연구논문

시험 중단 시점에 관한 소프트웨어 신뢰도 모델

문숙경
목원대학교 응용통계학과

Software Reliability Model for the Stopping Rule

Sug-Kyung Moon
Dept. of Applied Statistics, Mokwon University

Abstract

Most software reliability models and other methods attempt to estimate some measures based on its fault history. There are several phases of the software life cycle including testing phase. We can propose it's stopping rule to decide when to stop the testing and pass it on to the next phase by considering the detailed structure of software and calculating the failure rate when each fault was detected. Downs (1985) proposed a method which was developed for estimating the failure rate applicable only to two-level profiles. In this paper, I extended to profiles involving more levels.

1. 서론

소프트웨어 신뢰도에 관한 많은 논문들 중에서 한 분야가 "소프트웨어 시험을 어느 시점 까지 하는 것이 좋은가?"에 대한 연구들이다. 소프트웨어의 신뢰도는 시험 기간이 길어 질수록 높아진다. 그러나 시험 기간의 증가는 시험 비용의 증가를 수반하므로 적은 노력으로 최대의 효율을 올리는 방법에 대한 연구가 활발히 이루어 지고 있다 [4, 8, 9]. 이러한 연구들은 시험 기간 중 발견되는 결함(Fault) 수를 기초로 고장률을 산정하여, 적정 시험 기간을 예측하는 방법을 제시하였다.

소프트웨어 공학 이론에 따르자면, 소프트웨어 제품을 개발할 시 여러 단계(phase)를 설정하여 개발을 하도록 권유하고 있다. 흔히 설정되는 단계로서, 계획 단계(design phase), 실현 단계(implementation phase), 시험 단계(testing phase), 운용/보전 단계(operation/maintenance phase) 등을 꼽을 수 있다. 이러한 개발의 각 단계마다 여러가

지 품질 특성치들에 대한 목표치를 설정하여 만족되면 다음 단계로 넘어갈 수 있도록 한다. 품질 특성치는 소프트웨어 제품에 따라 달리 설정될 수 있으나 신뢰도는 빠드려서는 안될 중요한 품질 특성치 중의 하나이다.

본 고에서는 위에서 열거한 여러 단계 중 시험 단계에서 주로 적용 가능한 이론을 전개하려 한다. 즉, 소프트웨어 제품에 대해 시험을 어느 정도 더 하여야할지에 관한 문제를 다루려 한다. 이러한 시험 중단 결정의 기준에 관한 이론을 Stopping Rules 이라고 부르기도 한다 [Ross, 1985]. 이러한 시험 중단 시점을 결정하는 판단 근거는 대부분 고장률(failure rate)에 의한 것이므로, 고장률에 대한 정의를 우선 간략히 살펴보자. 고장률은 단위 시간에 고장이 몇번 발생하는가를 나타내어주는 양적인 수치이다. 흔히 신뢰도 목표치를 고장률로 표시하는 것이 일반적인 양상이다. 한편, 소프트웨어 제품이 복잡하고 거대할수록 이 고장률이 0인 시스템을 만들기란 거의 불가능할 것이다. 그래서 대부분 적정한 신뢰도 목표치를 산정하여 이 목표치에 도달되도록 소프트웨어 제품을 개발하고 있다. 이런 맥락에서 대상인 소프트웨어 제품에 대해 받아들일 수 있는 목표 고장률을 설정하여야 하겠다. 그래서 시험 기간 중 결함이 하나씩 발견될 때마다 고장률을 구하여 이것을 목표 고장률과 비교하여 가면서 마침내, 목표 고장률 수준이 되면, 시험을 중단 한다는 것이다.

이처럼 시험 중단 시점의 판단 기준이 되는 고장률을 유도하기 전에 몇가지 가정이 필요하다. 우선, 하나의 결함, 혹은 에러(error)는 하나의 고장(failure)을 유발시킨다는 것이며, 이렇게 발견된 에러는 즉시 수정되며, 수정과정 중 새로운 에러는 발생치 않는 것으로 본다. 또한 소프트웨어는 몇 개의 큰 조각 부분(section)으로 나눌 수 있다는 가정인데, 이 때, 큰 조각 부분이라함은 소프트웨어 제품의 한 부분으로서, 그 기능면이나 특징면에서 다른 부분과 그 성격을 달리하는 하나의 꾸러미에 해당될 수 있으며, 이러한 몇 개의 조각 부분들을 다 모으면 원래 대상인 소프트웨어 제품이 되는 것이다. 예를 들자면, 교환기 프로그램 중에서 호처리(call processing) 프로그램, 운용 보전(administration, maintenance) 프로그램, 제어(control) 프로그램 등의 각각을 하나의 조각 부분으로 볼 수 있다는 것이다. 그래서 본 고에서는 또한 각 조각 부분마다 사용 빈도 및 개발자의 역량에 따라 고장의 발생 정도가 다를 수 있다는 가정으로 비롯된다. Downs (1985)의 논문에서는 소프트웨어를 단지 두 조각 부분으로만 나누어 놓은 것을 본 고에서 이를 여러 조각 부분으로 일반화시켜 전개하려 한다. 앞의 예제에서 본바와 같이, 현실적으로 여러 조각으로 나누어 놓을 수 있는 경우가 더욱 많이 발생할 수 있기 때문이다.

2. 고장률 계산

2.1 가정들(assumptions)과 기호

우선 임의의 소프트웨어를 n개의 조각 부분으로 나눈다고 가정하자. 이 때, 각 조각 부분들을 $D_i, i=0, 1, 2, \cdots n$ 라 표기한다. 그리고 각 조각들에 대한 시험을 실행할 경우, 각 조각 부분들의 중요도 및 사용 빈도 등에 따라 시험 횟수, 즉 시험에 드는 노력량이 다

르다는 가정을 한다. 이 때, 각각의 조각 부분들에 대한 시험 비율을 p_i, $i=0, 1, 2, \cdots n$ 라 한다. 단 이 때, 각 조각들은 서로 독립임을 가정한다. i번째 조각 부분을 시험하기 위해 하나의 시험 경로가 선택될 확률이 p_i가 되는 셈이다.

이런 가정하에 시험을 진행하는 중에는 에러가 하나씩 발견되어 수정되어질 것이다 이 때, j번째 에러가 제거된 후의 고장률을 λ_j라 하면, 이 고장률 λ_j를 추정하는 것이 중요한 문제라 할 수 있겠다. 우선, 이러한 고장률 λ_j의 추정치를 구하기 위해 Downs (1985)의 논문에서 이미 증명된 Lemma를 본 고에서도 이용하려함으로 먼저 소개 한다.

Lemma 1 : 한 에러가 발생하고 또 다른 다음 에러가 발생할 시간 동안, 만약, 이 기간 동안 실행 환경(execution profile)의 변화가 없다고 가정하면 이 기간 동안의 고장률 λ는 아래 식과 같이 나타내어진다.
$\lambda = -r \ln [Pr\{$ 실행시키기 위해 선택된 하나의 시험 경로에 결함이 하나도 발견되지 않음 $\}]$,
, 단 이 때 r은 단위 시간 동안 수행시킨 시험 경로 수(number of test paths), 즉, 단위 시간 동안 수행시킨 평균 시험 가지 수임.

위 lemma는 에러의 분포뿐만 아니라 시험 환경(testing profile)에 무관하게 항상 성립함이 입증되었으나 시험 실행 중에 임의의 시험 경로가 선택되어질 확률은 고려되어야 할 사항이다.

2.2 시험 경로에서의 에러 수에 대한 분포 유도

하나의 시험 경로에 존재하는 에러의 수를 확률 변수라 할 경우, 다음과 같이 몇가지를 가정하고나면 쉽게 유도할 수 있다. D_i 조각에서의 총 에러수를 n_i라 가정하고, 시험 가능한 경로 수가 m_i라 한다. 단, 이 때, m_i는 상당히 큰 수가 될것이다. 그리고, 각각의 조각 부분에서 하나의 에러가 평균적으로 c_i개의 시험 경로에 영향을 끼친다고 가정을 한다면, D_i 조각 부분에서 임의의 한 에러가 시험 경로 위에 발견될 확률은 $\frac{c_i}{m_i}$ 이고, 그렇지 못할 확률은 $1-\frac{c_i}{m_i}$ 이다. 즉, 다시말하자면, D_i 조각 부분에서 m_i개의 시험 경로로 모든 시험을 실행했을 경우, 오직 c_i개의 시험 경로에서 특정 에러를 발견할 수 있다는 것이다.

그러므로 하나의 시험 경로에 t개의 에러를 포함할 확률은 아래와 같은 이항 분포(binomial distribution)형으로 나타내어진다.
즉, $Pr\{$ 어떠한 결함도 제거되기 전에 i 조각 부분에 있는 한 시험 경로에서 t개의 결함이 발견됨 $\}$

$$= \binom{n_i}{t}\left(\frac{c_i}{m_i}\right)^t\left(1-\frac{c_i}{m_i}\right)^{n_i-t}, \quad t=0, 1, 2, \cdots, n_i.$$

윗 식에서 단 하나의 에러도 제거되어지기 전에 D_i조각 부분의 한 시험 경로에서 적어도

하나의 에러가 발견되어질 확률을 $P_{f_i}(0)$ 라 표기하면 다음과 같다.

$$P_{f_i}(0) = 1 - \left(1 - \frac{C_i}{m_i}\right)^{n_i}.$$

$P_{f_i}(0)$에서의 0는 아직까지 하나의 에러도 발견 되어지지 않았음을 의미한다. 그러므로 j번째 결함이 제거된 후 적어도 하나의 에러가 발견되어질 확률은 $P_{f_i}(j)$로 표시될 것이다.

2.3 고장률 계산식 유도

앞의 가정에서 D_i 조각 부분을 통과하는 임의의 시험 경로가 뽑힐 확률이 p_i라 하였으므로 단위 시간 동안 D_i 조각을 시험하는 경로 수는 rpi가 된다. 물론 시험 시간은 각 조각 부분마다 같게 배정할 경우이다. 그런데 한번 뽑힌 시험 경로는 다시 뽑히지 않으므로, 즉 일단 한번 시험하면 다시 똑 같은 시험은 하지 않으므로 시험환경이 점차 변화된다는 사실이다. 그래서 앞 장의 lemma 가정에 위배가 되어 바로 적용하기가 어렵다. 그렇지만 대부분의 소프트웨어의 시험 경로 수가 상당히 큰 수이므로 위의 사실을 무시해도 과오가 없으므로 lemma에 적용시켜 고장률을 유도해 볼 것이다.

$Pr\{$ 어떠한 결함도 제거되기 전에, 시험을 하기 위해 선택된 임의의 한 시험 경로에 결함이 발견되지 않음 $\} = \sum_{i=0}^{n} p_i \left(1 - \frac{C_i}{m_i}\right)^{n_i}$.

그러므로, 어떠한 에러도 제거되기 전에의 고장률을 λ_0라 표기할 때, lemma 식에 의하면 다음과 같다.

$$\lambda_0 = -r \ln \left\{ \sum_{i=0}^{n} p_i \left(1 - \frac{C_i}{m_i}\right)^{n_i} \right\}.$$

λ_0의 0는 어떠한 결함도 제거되어지지 않은 상태를 나타내므로 j번째 결함이 제거된 후의 고장률은 λ_j가 될 것이다.

그러면 하나의 에러도 제거되기 전에, 시험을 위해 뽑힌 하나의 시험 경로에서 적어도 하나 이상의 결함이 발생될 확률은 다음과 같다.

즉, $Pr\{$ 어떠한 결함도 제거되기 전에, 시험을 하기 위해 선택된 임의의 한 시험 경로에 적어도 하나의 결함이 발견 됨 $\} = \sum_{i=0}^{n} p_i P_{f_i}(0)$.

그리고, 아직까지 어떠한 결함도 발견되지 않은 상태에서 시험을 위해 임의로 뽑힌 시험 경로에서 처음으로 결함이 발견되어진 경우 그 결함이 D_i 조각 부분에서 발견되어질 확률을 $P_{r_i}(0)$으로 표시하면, 그 계산식은 다음과 같다.

즉, $P_{r_i}(0) = \dfrac{p_i P_{f_i}(0)}{\sum_{i=0}^{n} p_i P_{f_i}(0)}$.

마찬가지로, $P_{r_i}(0)$의 0는 아직까지 어떠한 결함도 발견된적이 없음을 나타내므로, $P_{r_i}(j)$는 j번째 결함이 제거되어진 후 임의의 시험 경로에서 발견된 결함이 D_i조각 부분에서 발견될 확률을 나타낼 것이다.

그러므로 첫번째 결함이 제거된 후 시험을 위해 선택된 시험 경로에서 결함이 발견되지 않을 확률은 다음과 같이 표현 될 수 있다.

$Pr\{$ 첫번째 결함이 제거된 후, 시험을 하기 위해 선택된 임의의 한 시험 경로에 결함이 발견되지 않음 $\} = P_{r_1}(0)\left\{p_1\left(1-\frac{c_1}{m_1}\right)^{n_1-1}+p_2\left(1-\frac{c_2}{m_2}\right)^{n_2}+\cdots+p_n\left(1-\frac{c_n}{m_n}\right)^{n_n}\right\}$

$+P_{r_2}(0)\left\{p_1\left(1-\frac{c_1}{m_1}\right)^{n_1}+p_2\left(1-\frac{c_2}{m_2}\right)^{n_2-1}+\cdots+p_n\left(1-\frac{c_n}{m_n}\right)^{n_n}\right\}$

$+\cdots+P_{r_n}(0)\left\{p_1\left(1-\frac{c_1}{m_1}\right)^{n_1}+p_2\left(1-\frac{c_2}{m_2}\right)^{n_2}+\cdots+p_n\left(1-\frac{c_n}{m_n}\right)^{n_n-1}\right\}$

$\simeq p_1\left(1-\frac{c_1}{m_1}\right)^{n_1-1}\left(1-\frac{c_1}{m_1}\right)^{1-P_{r_1}(0)}+p_2\left(1-\frac{c_2}{m_2}\right)^{n_2-1}\left(1-\frac{c_2}{m_2}\right)^{1-P_{r_2}(0)}$

$+\cdots+p_n\left(1-\frac{c_n}{m_n}\right)^{n_n-1}\left(1-\frac{c_n}{m_n}\right)^{1-P_{r_n}(0)}$

, $\frac{c_i}{m_i} \ll 1,\ i=1, 2, \cdots, n$

$=\sum_{i=1}^{n} p_i\left(1-\frac{c_i}{m_i}\right)^{n_i-P_{r_i}(0)}$.

그래서 lemma의 식을 적용하여 첫번째 결함이 발견, 수정된 후 고장률 λ_1을 아래식과 같이 쓸 수 있다.

$$\lambda_1 = -r \ln\left\{\sum_{i=1}^{n} p_i\left(1-\frac{c_i}{m_i}\right)^{n_i-P_{r_i}(0)}\right\} .$$

그러므로, 첫번째 결함이 제거된 후 $P_{f_i}(1)$를 구하면 다음과 같이 나타낼 수 있다.

$$\begin{aligned} P_{f_i}(1) &= P_{r_i}(0)\left\{1-\left(1-\frac{c_i}{m_i}\right)^{n_i-1}\right\}+\left(1-P_{r_i}(0)\right)\left\{1-\left(1-\frac{c_i}{m_i}\right)^{n_i}\right\} \\ &= 1-\left(1-\frac{c_i}{m_i}\right)^{n_i-1}+\left\{1-(1-P_{r_i}(0))\frac{c_i}{m_i}\right\} \\ &\simeq 1-\left(1-\frac{c_i}{m_i}\right)^{n_i-P_{r_i}(0)} . \end{aligned}$$

따라서, $P_{r_i}(1) = \frac{p_i P_{f_i}(1)}{\sum_{i=1}^{k} p_i P_{f_i}(1)}$ 임을 쉽게 알 수 있다.

비슷한 방법으로,

$Pr\{$ 두번째 결함이 제거된 후, 시험을 하기 위해 선택된 임의의 한 시험 경로에 결함이 발견되지 않음 $\}$

$$\simeq p_1\left(1-\frac{c_1}{m_1}\right)^{n_1-1-P_{r_1}(0)}\left\{P_{r_1}(1)+\left(1-P_{r_1}(1)\right)\left(1-\frac{c_1}{m_1}\right)\right\}$$

$$+\ p_2\left(1-\frac{c_2}{m_2}\right)^{n_2-1-P_{r_2}(0)}\left\{P_{r_2}(1)+\left(1-P_{r_2}(1)\right)\left(1-\frac{c_2}{m_2}\right)\right\}$$

$$+\ \cdots\ +\ p_n\left(1-\frac{c_n}{m_n}\right)^{n_n-1-P_{r_n}(0)}\left\{P_{r_n}(1)+\left(1-P_{r_n}(1)\right)\left(1-\frac{c_n}{m_n}\right)\right\}$$

$$\simeq \sum_{i=1}^{n} p_i\left(1-\frac{c_i}{m_i}\right)^{n_i-P_{r_i}(0)-P_{f_i}(1)}$$

그리고 각각의 발견된 결함이 제대로 제거되어진다면 j번째 결함이 제거되어진 후의 고장률 λ_j는 다음과 같이 얻어질 수 있다.

$$\lambda_j = -r\ln\left\{\sum_{i=1}^{n} p_i\left(1-\frac{c_i}{m_i}\right)^{n_i-\sum_{k=0}^{j-1}P_{r_i}(k)}\right\},$$

where, $P_{r_i}(k) = \dfrac{p_i P_{f_i}(k)}{\sum_{i=1}^{n} p_i P_{r_i}(k)}$, $P_{f_i}(k) = 1-\left(1-\dfrac{c_i}{m_i}\right)^{n_i-\sum_{h=0}^{k-1}P_{r_i}(h)}$.

2.4 시험 중단 시점 결정 과정

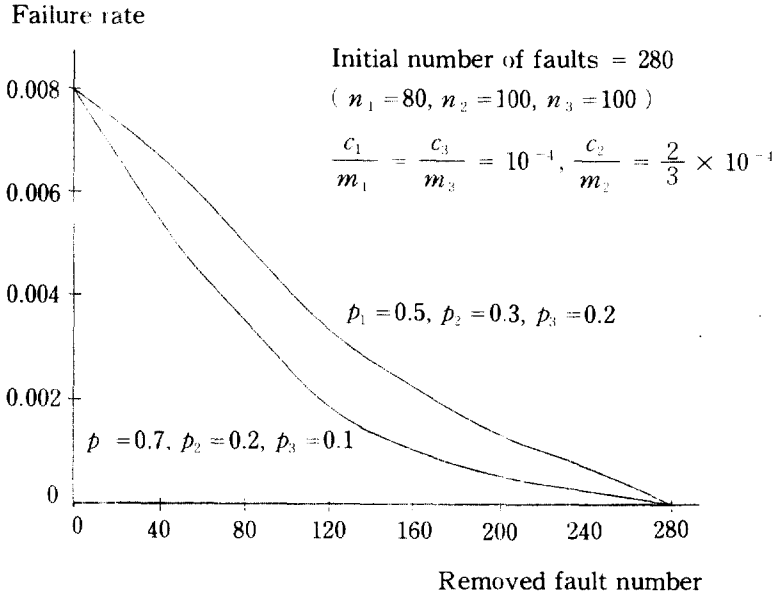
앞에서 유도한 고장률을 이용하여 시험을 어느정도 더 할 것인지 말 것인지를 결정하는 문제에 대해 생각을 해보자. 대상이 되는 소프트웨어 시스템에 대해 받아들일 수 있는 목표 고장률을 설정 한다. 이것을 λ_{opt}라 표시하기로 하자. 시험 중 결함이 하나씩 감소할 때마다 고장률 $\lambda_1, \lambda_2, \cdots$ 들을 계속해서 구할 수가 있겠고 이것을 λ_{opt}과 비교하여 시험을 중단해도 좋은지, 계속 시험을 더하여야하는지를 결정하면 된다. 이 때, n_i와 $\dfrac{c_i}{m_i}$의 대부분의 경우엔, 실제 값을 모르므로 이들의 추정치인 n_i와 $\dfrac{c_i}{m_i}$를 구하여야 하는 문제가 제기된다. 이들에 관한 정확한 추정(estimation) 방법들에 관한 연구는 나중 연구로 남겨두려 한다.

3. 고장률 검토 및 비교

본 고에서 제시된 고장률 λ_j를 다음 두 경우에 대하여 각각 실제 계산된 결과를 〈그림 1〉에 제시 하였다. 두 경우 모두 대상 소프트웨어 제품은 3조각 부분으로 나눠 있고, 총

결함수를 280개로 간주하고, 각 조각 부분마다 80, 100, 100개씩 초기 결함이 존재한다고 보고, 그리고 두 경우 모두, $\frac{c_1}{m_1}=10^{-4}$, $\frac{c_2}{m_2}=\frac{2}{3}10^{-4}$, $\frac{c_3}{m_3}=10^{-4}$로 가정한 후, 첫번째 경우엔, p_1, p_2, p_3의 값을 0.5, 0.3, 0.2으로, 두번째 경우엔 0.7, 0.2, 0.2 값을 각각 대입하여 구한 결과이다. 〈그림 1〉의 그래프를 보면 두 경우 모두 convex curve 모양임을 볼 수 있다. 또한, 두번째 경우가 첫번째 보다 convex 경향이 더욱 뚜렷이 드러나는 것을 알 수 있었다. 즉, p_i 값들 간의 값 차이가 심할수록 convex curve 경향이 더욱 심화된다는 것이다. 이러한 사실은 곧 실제적으로 많이 사용되는 부분과 사용이 많이 되지않는 부분과의 차이가 심할 경우 고장률에 대한 그래프는 convex 경향이 뚜렷이 나타난다는 것이다.

신뢰도 모델 중에서 가장 고전적인 모델인 JM(Jelinski-Moranda) 모델에서의 고장률은 제거되는 각 결함마다에 똑같은 비중을 두므로, 각 결함이 제거될 때마다 같은 크기로 줄어든다. 이 경우 고장률에 대한 그래프를 그려보면 기울기가 음인 직선 모양으로 표현될 것이다. 이것이 JM모델의 가장 큰 특징이자 비판의 대상이 되어왔다. 이러한 비판적인 관점에 선 학자 중에 한 사람인 Littlewood(1797)가 빈번히 많이 사용되어지는 부분에서 발견된 결함들은 그렇지 못한 부분들에서의 결함보다 고장률에 미치는 영향이 더 커야하며, 마땅히 구분이 되어져야 한다고 주장하였다. 빈번하게 사용되어지는 부분에서의 결함들은 그렇지 못한 부분들에서의 결함들보다 발견이 훨씬 수월하기도 하기 때문이다. 이러한 경향들을 그림으로 나타낸다면 convex curve가 될 것이다. 본 고에서 제시한 고장률 λ_i도 아래 〈그림 1〉처럼 이와 비슷한 convex curve가 됨을 알 수 있었다.



〈그림 1〉 고장률 λ_i

4. 맺음말

본 고에서는 소프트웨어 제품의 구조적인 분석을 통하여 소프트웨어를 여러 조각부분(sections)으로 나누어 소프트웨어의 고장률을 구하여, 이것으로 시험을 계속하여야할지 중단해도 좋은지를 판단하는 룰을 제공하였다. 오늘날, 소프트웨어 제품이 날로 복잡하고 대형화 되어지기 때문에 Downs(1985)가 제시한 두 부분을 몇개의 부분으로 나누어 확장시킨 본 고의 이론이 더욱 일반적이고, 나아가 유용하게 많이 활용되어질 수 있으리라 생각되어진다.

본 고의 전개 과정에서 가장 중심적으로 사용되어진 가정으로 i번째 부분에서 각각의 결함은 한결같이 c_i 갯수의 시험 경로에 영향을 미친다는 것인데, 현실적으론 각각의 결함이 영향을 미치는 시험 경로 수는 각각 다를 것이다. 이런 점에서 본 고에서 처럼 c_i를 상수로 고정시키지 말고, 각 결함마다 영향을 미치는 시험 경로 수를 확률 변수로 간주하여 본 고에서 제시한 이론보다 더욱 일반화 시킬 수 있을 것이다.

참고문헌

[1] Abdel-Ghaly, A. A., Chan, P. Y., Littlewood, B. (1986), "Evaluation of computing software reliability predictions," *IEEE Trans. SW Eng.*, Vol. 12, pp. 950-967.

[2] Brown J. R. and Lipow, M. (1975), "Testing for software reliability," *Int. Conf. Rel. Software*, pp. 518-527.

[3] Downs, T. (1985), "An Approach to the modeling of testing with some applications," *IEEE Tran. SW Eng.*, Vol. 4, pp. 375-386.

[4] Forman, E. H. and Singpurwalla, N. D. (1977), "An empirical stopping rule for the debugging and testing computer software," *J. Amer. Ass.*, Vol. 72, pp 750-757.

[5] Jelinski, Z. and Kubat, P. B. (1972), *Software reliability research, Statistical computer Performance Evaluation*, New York: Academic, pp. 465-484.

[6] Littlewood, B. (1979), "How to measure software reliability and how not to," *IEEE Tran. Rel.*, Vol. 28, pp. 103-110.

[7] Littlewood, B. (1980), *Bayesian differencial debugging model for software reliability*, COMPSAC, chicago, IL, pp. 511-519.

[8] Musa, J. D. (1975), "A theory of software reliability and its application," *IEEE Trans. SW Eng.*, Vol. 1, pp. 312-327.

[9] Ross, R. M. (1985), "Software reliability: The stopping rule problem," *IEEE Trans. SW Eng.*, Vol. 12, pp. 1472-1476.