다는 사실에 유의해야 한다.

2.3 신뢰도 개념의 그래프적 표현

공식(2)를 다음과 같이 다시 쓸 수 있다는 사실을 유의해야 한다.

$$R_{te}(x|s)=\exp\{-[m(s+x)-m(s)]\} = \exp\left[-\int_s^{s+x}\lambda(t)dt\right]=\exp(-S_{ABCD}) \quad (4)$$

여기서, S_{ABCD}는 그림 2의 곡선 사다리꼴 ABCD의 면적이다. 이와 마찬가지로 공식(3)도 다음과 같이 나타낼 수 있다.

$$R_{op}(x|s)=\exp[-\lambda(s)x]=\exp(-S_{ABC'D}) \quad (5)$$

여기서, $S_{ABC'D}$는그림 1의 사각형 ABC'D의 넓이를 나타낸다.

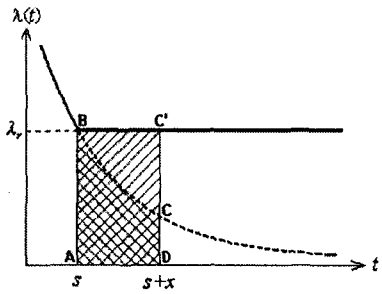


그림 2. $\lambda(t)$가 $t\geq 0$에서 단조감소하는 경우

공식(4)와 (5)로부터 두 신뢰도 개념간의 차이를 명확하게 알 수 있다. 사실상, 두 신뢰도 정의를 아래와 같이 표현할 수 있다.

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

여기서, S는 시간간격 x와 고장강도곡선으로 둘러싸인 면적을 표시한다. 테스트 신뢰도인 경우, 소프트웨어를 테스트함에 따라, 고장강도 $\lambda(t)$가 시간에 따라 변하기 때문에 둘러싸인 면적은 곡선 사다리꼴 ABCD이다. 그러나, 운영신뢰도인 경우, 소프트웨어가 시각 s에서 고객에게 발행되었으므로 고장강도는 운영기간동안 변하지 않고 그것이 발행될 시점의 값과 동일한 값 λ_f를 유지한다. 그러므로, 둘러싸인 넓이는 직사각형 ABC'D이다.

2.4 현재 통용되는 신뢰도 정의

두 가지 신뢰도 측정에 대한 상기 고찰에서 살펴본 바와 같이 고객의 입장에서 보면 운영신뢰도가 테스트 신뢰도보다 훨씬 중요하다는 것은 의심할 여지가 없다. 고객은 소프트웨어를 항상 소프트웨어의 전 수명기간 중 운영단계에서 사용하고, 고장공정도 테스트 단계와 달리 NHPP를 따르지 않기 때문에 운영신뢰도정의가 고객에게 더욱 더 중요하다는 것이 명백하다. 그러나, 이러한 개념은 많은 관심을 끌지 못하고 있다.

NHPP SRGM에 관련된 대부분의 기존 문헌에서 채택된 소프트웨어 신뢰도 측정은 테스트신뢰도이다. 참고문헌[9-15] 참조. 운영신뢰도 측정은 신뢰도 분석에서 고객이 필히 경험하게 되는 신뢰도와 관련되어 있음에도 불구하고 참고문헌 [6-8]와 같은 문헌에서만 일부 연구되고 있을 뿐이다.

3. 두 신뢰도 정의의 비교

운영 신뢰도 개념과 테스트 신뢰도 개념이 다르므로 실제적인 차이를 연구하여 그 특성을 비교하는 것이 중요하다. 테스트단계 기간동안에는 평균치함수 $m(t)$가 지수함수형[9], 즉 $\lambda(t)$가 단조감소형이거나 S-형[10], 즉 $\lambda(t)$가 처음에는 증가하다가 그 다음에 감소하는 형이다. 아래의 정리에서 두 가지 경우의 결과를 요약한다.

정리 1.1 $T\geq 0$와 $x>0$에서 주어진 어떠한 값에 대해서도 $\lambda(t)$가 $t\geq 0$에서 단조 감소하는 함수이면, $R_{op}(x|T)<R_{te}(x|T)$이다.

일반적으로 소프트웨어를 테스트하여 결함을 제거하게 되면 고장강도함수가 절대적으로 감소하는 것을 예상하게 된다. 반대로, 개발 초기에는 소프트웨어를 배우는 단계이고 또한 그래서 불안정하므로, 참고문헌[10]에 의하면 테스트 단계에서는 그 결함들이 감소하기 전에 결함발생빈도가 증가되는 것으로 관찰되었다. 정리 1.2에서 이러한 상황 결과를 정리하였다.

정리 1.2 $\lambda(t)$가 처음에 증가하다가 그 다음에 감소하는 경우, $m(t)$의 변곡점을 T_o로 표시하기로 하면 (그림 3 참조),

케이스 1 : $T_o\leq T$이면 $R_{op}(x|T)<R_{te}(x|T)$이다.(그림 3(a))

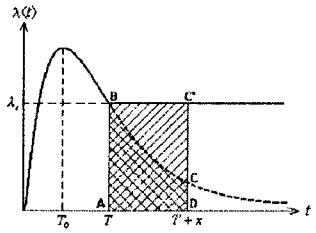
케이스 2 : $T_o>T$이면 T_1을 $\lambda(T)=\lambda(T_1)$, $T_1>T$의 해라고 할 때 다음과 같다.

(i) $T_1\geq T+x$이면 $R_{op}(x|T)>R_{te}(x|T)$이다. (그림 3(b))

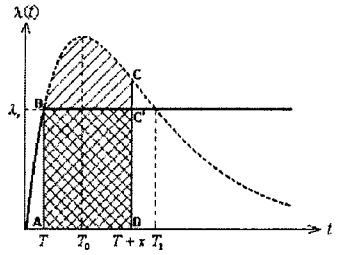
(ii) $T_1<T+x$이면 (그림 3(c))

$M<0$일 때 $R_{op}(x|T)R_{te}(x|T)$
$M=0$일 때 $R_{op}(x|T)=R_{te}(x|T)$
$M>0$일 때 $R_{op}(x|T)<R_{te}(x|T)$

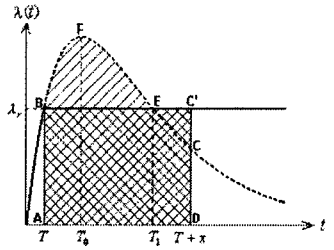
이며, 여기서, $M\equiv\lambda_1 x-m(T+x)+m(T)=S_{ECC}-S_{BFE}$ 이다.(그림 3(c) 참조)



(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

4. 최적소프트웨어 발행시각에 대한 영향

본 항에서는 최적 소프트웨어 발행에 대한 상이한 신뢰도개념을 도입할 때의 영향을 고찰하고자 한다. 이는 소프트웨어 신뢰도모델 적용에 이것이 매우 중요하기 때문이다.

소프트웨어 개발에서 중요한 과제는 소프트웨어의 최적발행시각을 결정하는 일이다. 소프트웨어에서 성취해야 할 필수요건이 바로 최저 신뢰도 레벨이므로, 상이한 소프트웨어 신뢰도 정의를 사용하게 되면 산출된 최적 발행시각에 영향을 미칠 것이다. 그러므로, 그 상이점을 조사하여 좀더 적절한 신뢰도 개념을 적용하는 것이 필요하다.

최적 발행시각 문제는 일반적으로 다음과 같이 공식화한다.

$R(x|T)\geq R_o$ (7)

인 조건하에서

$C(T)$ (6)

를 최소화한다. 대부분의 문헌에서는 공식(7)에 적용하는 소프트웨어 신뢰도가 테스트신뢰도이다. 즉,

$\exp\{-[m(T+x)-m(T)]\}\geq R_o$ (8)

사실상, 소프트웨어가 T의 시간 단위동안 테스트되

어 고객에게 발행되므로 운영기간중에는 신뢰도 성장이 없다. 그러므로, 신뢰도를 운영신뢰도 요건으로 언급하는 것이 더욱 더 적절하다. 즉,

$$\exp[-\lambda(T)x] \geq R_o \qquad (9)$$

공식(6)과 (8)에 있는 최적 발행시각 문제는 아래와 같이 이후 P1으로 언급하고 공식(6)과 (9)에 있는 최적 발행시각 문제는 P2로 언급하기로 한다.

정리 2.1 $t \geq 0$에서 $\lambda(t)$가 절대적으로 감소하면,

케이스 1 : *If* $R_{op}(x|0) \geq R_o$, *then* $T^*_{P1} = T^*_{P2} = T_C$

케이스 2 : *If* $R_{te}(x|0) \geq R_o > R_{op}(x|0)$, *then* $T^*_{P1} = T_C$ and $T^*_{P2} = \max\{T_C, T^2_R\}$

케이스 3 :

If $R_{te}(x|0) < R_o$, *then* $T^*_{P1} = \max\{T_C, T^1_R\}$ and $T^*_{P2} = \max\{T_C, T^2_R\}$

이 때,

(a) *If* $T_C \geq T^2_R$, *then* $T^*_{P1} = T^*_{P2} = T_C$

(b) *If* $T^1_R < T_C < T^2_R$, *then* $T^*_{P1} = T_C < T^*_{P2} = T^2_R$

(c) *If* $T_C \leq T^1_R$, *then* $T^*_{P1} = T^1_R < T^*_{P2} = T^2_R$

정리 2.2 $\lambda(t)$가 처음에 증가하다가 감소하며, $T_o \leq T$이면

케이스 1 : $R_{op}(x|T_o) \geq R_o$이면

$T^*_{P1} = T^*_{P2} = \max\{T_C, T_o\}$이다.

케이스 2 :

$R_{te}(x|T_o) \geq R_o > R_{op}(x|T_o)$, *then* $T^*_{P1} = \max\{T_C, T_O\}$, $T^*_{P2} = \max\{T_C, T^2_R\}$

이고,

(a) *If* $T_C \geq T^2_R$, *then* $T^*_{P1} = T^*_{P2} = T_C$

(b) *If* $T_o < T_C < T^2_R$, *then* $T^*_{P1} = T_C < T^*_{P2} = T^2_R$

(c) *If* $T_C \leq T_o$, *then* $T^*_{P1} = T_o < T^*_{P2} = T^2_R$

케이스 3 :

If $R_{te}(x|T_o) < R_o$, *then* $T^*_{P1} = \max\{T_C, T^1_R\}$ and $T^*_{P2} = \max\{T_C, T^2_R\}$

그리고,

(a) *If* $T_C \geq T^2_R$, *then* $T^*_{P1} = T^*_{P2} = T_C$

(b) *If* $T^1_R < T_C < T^2_R$, *then* $T^*_{P1} = T_C < T^*_{P2} = T^2_R$

(c) *If* $T_C \leq T^1_R$, *then* $T^*_{P1} = T^1_R < T^*_{P2} = T^2_R$

정리 2로부터 언제나 $T^*_{P1} \leq T^*_{P2}$의 관계가 성립하는 것이 분명해졌다. 이는, 최적발행시간에 있어서 테스트 신뢰도를 택하면 좀더 짧은 T^*를 얻을 수 있다는 것을 말해준다. 이는 요구되는 테스트시간에 대해서 과도한 최적 추정에 이르게 하는 것이 되어 소프트웨어가 요구되는 신뢰도 레벨에 이르기 전에 소프트웨어를 부적절하게 발행하는 것이 된다. 사실상, 고객에게 의미가 있는 것은 운영신뢰도이므로, 최적 발행문제는 P1보다 P2로 공식화해야 한다.

5. 결론

소프트웨어시스템은 일단 발행되면 신뢰도가 더 이상 개선되지 않으므로 고장 패턴과 행위에 변화가 있게 된다. 본 논문에서는 두 가지 소프트웨어 신뢰도 개념 즉, 상이한 단계에서 유용한 테스트신뢰도와 운영신뢰도을 구분하여 연구하였다. 그 두 개의 신뢰도 개념은 서로 다르므로 주의하여 사용해야 한다.

최적 소프트웨어 발행 시각에 상이한 개념을 적용한 영향을 조사하였다. 테스트신뢰도개념을 사용하게 되면 부정확한 결과가 초래되어 결정방법론을 그르치게 된다. 고객에게 중요한 것은 운용신뢰도이므로, 소프트웨어 발행시각 문제와 기타 다른 관련 결정론 법적인 공정에 운용신뢰도개념을 채택해야 한다.

실례로서 우리는 G-O모델을 사용하였지만, 다른 모델에 대해서도 유사한 결과가 나타날 것으로 예상된다. 그 외에도, 소프트웨어 신뢰도 모델을 적용하는데 있어서 발행시각을 결정하는 것이 중요한 문제이므로, 여기에서 자세하게 고려하였다. 상이한 개념을 사용하면 다른 신뢰도 관련 양이나 신뢰도 관련 결정사항에 유사한 상이점이 예상된다.

감사의 글
본 연구는 한국과학재단 목적기초연구 (R01-2000-00273) 지원으로 수행되었음