운영중인 소프트웨어의 신뢰도 향상 연구

A Study on the SRGM of Operational Software

최규식
건양대학교 의공학과

Che Gyu-Shik
Konyang Univ.

요약

운영 중에는 결함 수정이 쉽지 않아서 고장률이 일정하다고 하는 연구도 있다. 그러나, 운영단계 중에는 결함을 수정하건 안하건 시간이 지남에 따라서 고장율이 줄어들어서 신뢰도가 향상된다는 사실이 관찰되고 있다. 이러한 신뢰도 성장에 대한 이유는 제품을 사용하게 됨에 따라 그에 대한 경험이 쌓이게 되며, 제품을 정확하게 사용하는 법을 배우게 되고, 또한 결함을 일으키는 환경을 발견하여 조치를 취할 수 있기 때문이다. 이러한 성장모델에 영향을 미치는 또 다른 요인을 살펴 보면 제품을 설치한 후에 사용자가 새로운 드라이버를 설치한다든지, 소프트웨어를 더 나은 새로운 버전으로 업그레이드 시킨다든지 하는 행위를 하는 것이 필요하다는 것을 알게 된다는 것이다. 본 논문에서는 이러한 현상을 표현하는 간단한 모델을 제시한다.

Abstract

It might be difficult to reduce failure rate for most of the cases are not available for debugging during operational phase, hence, there are some literatures to study that the failure rate is uniform throughout the operational time. The failure rate reduces and the reliability grows with time regardless of debugging.

As a result, the products reliability varies with the time duration of these products in point of customer view. It accumulates the products experience, studies the exact operational method, and then finds and takes action against the fault circumstances.

I propose the simple model to represent this status in this paper.

1. 서 론

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 왔다.

신뢰도가 향상되는 이유 중의 하나는 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 고장에만 직면하기 때문이다. 이는 초기에 매우 높은 고장률을 경험한 후에 사용자가 소프트웨어 제품의 정상상태 고장률에 이르기 때문이다.

본 논문에서는 운영단계에 있는 소프트웨어의 신뢰도를 고려함에 있어서 모델을 수정하지 않고 이러한 신뢰도 현상을 모델화하는 단순한 접근법을 제안한다.

2. 운영중 소프트웨어의 신뢰도 및 고장률

2.1 운영단계 신뢰도

운영신뢰도의 경우, 지금까지 연구된 문헌을 검토해보면 소프트웨어가 발행된 뒤에도 A/S가 가능한 것으로 가정하여 발행 시기를 연구한 것들이다. 그러나, MS word, 훈글 등과 같이 개발자가 불특정 다수의 사용자를 대상으로 개발하여 발행하는 경우에는 A/S가 어렵고, 따라서 그 이상 신뢰도의 성장이 없다. 신뢰도함수는 아래와 같다.

$$R_o(x|s) = \exp(-\int_0^x \lambda_s dt) = \exp[-\lambda(s)x] \quad (2.1)$$

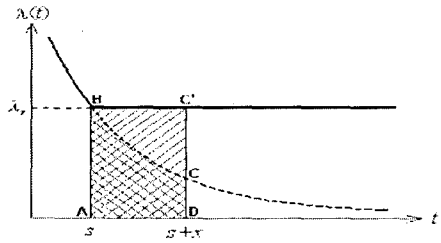
[그림 2.1]의 고장강도 곡선에서 처음 감소부분은 테스트 기간 중의 고장강도 감소를 표현하고 $\lambda = \lambda_\gamma$인 시점부터는 발행시점에서의 고장강도이므로 이것이 시간에 대하여 일정하게 되어 직선(실선)의 형태를 취한다. 테스트 신뢰도는 다음과 같이 쓴다.

$$R(x|s) = \exp[-\int_s^{s+x} \lambda(t)dt] = \exp(-S_{ABCD}) \quad (2.2)$$

여기서, S_{ABCD}는 [그림 2.2]의 곡선 사다리꼴 ABCD의 면적이다. 이와 마찬가지로 식(2.1)도 다음과 같이 나타낼 수 있다.

$$R_o(x|s) = \exp(-\int_0^x \lambda_s dt) = \exp[-\lambda(s)x] \quad (2.3)$$

여기서, $S_{ABC'D}$는 [그림 2.1]의 사각형 ABC'D의 넓이를 나타낸다.



▶▶ 그림 2.1 $\lambda(t)$가 $t \geq 0$에서 단조감소

두 가지 신뢰도의 경우를 일반적으로 다음과 같이 나타낼 수 있다.

$$R(x|s) = \exp(-S) \quad (2.4)$$

$T \geq 0$와 $x > 0$에서 주어진 어떠한 값에 대해서도 $\lambda(t)$가 $t \geq 0$에서 단조 감소하는 함수이면,

$$R_o(x|T) < R_t(x|T) \quad (2.5)$$

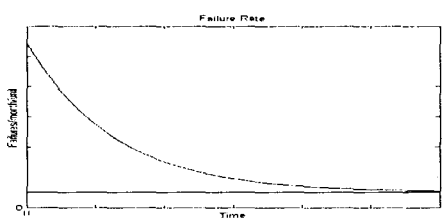
이다.

2.2 시간경과에 따른 고장률 저감

제품의 소프트웨어 유니트에서 경험된 고장률은 인도 초기 몇 달 동안 매우 높으며, 시간이 경과함에 따라 점차 감소한다는 사실이 밝혀지고 있다[9]. 이 경우, 제품 유니트에 있어서 그 제품이 판매된 t 개월 동안 경험된 고장률을 다음과 같이 모델화한다.

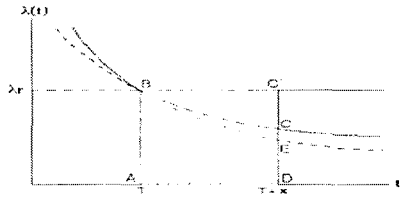
$$\lambda(t) = \lambda_0 \cdot \alpha^t + \lambda_f, \quad 0 < \alpha < 1 \quad (2.6)$$

이러한 현상을 [그림 2.2]에 표시하였다.



▶▶ 그림 2.2 유니트의 고장률

일반적으로 소프트웨어를 테스트하여 결함을 제거하게 되면 결함발견 빈도수가 줄어들기 때문에 고장강도함수가 지수함수적으로 감소하는 것으로 되어있다. 이 경우의 고장강도와 시간과의 관계를 [그림 2.3]에 보였다. 고장강도 곡선에서 처음 감소부분은 테스트 기간 중의 고장강도 감소를 표현하고 $\lambda = \lambda_\gamma$인 시점부터는 발행시점에서의 고장강도이다. 고장강도가 테스트단계보다는 완만하게 감소할 것이므로 곡선 B-E(점선)가 아니라 곡선 B-C(실선)와 같은 형태를 취한다.



▶▶ 그림 2.3 $\lambda(t)$가 $t \geq 0$에서 단조감소

2.3 파라미터 산출법

최저 제곱합에 대한 산출공식 SSE(a, λ_0, λ_f)는 다음과 같다.

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

$$= \sum_{k=1}^{n} [(\lambda_0 a^k + \lambda_f) - \lambda_k]^2 \qquad (2.6)$$

이 식을 a, λ_0, λ_f에 관하여 각각 편미분하여 비선형 최소자승 문제를 푼다.

$$\frac{\partial SSE}{\partial a} = \sum_{k=1}^{n} \{\lambda_0 a^k + \lambda_f - \lambda_k\} k a^{k-1} = 0 \qquad (2.7)$$

$$\frac{\partial SSE}{\partial \lambda_0} = \sum_{k=1}^{n} \{\lambda_0 a^k + \lambda_f - \lambda_k\} a^k = 0 \qquad (2.8)$$

$$\frac{\partial SSE}{\partial \lambda_f} = \sum_{k=1}^{n} \{\lambda_0 a^k + \lambda_f - \lambda_k\} = 0 \qquad (2.9)$$

구한 파라미터가 얼마나 실제에 적합한가를 검토하려면 5개의 성능비교기준을 검토한다.

1) $$AE = \left| \frac{M_a - a}{M_a} \right| \qquad (2.10)$$

실제적으로 사용하기 위한 목적으로 Ma는 소프트웨어 테스트 후 소프트웨어 결함 추적 과정으로부터 얻는다.

2) $$RE = \frac{m(t_q) - q}{q} \qquad (2.11)$$

여러 가지 te에 대해서 이 절차를 반복한다. tq에 대한 RE를 그려봄으로써 그 예측의 정당성을 점검한다.

3) $$Noise = \sum_{i=1}^{n} \left| \frac{r_i - r_{i-1}}{r_{i-1}} \right| \qquad (2.12)$$

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

4) $$MSF = \frac{1}{k} \sum_{i=1}^{k} [m(t_i) - m_i]^2 \qquad (2.13)$$

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) $Var=\sqrt{\frac{\sum_{k=1}^{n}(\lambda_k-\lambda)^2}{n}}$ (2.14)

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

3. 총 고장의 모델링

매월 판매되는 총유니트수를 생각해 보기로 한다. 첫 번째 월로부터 시작하여 $N_1, N_2, N_3,\ldots,N_t$라 하자. 시간에 대한 고장률 감쇠를 가진 모델을 사용하여 t 월에서의 총 고장 F_t를 구하는 다음 방정식을 얻는다.

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

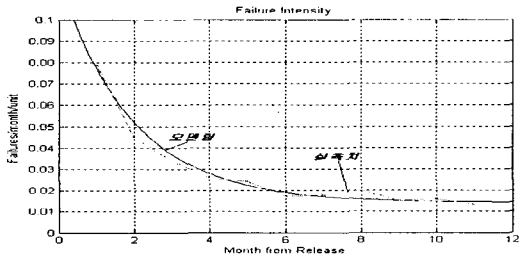
$$\lambda_f(N_t+(1-\alpha)(N_1+N_2+\cdots+N_{t-1}))$$
$$=\alpha F_{t-1}+\lambda_f(N_1+N_2+\cdots N_t)$$
$$-\alpha\lambda_f(N_1+N_2\cdots+N_{t-1})+\lambda_0 N_t \quad (3.1)$$

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

수개월 동안에 걸친 실제 고장률 데이타와 판매 유니트의 수로부터 3개의 파라미터를 결정할 수 있다. - 정상상태 고장률 λ_f, 최초고장률 λ_0, 그리고 감쇠인자 α이다.

4. 예 제

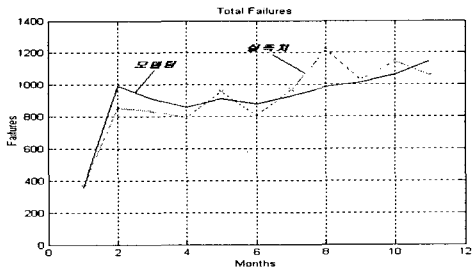
소프트웨어 제품에서 소프트웨어를 변경시키지 않고 그냥 사용하는 경우에도 시간이 경과함에 따라 고장률이 감소하는 현상이 계속 관찰되고 있다. 예를 들어 보고된 고장에 의하여 결정된 실제 제품의 사용 시간에 대한 전체적인 고장률과 판매된 총 유니트수의 관계를 [그림 4.1]에서 설명한다. 이는 실제 제품의 고장과 개수 데이타에 근거를 두고 있다.



▶▶ 그림 4.1 제품의 전체적인 고장률

전체 총 고장률의 예측치와 현장에서 실제 발견된 값의 차이를 제곱해서 합한 총합이 최소가 되도록 하는 파라미터를 찾는다. 이 방법을 이용하여 아래와 같은 값을 구했다. 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.



▶▶ 그림 4.2 예측 및 실제의 총 고장

5. 결 론

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.

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.

▌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, 1982 Aug.

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

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

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