운영중 소프트웨어의 고장률에 의한 신뢰도 비교 연구

최 규 식* · 문 명 호** · 전 인 오***

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

Gyu Shik Che* · Myung Ho Moon** · In Oh Jeon

Abstract

The SRGM has been studied under the assumption that S/W reliability can grow as the fault causing failure is removed even during operational phase because the debugging is available. On the other hand, some papers insist on the uniform failure rate during operational phase because the debugging may not be available in case of universal software. The phenomenon, however, has been observed informally many times that the products S/W reliability grows as the time goes by even without any debugging in point of customer view.

I propose the simple approaching method to model the S/W reliability phenomenon that the failure rate reduces as time goes on without modifying the existing reliability model in this paper.

Keywords : SRGM, Operatinal S/W Reliability, Failure Reduction, Total Failure Modeling

논문접수일 : 2005년 6월 9일 논문게재확정일 : 2006년 1월 17일

* 건양대학교 의공학과 교수, (320-711)충남 논산시 내동 26번지, Tel : 041-730-5283, e-mail : che@konyang.ac.kr
** 건양대학교 의공학과 부교수
*** 벤처전문대학원 정보경영학과 교수

1. 서 론

소프트웨어가 주어진 시간 간격 동안 고장이 발생하지 않을 확률 즉, 신뢰도는 소프트웨어의 테스트 과정을 계속해서 반복 및 수정하면 더욱 더 향상된다. 그러한 결함 검출 현상을 설명해 주는 소프트웨어 신뢰도 모델을 소프트웨어 신뢰도 성장 모델(SRGM)[1]이라 한다.

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.

그간의 연구[2]에서는 테스트기간 및 운영기간의 고장에 의해서 디버깅을 하고, 디버깅에 의해서 신뢰도가 성장된다는 이론에 근거하여 또 다른 연구를 하였다. 즉, 운영기간중에도 디버깅에 의해서 소프트웨어의 신뢰도가 성장할 수 있다는 것이다. 반면에, [3]에서는 전용 소프트웨어를 제외한 일반적인 범용 소프트웨어의 경우, 현실적으로 운영기간 중에는 고장발생률이 줄어들지 않아서 소프트웨어의 신뢰도가 시간의 경과에 따라 저하되는 것으로 결론지었다. 그러나, 고객의 입장에서 소프트웨어 제품에서 소프트웨어를 변경시키지 않고 그냥 사용하는 경우에도 시간이 경과함에 따라 고장률이 감소하는 현상이 계속 관찰되고 있다. 즉, 소프트웨어 제품에 있어서 결함 제거를 하지 않아도 동일한 경과시간에 대하여 신뢰도 성장이 이루어지는 현상을 나타내는 것이다. 이러한 현상은 많은 연구가들에 의하여 비공식적으로 검토되어 왔으며, 다른 문헌[4, 5]에서도 이를 지적하고 있다. 제품들에 대한 이러한 신뢰도 성장이 그간의 대부분의 신뢰도 성장 모델들과 다르다는 것이 명백하며, 이는 단지 결함 제거 뿐만 아니라 여러 가지 인자들에 의한 것이기 때문이다. 운영기간 중에 신뢰도가 향상되는 이유 중의 하나는 사용기간이 경과함에 따라 사용자가 그 사용법을 점차 익히게 되어 고장을 일으킬 만한 상황, 명령어, 행위를 피할 수 있기 때문이다. 이렇게 함으로써 사용자는 반복적으로 나타날 수 있는 고장 경험을 방지할 수 있어서, 무작위적이고 예측 불가능한 고장을 제외한 모든 경우에 대비할 있다. 따라서, 초기에 매우 높은 고장률을 경험한 후에 소프트웨어 제품이 정상상태 고장률에 이르게 된다. 이러한 현상은 제품 사업에 있어서 많은 사람들에 의하여 비공식적으로 인용되어 왔다. 또 다른 인자는 소프트웨어의 형상에 의한 신뢰도의 문제이다. 기존의 시스템에 새로운 소프트웨어를 설치하면 새로운 소프트웨어가 현재 사용되는 기계 시스템의 작업과 맞지 않기 때문이다. 따라서 사용자는 새로운 제품이 적절하게 작동될 수 있도록 형상을 맞추기 위해 드라이버와 어플리케이션을 업그레이드 시킨다.

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.

다음 2항에서는 시간경과에 따른 소프트웨어의 운영신뢰도와 고장 변화를 연구하고, 각 경우의 파라미터를 구하는 방법에 대해서 검토하며, 고장률이 시간 경과에 따라 어떻게 변하는지를 모델화하는 것에 대해서 검토한다. 3항에서는 고장률 모델과 월간 판매를 사용하여 총 예측고장을 어떻게 결정할 수 있는가를 고찰한다. 4항에서는 실제의 예를 들어서 고장률을 예측하고 총고장을 산출하여 실제와 얼마나 유사한가를 검토하며, 각각의 신뢰도를 비교한다.

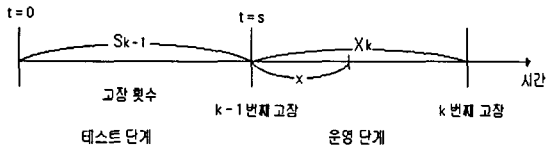
2. 운영중 소프트웨어의 신뢰도 및 고장률

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].

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

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(<그림 2-1> 참조).



〈그림 2-1〉 고장발생 표현

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[3].

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

즉, 운영기간 동안에는 신뢰도 함수의 지수부분이 평균치 함수의 차이가 아니라 고장강도 함수에 경과 시간을 곱한 값으로 된다. 따라서, 운영기간 중에는 대부분의 경우, 소프트웨어의 결함 수정이 쉽지 않기 때문에 고장강도가 일정하여 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 않아 결함을 수정하기란 더욱 더 어렵다. 그러므로, 일반적으로 테스트 기간에 비하여 운영기간 중에는 디버깅이 어렵기 때문에 신뢰도 성장이 쉽지 않아 경과시간에 대하여 신뢰도가 크게 저하되는 것으로 본다.

그러나, 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 2-2>에 보였다. <그림 2-2>의 고장강도 곡선에서 처음 감소부분은 테스트 기간 중의 고장강

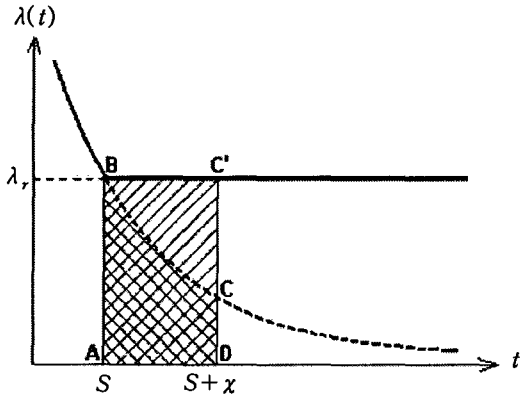
도 감소를 표현하고 고장강도 $\lambda=\lambda_f$인 시점부터는 발행시점에서의 고장강도이므로 이것이 시간에 대하여 일정하게 되어 직선(실선)의 형태를 취한다. 테스트 기간중의 신뢰도는 다음과 같이 쓴다.

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

여기서, m(t)는 평균치 함수이고, S_{ABCD}는 <그림 2-2>의 곡선 사다리꼴 ABCD의 면적이다. 이와 같은 방법으로 운영기간중의 신뢰도를 나타내는 식 (2.2)도 다음과 같이 나타낼 수 있다.

$$R_o(x|s)=\exp(-\int_0^x\lambda_f dt)=\exp[-\lambda(s)x] \quad (2.4)$$

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



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

두 가지 신뢰도의 경우를 일반적으로 다음과 같이 나타낸다.

$$R(x|s)=\exp(-S) \quad (2.5)$$

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 단조 감소하는 함수이면,

$$R_o(x|T)<R_t(x|T) \quad (2.6)$$

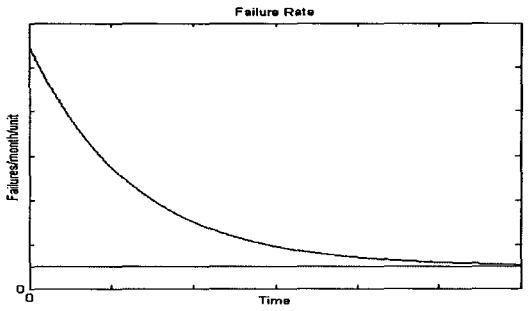
이다. 이는 어느 경우에나 유사한 결과를 얻게 된다. 따라서, 일반적으로 운영단계에서는 순간 고장강도가 일정하여 운영신뢰도를 성장시키지 못하며, 시간 경과에 따라 신뢰도가 감소하므로 테스트 단계의 신뢰도보다 저하된다.

2.2 시간경과에 따른 고장률 저감

한편, 제품을 인도하였을 경우, 사용자인 고객의 입장에서 보면 제품의 소프트웨어 유니트에서 경험된 고장률은 인도 초기 몇 달 동안 매우 높으며, 시간이 경과함에 따라 점차 감소한다는 사실이 밝혀지고 있다. 이러한 현상은 실제 경험을 통하여 수집된 데이터에 의해서 입증되고 있다. 이러한 현상은 실제 현장에서 수집된 고장관련 데이터에 의해서 증명되고 있다[8, 9]. 이 경우 제품 유니트에 있어서 그 제품이 판매된 t개월 동안 경험된 고장률을 다음과 같이 모델화한다.

$$\lambda(t)=\lambda_0\cdot\alpha^t+\lambda_f,\ 0<\alpha<1 \quad (2.7)$$

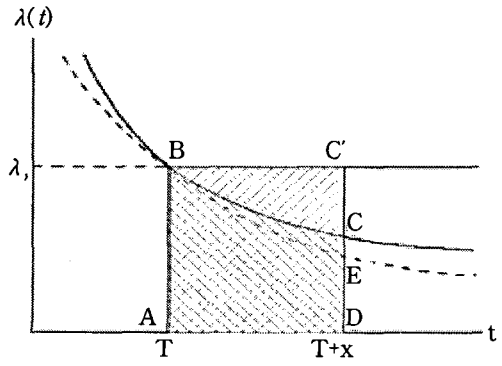
여기서, ($\lambda_0+\lambda_f$)는 제품이 인도되었을 경우의 초기 고장률이며, α는 감쇠정수, λ_f는 최종 정상상태 고장률이다. 처음 소프트웨어 제품이 설치되면 잉여 과도 고장률 λ_0가 있다는 것을 의미한다. 이 과도 고장률은 매월 감쇠정수 α에 의하여 감쇠된다. 몇 달 후에 이 과도 고장률은 0으로 접근하여 사용자는 λ_f라는 고장률에 접하게 된다. 이러한 현상을 <그림 2-3>에 표시하였다. 그림에서 위의 수평선은 디버깅을 수행하지 않는 경우의 운영중 소프트웨어 고장률을 나타낸다.



〈그림 2-3〉 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.

<그림 2-4>의 고장강도 곡선에서 $t=0$에서 $t=T$까지의 감소부분은 테스트 기간 중의 고장강도 감소를 표현하고 $\lambda=\lambda_f$인 시점부터는 인도 시점에서의 고장강도이다. 제품 인도 후에도 디버깅이 가능하다면 고장률이 곡선 B-E를 따르게 되나 이는 현실적으로 어려우므로 곡선 B-C를 따르게 된다. 그간의 논문[3]에서는 직선 B-C'를 따르는 것으로 가정하였었다.



〈그림 2-4〉 $\lambda(t)$가 $t\geq 0$에서 단조감소하는 경우

운영신뢰도 측정은 신뢰도 분석에서 고객이 필히 경험하게 되는 신뢰도와 관련되어 있음에도 불구하고 참고문헌[10-12]와 같은 문헌에서만 일부 연구되고 있을 뿐이다.

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[13]. MLE는 한 집합의 연립방정식을 풀어서 파라미터를 산출하며, s-신뢰구간을 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(α, λ_0, λ_f)는 다음과 같다.

$$SSE=\sum_{k=1}^{n}[\lambda-\lambda_k]^2 \qquad (2.8)$$

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

$$SSE(\alpha,\lambda_0,\lambda_f)=\sum_{k=1}^{n}[(\lambda_0\alpha^k+\lambda_f)-\lambda_k]^2 \qquad (2.9)$$

여기서, α,는 감쇠인자, ($\lambda_0+\lambda_f$) 초기고장률, λ_f는 최종적인 정상상태 고장률이다.
이 식을 α, λ_0, λ_f에 관하여 각각 편미분하여 그 편미분치를 0으로 놓고, 이 항들을 재정비하여 비선형 최소자승 문제를 푼다.

$$\frac{\partial SSE}{\partial \alpha} = \sum_{k=1}^{n} \{\lambda_0 \alpha^k + \lambda_f - \lambda_k\} k\alpha^{k-1} = 0 \quad (2.10)$$

$$\frac{\partial SSE}{\partial \lambda_0} = \sum_{k=1}^{n} \{\lambda_0 \alpha^k + \lambda_f - \lambda_k\} \alpha^k = 0 \quad (2.11)$$

$$\frac{\partial SSE}{\partial \lambda_f} = \sum_{k=1}^{n} \{\lambda_0 \alpha^k + \lambda_f - \lambda_k\} = 0 \quad (2.12)$$

이 세 가지 방정식을 만족하는 계수를 구하기는 쉽지 않으므로 수치해석적인 방법으로 구한다.

구한 파라미터가 얼마나 실제에 적합한가를 검토하려면 5개의 성능비교기준을 검토한다[13].

1) 평가의 정확도 $AE = \left| \frac{M_a - a}{M_a} \right|$ (2.13)

Ma= 테스트후의 검출되는 결함의 실제적인 누적수

a = 초기결함의 추정치

실제적으로 사용하기 위한 목적으로 Ma는 소프트웨어 테스트후 소프트웨어 결함 추적 과정으로부터 얻는다. 이 값이 작을수록 실제에 가까운 추정이 된다.

2) 상대오차 $RE = \frac{m(t_q) - q}{q}$ (2.14)

$m(t_q)$는 시간 t_q에서의 평균치함수이며, q는 그 시간까지 발견되는 고장의 수이다.다. 테스트시간 tq에서 q개의 고장이 관찰되면 파라미터를 산출하기 위해 te(te<tq)까지의 고장 데이타를 사용한다. 그 산출을 실제수 q와 비교한다. 여러 가지 te에 대해서 이 절차를 반복한다. tq에 대한 RE를 그려봄으로써 그 예측의 정당성을 점검한다.

3) 잡음 $Noise = \sum_{i=1}^{n} \left| \frac{r_i - r_{i-1}}{r_{i-1}} \right|$ (2.15)

ri = 예측 고장율

모델 예측 거동에서 그 값이 작으면 잡음이 적은 것을 의미하며, 곡선이 매우 원활함을 나타낸다. 잡음척도 ∞는 모델이 0의 고장율을 가짐을 의미한다.

4) 제곱평균 $MSF = \frac{1}{k} \sum_{i=1}^{k} [m(t_i) - m_i]^2$ (2.16)

여기서, $m(t)$는 평균치함수이다. MSF는 장기 예측에 대한 정량적 비교에 쓰인다. 실제와 예측치 사이의 차이의 척도를 좀더 이해하기 쉽도록 하기 때문이다. MSF의 값이 작으면 적합성 오차가 적고 성능이 양호하다는 것을 나타낸다.

5) 표준편차 $Var = \sqrt{\frac{\sum_{k=1}^{n} (\lambda_k - \lambda)^2}{n}}$ (2.17)

이 표준편차는 추정한 값이 분포의 평균값에서 얼마나 벗어나 있는지를 평가하는 기준으로 사용한다.

2.4 운영중 신뢰도

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

$$R_p(x|s) = \exp\left[- \int_s^{s+x} \lambda(t) dt \right] = \exp\left[- \int_s^{s+x} (\lambda_0 \alpha^t + \lambda_f) dt \right] \quad (2.18)$$

이는 <그림 2-4>에서 보는 운영중 실선으로 표시된 고장강도를 이용한 것이다.

제안된 방정식과 기존의 방정식(2.3), (2.4)에 의한 방법을 비교한다.

방정식 (2.3)에서

$$R_t(x|s) = \exp\{-[m(s+x) - m(s)]\} = \exp[-ae^{-bs}(1 - e^{-bx})] = \exp[-m(x)e^{-bs}] \quad (2.19)$$

이며, 여기서, a는 초기결함의 수, b는 결함검출비이다.
방정식 (2.4)에서

$$R_0(x|s) = \exp[-\lambda_r(s)x] = e^{-\lambda_r x} \qquad (2.20)$$

로서 운영신뢰도가 시간경과에 대해서 지수함수적으로 단조감소하며, 발행시간 s와 무관하다.
방정식 (2.18)에서

$$\begin{aligned} R_p(x|s) &= \exp\left[-\int_s^{s+x}\lambda(t)dt\right] \\ &= \exp[-\int_s^{s+x}(\lambda_0\alpha^t+\lambda_f)dt] \\ &= \exp\left[\frac{\lambda_0\alpha^s}{\log\alpha}(1-\alpha^x)-\lambda_f x\right] \qquad (2.21) \end{aligned}$$

와 같이 된다.

3. 총 고장의 모델링

이전 항에서는 유니트의 고장률이 어떤 시간함수인가를 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.

$$F(t) = \frac{\text{(최초 인도 후 시간 } t\text{까지 발견된고장 유니트의수)}}{\text{(최초 인도 후 시간 } t\text{까지의 총운영대수)}} \qquad (3.1)$$

이 정의에 의하여 제안된 각각의 경우에 해당하는 고장률을 검토해보면 다음과 같다.

방정식 (2.3)에 의한 테스트중 총고장률

$$F_t(t) = \frac{\sum_{i=1}^{t}\lambda_i N_i}{\sum_{i=1}^{t} N_i} \qquad (3.2)$$

방정식 (2.4)에 의한 운영중 총고장률(고장률이 일정하다고 가정)

$$F_0(t) = \frac{\sum_{i=1}^{t}\lambda_i N_i}{\sum_{i=1}^{t} N_i} = \frac{\lambda_r\sum_{i=1}^{t} N_i}{\sum_{i=1}^{t} N_i} = \lambda_r \qquad (3.3)$$

방정식 (2.17)에 의한 운영중 총고장률(고장률이 감소한다고 가정)

$$F_t(t) = \frac{\sum_{i=1}^{t}\lambda_i N_i}{\sum_{i=1}^{t} N_i} \qquad (3.4)$$

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

$$\begin{aligned} F_1 &= \lambda_0 N_1 + \lambda_f N_1 = (\lambda_0+\lambda_f)N_1 \\ F_2 &= \lambda_0(\alpha N_1 + N_2) + \lambda_f(N_1+N_2) \\ F_3 &= \lambda_0 N_3 + \lambda_0 N_2\alpha + \lambda_0 N_1\alpha^2 + \lambda_f(N_1+N_2+N_3) \\ &= \lambda_0(\alpha^2 N_1 + \alpha N_2 + N_3) + \lambda_f(N_1+N_2+N_3) \\ &\cdots\cdots\cdots \\ F_t &= \lambda_0(\alpha^{t-1}N_1 + \alpha^{t-2}N_2 + \cdots + N_t) \\ &\quad + \lambda_f(N_1+N_2+\cdots+N_t) \end{aligned}$$

이 되고 이 방정식을 정리하면

$$F_t = F_{t-1}\alpha + \lambda_0 N_t + \lambda_f(N_t + (1-\alpha)(N_1 + N_2 + \cdots + N_{t-1})) = \alpha F_{t-1} + \lambda_f(N_1 + N_2 + \cdots N_t) - \alpha\lambda_f(N_1 + N_2 \cdots + N_{t-1}) + \lambda_0 N_t \quad (3.5)$$

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 α이다. 이는 방정식 (2.8)-(2.9)를 풀어서 구할 수 있다.

4. 실 례

상기 모델을 그간의 제품 데이타에 적용해보기로 한다. 참고문헌[8]에서의 데이터를 <표 4-1>에 정리하였다. 이 데이타로부터 총 유니트 수, 총 고장이 알려져 있으므로 총 고장률을 계산할 수 있으며, 이 총 고장률은 이미 제시하였다.

4.1 고장강도

일반적인 목적의 소프트웨어 제품에 대해서는 일단 그 제품이 불특정 다수의 사용자에게 인도되면 여러 가지 다양한 데이터 수집과정을 통하여 최종 사용자로부터 대량의 고장 데이타를 포획할 수 있으므로 제품 인도 직후에 신뢰도를 정확하게 측정하고자 할 때 좀더 유연성이 있다. 측정 신뢰도에 대해서 생각해 보면 과거에 사용되던 단순한 방법으로서 운영중인 모든 유니트에서 경험된 고장 총수를 결정하여 현장에서 사용되는 유니트수로 나누는 것이었다. 우리가 N개의 유니트를 설치했다고 하면, 그리고 시간 간격 T동안 F개의 고장이 발생했다고 하면 소프트웨어의 고장률은 $\lambda = F/(N*T)$[14]로 계산할 수 있다.

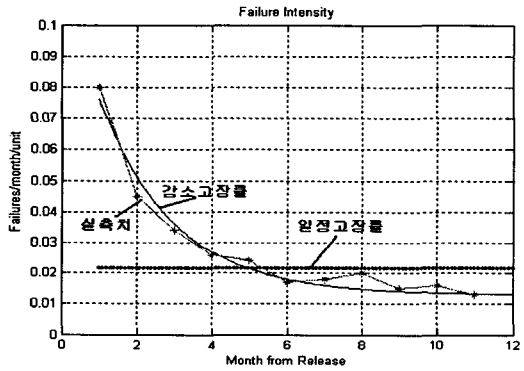
제품의 고장률을 나타내기 위해 이러한 접근법을 사용하는 것은 고장률이 단지 소프트웨어 내의 결함의 수에만 의존하며, 시스템을 다루는 사람들의 경험 레벨과 같은 다른 요인과는 관계가 없는 것으로 가정하는 것이다. 달리 말한다면 고장률이 소프트웨어 제품에 대해서 일정한 것으로 가정하고 있다.

그러나, 소프트웨어 제품에서 소프트웨어를 변경시키지 않고 그냥 사용하는 경우에도 시간이 경과함에 따라 고장률이 감소하는 현상이 계속 관찰되고 있다. 예를 들어 보고된 고장에 의하여 결정된 실제 제품의 사용 시간에 대한 전체적인 고장률과 판매된 총 유니트수의 관계를 <그림 4-1>에서 설명한다. 이는 실제 제품의 고장과 갯수 데이타에 근거를 두고 있다.

〈표 4-1〉 제품에 대한 고장 및 판매 데이타

월	고장수	누적고장수	월간판매량	누적판매량	고장률
1	367	367	4618	4618	0.080
2	853	1220	14385	19003	0.045
3	835	2055	5608	24611	0.034
4	791	2846	6186	30797	0.026
5	956	3802	9829	40626	0.024
6	805	4607	5584	46210	0.017
7	967	5574	8240	54450	0.018
8	1218	6792	7656	62106	0.020
9	1031	7823	4914	67020	0.015
10	1144	8967	5295	72315	0.016
11	1058	10025	7418	79733	0.013

시간경과에 따라 고장률이 감소하는 경우, 유니트 고장률을 나타내는 방정식 (2.7)을 데이터에 적용하여 최소자승법으로 파라미터를 결정한다. 이러한 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.



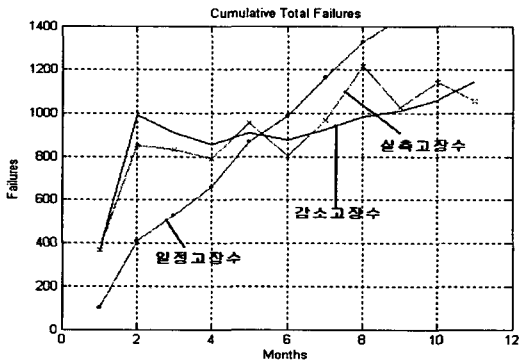
〈그림 4-1〉 제품의 총 고장률

초기과도고장률 λ_0 = 0.105 고장/월, 정상상태고장률 λ_f = 0.014 고장/월, 감쇠인자 α = 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), 식 (2.4)에 의하여 발행 후 고장률이 λ_r = 0.02141로 일정하다고 가정한 경우에는 발행 후 시간 경과에 관계 없이 그 값이 일정하여 <그림 4-1>의 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 고장데이타를 따라서 그렸으며, 그 결과를 <그림 4-2>에 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.

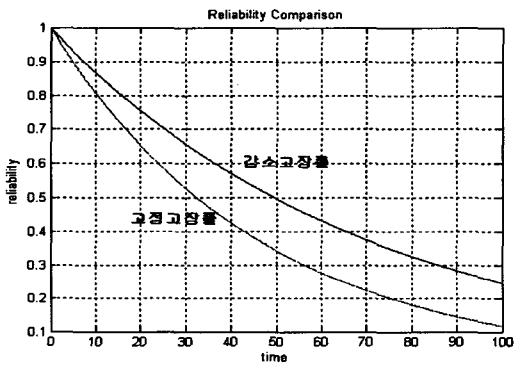
감소고장률인 경우, SSE=1.02×10^{-4}이고 표준편차는 0.003이다. 고정고장률인 경우, SSE= 4.35×10-3이고 표준편차는 0.02로서 감소고장률을 적용한 경우가 실제현상에 가깝다는 것을 알 수 있다.

4.3 신뢰도

동일한 조건 하에서 상기에서 구한 데이터를

이용하여 각각의 신뢰도를 구하여 비교한 결과는 <그림 4-3>과 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. 결 론

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.

한편, 소프트웨어 제품에 대한 고장률이 심지어 아무런 변화를 주지 않음에도 불구하고 시간에 따라 감소되는 것으로 관찰되는 것을 보아왔다. 이렇게 고장률이 감소하는 이유는 시간이 지남에 따라 사용자가 고장을 일으키는 그러한 상황, 명령, 행위를 피하는 방법을 배우기 때문이다. 또 다른 이유는 다음의 설치 사용자가 관련 부품, 드라이버, 적용 등 시스템 작업에 필요한 업데이트를 해야 하는 시스템의 형상이 부적절하여 발생했던 고장을 피할 수 있기 때문이다. 대부분의 소프트웨어 신뢰도 성장 모델은 이러한 현상을 모델화하지 않는데 이는 제품의 신뢰도가 소프트웨어 내에 존재하는 결함의 수에 의존한다고 가정하기 때문이다. 이러한 현상 때문에 인도된 제품의 정상 상태 신뢰도를 결정하는 것은 매우 어려워진다.

본 논문에서는 이러한 현상을 나타내기 위한 간단한 모델을 제시하였다. 초기 제품 사용자는 과도 고장률을 경험하게 되며, 매월 α 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.

우리는 3개의 다른 제품에 대한 고장과 판매 데이타에 우리의 모델을 적용해 왔다. 실례로서 발행 후 고장률이 일정하다는 연구에 의한 결과와 고장률이 감소한다는 연구에 의한

경우를 비교하기 위하여 실제 현장 습득 데이터를 적용하여 비교하였다. 그 결과 감소고장률에 의한 경우가 실제 현상에 적합하다는 결론을 얻었다. 감소고장률 모델이 실제 데이터에 매우 근접하다는 것을 발견했으며, 총 고장의 예측치와 실제값의 오류가 매우 합리적이었다.

참 고 문 헌

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

[2] Shigeru Yamada, and Shunji Osaki, "Cost-Reliability Optimal Release Policies for Software Systems", *IEEE Trans. on Reliability*, Vol. R-34, No. 5, 1985.12. pp. 422-424.

[3] Yang, B. and Xie, M., "A study of operational and testing reliability in software reliability analysis", *Reliability Engineering & System Safety*, Vol. 70, 2000.12. pp. 323-329.

[4] Chillarege, S. and Biyani, J. Rosenthal, "Measurement of failure rate in widely distributed software", *Proc. 25th Fault Tolerant Computing Symposium*, FTCS-25, 1995, pp. 424-433.

[5] Kan, S., Manlove, D., and Gintowt, B., "Measuring system availability - field performance and in-process metrics", *ISSRE 2003, Supplementary Proceedings*, pp. 189-199.

[6] Hiroshi Ohtera and Shigeru Yamada, "Optimal Software - Release Time Considering an Error-Detection Phenominon during Operation", *IEEE Trans. on Reliability*, Vol. 39, No. 5, 1990. 12. pp. 596-599.

[7] Min Xie, "On the Determination of Optimum Software Release Time", *IEEE*, 1991, pp. 218-224.

[8] Jalote, P. and Murphy, B., "Reliability Growth in Software Products", *Proced. of 15th Intern. Symposium on RSE*, 2004, pp. 1-7.

[9] Alan P. Wood, "Software Reliability from the Customer View", *Computing Practice*, 2003.8, pp. 37-42.

[10] Musa JD, "A theory of software reliability and its application", *IEEE trans. on software engineering*, 1975, SE-1, pp. 312-327.

[11] Govil KK, "A new model of software reliability prediction", *Microelectronics and reliability*, Vol. 23, 1983, pp. 1009-1010.

[12] Suresh N. and Babu AJG, "Software reliability estimation and optimization, a nonhomogeneous Poisson process approach", *International journal of quality and reliability management*, Vol. 14, 1997, pp. 287-300.

[13] Chin-Yu Huang and Sy-Yen Kuo, "Analysis of Incorporating Logistic Testing-Effort Function into Software Reliability Modeling", *IEEE Trans. on Reliability*, Vol. 51, 2002. Sep., pp. 261-270.

[14] Trivedi, K.S., *Probability and Statics with Reliability, Queuing and Computer Science Applications*, Second Edition, John Wiley and Sons, 2002.

□ 저자소개

최 규 식

서울대학교 공과대학 전기과를 졸업하고, 뉴욕공과대학에서 석사학위를 받았으며, 명지대학교 전기과에서 박사학위를 받았다.

OPC 중앙연구소와 KOPEC 중앙연구소에서 근무하였다.

현재는 건양대학교 의공학과 교수이며, 관심분야는 무선통신 및 소프트웨어 신뢰도이다.

문 명 호

숭실대학교 공과대학 전자공학과를 졸업하고 The Catholic Univ. of America에서 전자공학 석사를 받았으며, 건국대학교에서 박사학위를 받았다.

미국 Radiation System, Inc.에서 근무하였다.

현재는 건양대학교 의공학과 부교수이며, 관심분야는 무선통신 시스템이다.

전 인 오

호서대학교 벤처전문대학원 소프트웨어 공학박사를 취득하였다. 현재 벤처전문대학원 정보경영학과 교수이며 관심분야는 IT품질경영, S/W프로세스 평가 및 개선, S/W프로젝트관리, 컴퍼넌트기술, 정보경영, 벤처창업, 중소기업혁신 등이다.

◈ 이 논문은 2005년 6월 9일 접수하여 4차 수정을 거쳐 2006년 1월 17일 게재확정되었습니다.