운영중 고장률을 고려할 경우의 신뢰도

최규식

Software Reliability, Considering Failure Rate during Operation

Gyu Shik Che

요약

그동안 소프트웨어의 신뢰도가 테스트중은 물론 운영중에도 고장을 검출 및 수벙함으로써 성장될 수 있다는 가정 하에 SRGM이 연구되어왔다. 한편, 어떤 논문에서는 운영중에 소프트웨어를 수정한다는 것이 특히 범용 소프트웨어인 경우 불가능에 가깝기 때문에 테스트노력이 일정한 것으로 가정하기도 하였다.
. 저자는 소프트웨어의 신뢰도 현상에 접근할 수 있는 단순한 기법을 제안하여 기존신뢰도 모델을 수정하지 않고 고장률을 줄일 수 있도록 하는 방안을 제시한다.

1. 서론

개발 소프트웨어의 테스트기간이 길면 길수록 소프트웨어의 결함수가 더 줄어들기 때문에 늦게 발행하면 신뢰도는 향상될 것이나, 그에 따른 비용이 증가되고 적절한 발행시기를 놓칠 우려가 있으며, 반대로 너무 빨리 발행하면 개발비용은 감소하나 잦은 결함으로 인하여 A/S 비용이 증가되고 고객에게 불편과 불만을 주게 된다.

본 논문에서는 운영단계에 있는 소프트웨어의 신뢰도를 고려함에 있어서 그간의 연구에서 제시되어온 신뢰도 모델을 수정하지 않아도 운영중에 고장률이 향상된다는 이러한 신뢰도 현상을 모델화하는 단순한 접근법을 제안한다.

2. 운영중 소프트웨어의 신뢰도 및 고장률

2.1 운영단계 신뢰도

소프트웨어 신뢰도는 규정된 환경 하에서 주어진 기간에 소프트웨어를 결함 없이 운영할 수 있는 확률인 것으로 정의한다. 대부분의 경우에 있어서 소프트웨어의 결함이력이 알려져 있으므로 소프트웨어 신뢰도는 다음과 같이 조건확률로 표현한다.[6]

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

여기서, s는 소프트웨어 개발 후 테스트 기간을 거쳐 제품을 인도하게 되는 시각이고, x는 제품 인도 후 소프트웨어를 사용하는 경과시간이며 X_k, S_k는 각각의 확률변수이다. 이는 s라는 유니트 기간 동안 주어진 고장이력에서 다음 고장간격 x 시간동안에 고장 없이 소프트웨어가 동작할 확률인 것이다.

운영신뢰도의 경우, 지금까지의 연구를 검토해보면 소프트웨어가 발행된 뒤에도 A/S가 가능한 것으로 가정하여 발행시기를 결정했다.[2, 7] 그러나, MS OS, 혼글 word 등과 같이 개발자가 불특정 다수의 사용자를 대상으로 개발하여 발행하는 경우에는 A/S가 어려워 그 이상 신뢰도의 성장이 없다. 운영중에 발견되는 결함은 계속 데이터베이스화 했다가 그 다음 버전 발행시 반영될 수 있으며, 이미 발행된 버전에는 적용되기 어려우므로 고장율이 일정한 것으로 한다. 즉, 시간간격 X_k가 운영단계에 있다면 다음 고장시각은 파라미터 λ_r을 가진 지수분포를 따르며, 여기서 $\lambda_r = \lambda(s)$는 시각 s에서 계산된 본래 NHPP의 고장강도 함수이다. 운영중 신뢰도 함수는 아래와 같다

$$R_o(x|s) = \exp(-\int_0^x \lambda_r dt) = \exp[-\lambda(s)x] \quad (2.2)$$

개발 초기에는 소프트웨어에 익숙하지 못한 단계로서 불안정하여 테스트단계 동안에는 그 결함들이 감소하기 전에 결함발생 빈도가 오히려 증가한 후 어느 시점부터 다시 지수함수적으로 감소되는 경우도 있다. 본 논문에서는 고장강도가 지수함수적으로 감소되는 경우가 더 일반적이므로 이에 대해서만 고찰한다. 이 경우의 고장강도와 시간과의 관계를 그림 2-2에 보였다. 그림 2-2의 고장강도 곡선에서 처음 감소부분은 테스트 기간 중의 고장강도 감소를 표현하고 고장강도 $\lambda = \lambda_s$인 시점부터는 발행시점에서의 고장강도이므로 이것이 시간에 대하여 일정하게 되어 직선(실선)의 형태를 취한다. 테스트 기간중의 신뢰도는 다음과 같이 쓴다.

$$R_t(x|s) = \exp\{-[m(s+x) - m(s)]\}$$
$$= \exp[-\int_s^{s+x} \lambda(t)dt] = \exp(-S_{ABCD}) \quad (2.3)$$

S_{ABCD}는 그림 2-2의 곡선 사다리꼴 ABCD의 면적이다. 이와 같은 방법으로 운영기간중의 신뢰도를 나타내는 식(2.2)도 다음과 같이 나타낼 수 있다.

$$R_o(x|s) = \exp(-\int_0^x \lambda_r dt) = \exp[-\lambda(s)x]$$

(2.4)

여기서, $S_{ABC'D}$는 그림 2-1의 사각형 ABC′D의 넓이를 나타낸다.

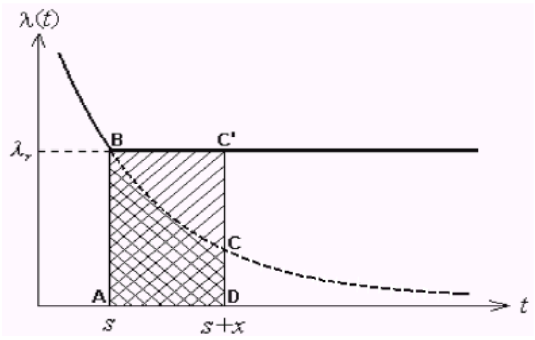

그림 2-1 $\lambda(t)$가 $t \geq 0$에서 단조감소하는 경우

두 가지 신뢰도의 경우를 일반적으로 다음과 같이 나타낸다.

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

(2.5)

여기서, S는 시간간격 x와 고장강도 곡선으로 둘러싸인 면적을 표시한다. 위의 두 가지로 정의한 신뢰도를 비교해보면, 인도시간 $s = T \geq 0$와 경과시간 $x > 0$에서 주어진 어떠한 값에 대해서도 $\lambda(t)$가 $t \geq 0$에서 단조감소하는 함수이면,

$$R_o(x|T) < R_t(x|T)$$

(2.6)

이다. 이는 어느 경우에나 유사한 결과를 얻게 된다.

2.2 시간경과에 따른 고장률 저감

제품 유니트에 있어서 그 제품이 판매된 t 개월 동안 경험된 고장률을 다음과 같이 모델화한다.

$$\lambda(t) = \lambda_0 \cdot \alpha^t + \lambda_f \ , \qquad 0 < \alpha < 1$$

(2.7)

여기서, (λ_0+λ_f)는 제품이 인도되었을 경우의 초기 고장률이며, α는 감쇠정수, λ_f는 최종 정상 상태 고장률이다. 처음 소프트웨어 제품이 설치되면 잉여 과도 고장률 λ_0 가 있다는 것을 의미한다. 이 과도 고장률은 매월 감쇠정수 α 에 의하여 감쇠된다. 몇 달 후에 이 과도 고장률은 0으로 접근하여 사용자는 λ_f 라는 고장률에 접하게 된다. 이러한 현상을 그림 2-2에 표시하였다. 그림에서 위의 수평선은 디버깅을 수행하지 않는 경우의 운영중 소프트웨어 고장률을 나타낸다.

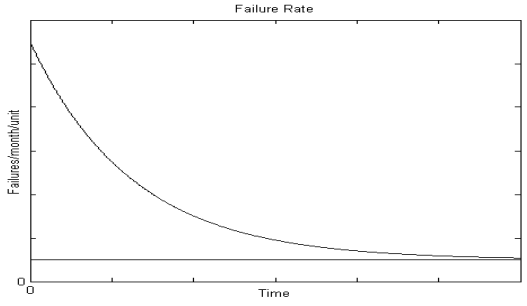

그림 2-2 유니트의 고장률

제품에 있어서 새로운 유니트들이 계속하여 인도되므로, 현장에 설치된 유니트들이 시간 간격의 어느 점에서도 그들의 시간 경력에 따라 상이한 고장률을 경험하게 된다. 따라서, 제품에 의하여 경험하게 되는 총 고장은 단순히 한 유니트의 고장률에 운영중인 전체 유니트수를 곱한 것이 아니라, 매 기간중의 인도되는 소프트웨어의 고장률에 인도수량을 곱하여 각각을 합한 값이다.

2.3 파라미터 산출법

실제 데이터가 수집되었을 경우 여기에 적합한 함수를 구하기 위하여 파라미터를 결정해야 한다. 여기에는 두 가지의 대중적인 산출기법으로서 최대가능성 평가자(MLE)와 최소자승법 평가자(LSE)가 있다. MLE는 한 집합의 연립 방정식을 풀어서 파라미터를 산출하며, s-신뢰 구간을 구동하는데 더 좋은 방법이다. LSE는 실제로 관찰/획득한 것과 예측한 것 사이의 차이를 제곱해서 더한 총 값을 최소화하는 것이다. LSE는 중간 크기의 표본에 최적인 것으로 알려지고 있으며, 최적점을 산출할 수 있게 해준다. 본 논문에서는 LSE에 의해서 실제 현상에 적합한 파라미터를 구하도록 한다. 최저 제곱합에 대한 산출공식 SSE(a, λ_0, λ_f)는 다음과 같다.

$$SSE = \sum_{k=1}^{n} [\lambda - \lambda_k]^2$$

(2.8)

이 방법에 의하여 파라미터를 구한다. 고장률이 방정식(2.7)의 형태로 주어진 경우에 대해서 이 방법을 적용해보면 다음과 같다.

$$SSE(\alpha, \lambda_0, \lambda_f) = \sum_{k=1}^{n} [(\lambda_0 \alpha^k + \lambda_f) - \lambda_k]^2$$

(2.9)

여기서, a,는 감쇠인자, (λ_0+λ_f) 초기고장률, λ_f는 최종적인 정상상태 고장률이다.

이 식을 a, λ_0, λ_f에 관하여 각각 편미분하여 그 편미분치를 0으로 놓고, 이 항들을 재정비하여 비선형 최소자승 문제를 푼다.

2.4 운영중 신뢰도

본 논문에서는 방정식(2.7)에 의한 고장률을 적용한 운영중 신뢰도 함수를 다음과 같이 제안한다.

$$R_p(x|s) = \exp\left[-\int_s^{s+x} \lambda(t)dt\right]$$

$$= \exp\left[-\int_s^{s+x} (\lambda_0 \alpha^t + \lambda_f)dt\right]$$

(2.10)

방정식(2.3)에서

$$R_t(x|s) = \exp\{-[m(s+x) - m(s)]\} = \exp[-ae^{-bs}(1-e^{-bx})]$$

$$= \exp[-m(x)e^{-bs}](2.19)$$

이며, 여기서, a는 초기결함의 수, b는 결함검출비이다.

방정식(2.4)에서

$$R_0(x|s) = \exp[-\lambda_r(s)x] = e^{-\lambda_r x}$$

(2.11)

로서 운영신뢰도가 시간경과에 대해서 지수함수적으로 단조감소하며, 발행시간 s와 무관하다.

$$R_p(x|s) = \exp\left[-\int_s^{s+x}\lambda(t)dt\right] = \exp[-\int_s^{s+x}(\lambda_0\alpha^t+\lambda_f)dt]$$

$$= \exp[\frac{\lambda_0\alpha^s}{\log\alpha}(1-\alpha^x)-\lambda_f x]$$

(2.12)

와 같이 된다.

3. 총 고장의 모델링

이전 항에서는 유니트의 고장률이 어떤 시간 함수인가를 기술하였다. 그러므로, 제품이 시각 t까지 인도된 후 그 모든 월 동안 판매된 유니트를 안다면 현장에서 경험하여 겪는 고장의 총수를 평가할 수 있다. 이는 상이한 이력을 가진 유니트의 수를 알기 때문에 가능하며, 상기 모델을 가지고 상이한 이력의 유니트에 대한 고장률을 결정할 수 있다.

판매 및 인도되어 운영중인 소프트웨어의 총 고장률은 매 기간 판매되는 총 유니트수 중에서 고장난 유니트의 수를 말한다. 따라서, 본 논문에서 총고장률을 다음과 같이 정의한다. 방정식(2.3)에 의한 테스트 중 총고장률

$$F_t(t) = \frac{\sum_{i=1}^{t}\lambda_i N_i}{\sum_{i=1}^{t} N_i}$$

(3.1)

방정식(2.4)에 의한 운영중 총고장률(고장률이 일정하다고 가정)

$$F_0(t) = \frac{\sum_{i=1}^{t}\lambda_i N_i}{\sum_{i=1}^{t} N_i} = \frac{\lambda_r\sum_{i=1}^{t} N_i}{\sum_{i=1}^{t} N_i} = \lambda_r$$

(3.2)

운영중 총고장률(고장률이 감소한다고 가정)

$$F_t(t) = \frac{\sum_{i=1}^{t}\lambda_i N_i}{\sum_{i=1}^{t} N_i}$$

(3.3)

매월 판매되는 총유니트수를 첫 번째 월로부터 시작하여 N_1, N_2, N_3,....,N_t라 한다. 시간에 대한 고장률 감쇠를 가진 모델을 사용하여 t월에서의 총 고장 F_t를 구하는 다음 방정식을 얻는다.[8]

$$F_1 = \lambda_0 N_1 + \lambda_f N_1 = (\lambda_0+\lambda_f)N_1$$

$$F_2 = \lambda_0(\alpha N_1 + N_2) + \lambda_f(N_1+N_2)$$

$$F_3 = \lambda_0 N_3 + \lambda_0 N_2\alpha + \lambda_0 N_1\alpha^2 + \lambda_f(N_1+N_2+N_3)$$

$$= \lambda_0(\alpha^2 N_1 + \alpha N_2 + N_3) + \lambda_f(N_1+N_2+N_3)$$

.....

$$F_t = \lambda_0(\alpha^{t-1}N_1 + \alpha^{t-2}N_2 + \cdots + N_t) + \lambda_f(N_1+N_2+\cdots+N_t)$$

이 되고 이 방정식을 정리하면

$$F_t = F_{t-1}\alpha + \lambda_0 N_t +$$

$$\lambda_f(N_t + (1-\alpha)(N_1+N_2+\cdots+N_{t-1}))$$

$$= \alpha F_{t-1} + \lambda_f(N_1+N_2+\cdots N_t)$$

$$-\alpha\lambda_f(N_1+N_2\cdots+N_{t-1}) + \lambda_0 N_t \qquad (3.5)$$

이 된다. 그러므로, 우리의 고장률 감쇠 모델을 사용하여 매월 관찰되는 총 고장, 매월 판매 유니트의 수, 모델 파라미터와 관련된 이러한 방정식을 유도할 수 있다.

매월 현장에서 수집되는 데이터를 통하여, 그리고 월간 판매 정보를 통하여 통계적인 기법을 이용하여 모델의 파라미터를 산출할 수 있다. 달리 말해서 수개월 동안에 걸친 실제 고장률 데이타와 판매 유니트의 수로부터 3개의 파라미터를 결정할 수 있다. - 정상상태 고장률 λ_f, 최초고장률 λ_0, 그리고 감쇠인자 a이다. 이는 방정식(2.8)-(2.9)를 풀어서 구할 수 있다.

4. 실례

4.1 고장강도

제품의 고장률을 나타내기 위해 이러한 접근법을 사용하는 것은 고장률이 단지 소프트웨어 내의 결함의 수에만 의존하며, 시스템을 다루는 사람들의 경험 레벨과 같은 다른 요인과는 관계가 없는 것으로 가정하는 것이다. 달리 말한다면 고장률이 소프트웨어 제품에 대해서 일정한 것으로 가정하고 있다.

그러나, 소프트웨어 제품에서 소프트웨어를 변경시키지 않고 그냥 사용하는 경우에도 시간이 경과함에 따라 고장률이 감소하는 현상이 계속 관찰되고 있다. 접근법에서 전체 총 고장률의 예측치와 현장에서 실제 발견된 값의 차이를 제곱해서 합한 총합이 최소가 되도록 하는 파라미터를 구할 때까지 수치적으로 계산하여 그 파라미터를 찾는다. 이 방법을 이용하여 아래와 같은 값을 구했다.

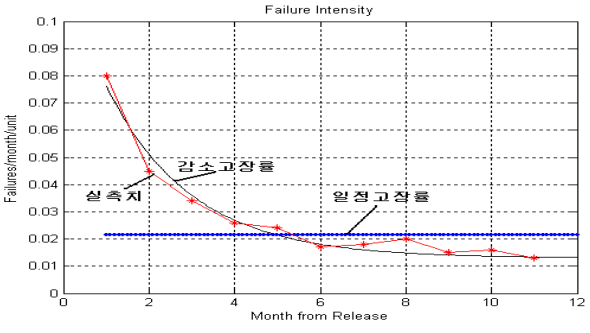

그림 4-1 제품의 총 고장률

초기과도고장률 λ_0 = 0.105 고장/월, 정상상태고장률 λ_f = 0.014 고장/월, 감쇠인자 a = 0.603. 이로부터 제품군의 정상상태 고장률이 월간 0.014이고 이는 2개월째 정상적인 방법으로 구한 값의 약 26.8%이며, 이는 총 고장을 총 판매대수로 나눈 값이다. 그리고, 4개월째 평균고장률의 약 50.2%이다. 6개월째 고장률과 비교하더라도 정상상태 고장률이 평균 고장률의

약 73.5%이다. 이 예에서 평균고장률 계산이 제품의 신뢰도를 제공한다. 평균치가 정상상태 신뢰도보다 훨씬 높다. 이 예에서는 과도 고장률의 감쇠인자가 대단히 높다는 것도 보여주고 있다. 다른 말로 말하면 사용자가 2-3개월 안에 정상상태 고장률로 접근하며, 일반적으로 첫 달은 최악이다.

한편, 참고문헌[3]에서 제시한 식(2.2), (2.4)에 의하여 발행 후 고장률이 λ_r=0.02141로 일정하다고 가정한 경우에는 발행 후 시간 경과에 관계 없이 그 값이 일정하여 그림 4-1의 수평선 형태를 취한다. 그림에서 보듯 실제 현상과는 많은 차이가 있어서 비교가 쉽지 않다.

이 모델이 얼마나 실제 데이타에 근접한 값을 나타내는지 검토해 보기로 한다. 우리는 방정식(3.1), (3.2)에 의하여 예측된 고장을 구하고 그 다음 실제 고장 데이타를 따라서 그렸으며, 그 결과를 그림 4-2에 나타냈다.

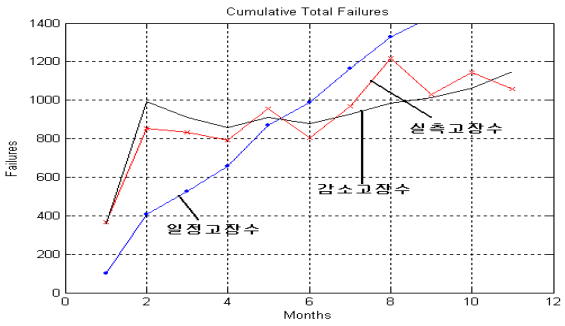

그림 4-2 예측 및 실제의 총 고장

4.2 파라미터의 적합성 검토

구한 파라미터의 적합성을 평가하기 위하여 제시한 5개의 성능 비교 기준중에서 SSE와 표준편차를 구하여 비교하기로 한다.

감소고장률인 경우, SSE=1.02×10^{-4}이고 표준편차는 0.003이다. 고정고장률인 경우, SSE=4.35×10-3이고 표준편차는 0.02로서 감소고장률을 적용한 경우가 실제현상에 가깝다는 것을 알 수 있다.

4.3 신뢰도

동일한 조건 하에서 상기에서 구한 데이터를 이용하여 각각의 신뢰도를 구하여 비교한 결과는 그림 4-3과 같다. 여기서 테스발행 전 테스트 시간 s=10이다.

이 그림에서 보는 바와 같이 발행 초기에는 두 가지 경우 모두 신뢰도가 100%이나 시간이 경과함에 따라 고정고장률을 적용한 경우가 급격하게 낮아짐을 볼 수 있다. 고정고장률을 적용한 경우 발행전 테스트 시간에 해당되는 s=10이 되면 그 신뢰도가 80%로 낮아지며, 좀더 시간이 지나 s=33을 경과하면 50% 이하로 낮아져서 제품의 신뢰도를 보증할 수 없게 된다.

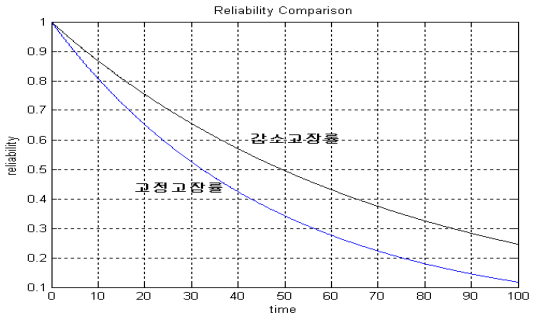

그림 4-3 판매 후 신뢰도 비교

5. 결론

테스트 단계에서는 테스트 공정이 NHPP를 따르는 것으로 연구되어 왔으므로 이 공정과 평균치 함수를 이용하여 어느 기간 동안 결함이 발견되지 않을 확률로 신뢰도를 정의한다. 반면, 운영단계에서는 일반적으로 불특정 다수의 사용자가 사용하다가 발견하는 결함에 의해 신뢰도가 영향을 받으므로, 사용자가 사용중인 소프트웨어의 결함을 발견하여 이를 개발자에게 알려서 수정하도록 하고 수정된 것을 불특정 다수인에게 다시 반영하는 것이 현실적으로 어렵다. 따라서, 운영단계에서는 신뢰도의 성장이 가능하지 않으므로 테스트 단계의 신뢰도와 다르며, 그 신뢰도가 훨씬 저하된다. 운영단계의 신뢰도는 신뢰도 함수의 지수부분이 평균치 함수의 시간적 차이가 아니라 일정한 고장강도에 경과시간을 곱한 형태를 취한다.

본 논문에서는 이러한 현상을 나타내기 위한 간단한 모델을 제시하였다. 초기 제품 사용자는 과도 고장률을 경험하게 되며, 매월 a 인자로 감소한다. 결국, 이러한 과도 고장률은 0으로 근접하며, 그러면 사용자는 재품의 정상상태 고장률을 겪게 되며 이 때 우리는 소프트웨어의 진정한 신뢰도를 나타내는 것으로 고려한다.

이러한 모델을 이용하여 매월 판매되는 유니트의 총수와 매월 관찰되는 고장의 총 수로부터 3개의 모델파라미터 초기 과도 고장률, 정상 상태 고장률, 감쇠 인자를 결정할 수 있다.

참고문헌

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

[2] 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

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

[4] Chillarege, S. Biyani, J. Rosenthal, "Measurement of failure rate in widely distributed software", Proc. 25th Fault Tolerant Computing Symposium, FTCS-25, 1995,pp424-433
[5] S. Kan, D. Manlove, B. Gintowt, "Measuring system availability - field performance and in-process metrics", ISSRE 2003, Supplementary Proceedings, pp189-199
[6] Hiroshi Ohtera, Shigeru Yamada, "Optimal Software - Release Time Considering an Error-Detection Phenominon during Operation", IEEE Trans. on Reliability, vol.39, no.5, 1990,12. pp596-599
[7] Min Xie, "On the Determination of Optimum Software Release Time", IEEE, 1991, pp218-224
[8] P. Jalote, B. Murphy, "Reliability Growth in Software Products", Proced. of 15th Intern. Symposium on RSE, 2004, pp1-7
[9] Alan P. Wood, "Software Reliability from the Customer View", Computing Practice, 2003.8, pp37-42