테스트노력과 결함검출비를 이용한 소프트웨어신뢰도 모델링에 관한 연구

최규식, 김종기, 장원석

A Study on the S/W Reliability Modeling using Testing Efforts and Detection Rate

Che Gyu Shik, Kim Jong Ki

요 약

NHPP에 근거한 SRGM을 구성하는 새로운 안을 제시한다. 본 논문의 주요 초점은 소프트웨어 신뢰도모델링에서 효과적인 파라미터분해기법을 제공하는 것이다. 이는 테스트노력과 결함검출비를 동시에 고려하는 것이다. 일반적으로, 소프트웨어결함검출/제거메카니즘은 이전의 검출/제거결함과 테스트노력을 어떻게 활용하느냐에 달려있다. 실제 현장 연구로부터 우리는 테스트노력소모패턴을 추론하여 FDR의 경향을 예측할 수 있을 것으로 생각된다. 결함검출이 증가, 감소 및 일정한 것 등 광범위에 걸쳐서 나타나는 경향을 잡아내는 고유의 융통성을 가지는 하나의 시변수집합인 FDR모델에 근거한 테스트노력을 개발하였다. 이 스킴은 구조에 융통성이 있어서 여러 가지 테스트노력을 고려하여 광범위한 소프트웨어 개발환경을 모델화할 수 있다. 본 논문에서는 FDR을 기술하고, 관련된 테스트 행위를 이러한 새로운 모델링접근법에 연합시킬 수 있다. 우리의 모델과 그리고 이것과 관련된 파라미터 분해기법을 적용한 것을 여러 가지 소프트웨어 프로젝트에서 도출한 실제 데이터집합을 통하여 시연한다. 분석결과에 의하면 SRGM에 관한 테스트노력과 FDR을 결합하기 위한 제안된 구조가 상당히 정확한 예측능력을 보여주고 있으며, 실제 수명상황을 좀더 정대하게 설명해 준다. 이 기법은 광범위한 소프트웨어시스템에 쓰일 수 있다.

요약어 : 결함검출, NHPP, 소프트웨어결함 및 고장, SRGM, 테스트 노력 함수

1. 서론

컴퓨터 및 정보통신의 급격한 발달 때문에 현대사회는 점점 더 소프트웨어에 집중된 시스템에 의존해가고 있다. 소프트웨어는 값비싼 과학계산시스템, 금융시스템, 산업에의 적용, 대학교의 컴퓨터 센터, 가정의 개인컴퓨터를 포함한 많은 모뎀시스템에 내장되어 있다. 복잡한 대형 소프트웨어 시스템에 대한 수요가 급속히 증가하고 있어서 시스템에서 프로그래머의 설계 오류확률이 상당히 증가해가는 경향을 보이고 있다. 결국, 소프트웨어 고장에 의한 위기확률은 계속해서 증가되고 있다. 이러한 고장은 많은 기업에서 세입상의 손실을 초래하게 된다. 그러므로, 시스템의 신뢰도를 결정하기 위해 소프트웨어의 신뢰도를 신중히 검토해야 한다.

소프트웨어신뢰도와 하드웨어신뢰도는 모두확률분포로 설명될 수 있으므로 둘이 유사하다. 그러나, 소프트웨어 결함은 눈에 보이는 하드웨어 결함에 비하여 실현하고 검출하고 교정하기가 쉽지 않다. 어떤 시스템이 되었든 그 신뢰도는 시스템 설계의 정확성, 시스템의 설계를 구현하여 매핑시키는 정확성, 시스템 부품의 신뢰도에 의존한다. 소프트웨어는 적어도 물리적인 의미에서 열화되거나 낡아지지 않으므로 소프트웨어의 시스템이 하드웨어의 시스템과는 상당히 다르다. 그러므로, 소풍트웨어와 하드웨어는 결함과 고장이 패턴이 상이하며, 그 상이성은 중요하고 기초적인 것이다. 그러므로, 하드웨어 신뢰도를 측정하는 동일한 모델을 소프트웨어 신뢰도측정 모델로 그대로 사용할 수 없다.

하드웨어는 고장율이 상승하는 것과 하강하는 것의 혼합이다. 하강 고장율은 본래 설계와 관련된 하드웨어 고장에 의한 것이다. 상승 고장율은 하드웨어 부품의 노화나

마모에 의한 것이다. 한편, 소프트웨어 시스템은 일반적으로 고장율이 감소하는 경향을 가진다. 소프트웨어 신뢰도가 확률적이냐 결정론적이냐 하는 데에는 약간의 혼동이 있다. 그 대답은 우리가 소프트웨어의 신뢰도를 어떻게 보냐에 달려있다. 그러나, 소프트웨어는 그 주위환경과 연관시켜 검토해야 한다. 소프트웨어 신뢰도를 측정하는 여러 가지 동일한 소프트웨어라도 상이한 테스트/운전 조건 하에서 다른 값을 나타내기 때문이다. ANSI의 정의에 의하면 "Software reliability is the probability of failure -free software operation for a specific period of time in a specified environment."이다. 따라서, 시스템 신뢰도를 결정함에 있어서 소프트웨어 신뢰도를 정확하게 모델링하고 그의 가능한 여러 가지 경향을 예측하는 것이 필수적이다. 소프트웨어 시스템을 신뢰성 높게 성취하려면 프로그램개발자나 테스트팀에 의해서 여러 소프트웨어 결함검출/제거 기법을 사용할 수 있어야 한다. 이러한 기법을 적용함에있어서 SRGM이 중요한데 그 이유는 테스트/디버그 단계 동안 이들이 매우 유익한 정보를 개발자나 테스트자에게 제공할 수 있기 때문이다.

여러 가지 결함 검출 기법들이 발행되었으며, 실제 DS로부터 소프트웨어 신뢰도를 도출하기 위한 많은 노력들이 있었으며, 그들 대부분은 다음과 같은 사항에 근거를 두고 있다.

- 역일 시간 예를 들면 J-M 모델
- 직원시간 예를들면 Shooman 모델
- 컴퓨터시간 예를들면 Musa 모델

무사[5-6]는 처음에 실제 소프트웨어시스템에서 DS를 취하여 실행시간 이론의 타당성을 논하였다. 그러나, 대부분의 기존 DS는 실행시간 개념에 의존하지 않는다. 최근에는 각각 웨이블형과 로지스틱TE 함수를 가진형을 가진 두 개의 단순한 SRGM을 제안하였다. 이러한 모델들은 역일테스트, TE의 양, 테스트 기간 중에 검출되는 결함의 수 사이의 관계를 설명하려 노력하였다. TE는 인력, 테스트 케이스의 수, CPU 시간 등에 의해 측정할 수 있다. 실제 소프트웨어 개발 프로젝트에까지 확장하여 적용할 때는 이러한 모델들이 실제 관찰 데이타와 잘 적합되고 소프트웨어 개발 단계기간 동안의 자원 소모 공정에 대해서 통찰력 있는 기술을 해준다.

일반적으로, 여러 SRGM 중에서 두 개의 가장 중요한 인자가 신뢰도에 영향을 준다. 그것은 초기결함의 수와 FDR이다. 초기결함의 수는 소프트웨어를 초기에 테스트할 때의 결함의 수이다. 이 수는 소프트웨어 신뢰도 측정의 대표치이다. 잔여결함의 수를 알게 되면 고객이 그 소프트웨어를 쓰기에 적합한가 아닌가, 또 얼마만큼의 테스트자원이 필요한가를 결정하는데 크게 도움이 된다. 이는 결국 고객에 의해서 접하게 되는 고장의 수를 추정할 수 있게 된다. 한편 FDR은 테스트기법과 테스트케이스에 의해서 검출되는 결함의 효율성을 측정하는데 쓰인다. 다량의 문헌에서 대부분의 연구자들은 SRGM을 유도할 때 FDR이 상수인 것으로 가정한다. 이는 소프트웨어 테스트 기간동안 결함검출확률이 동일한 것으로 가정하고, 결함발생 간격 전체에 걸쳐서 결함검출비가 일정한 것으로 가정한 것이다.

실제로는 FDR이 테스트 팀의 기술력, 프로그램의 크기, 소프트웨어의 테스트성에 강력하게 의존한다. 소프트웨어 개발 프로젝트에 대한 여러 가지 실험 및 분석에 걸쳐서 FDR이 시간 경과에 따라 3개의 가능한 경향을 가진 것이 밝혀졌다. 그것은 증가, 감소 또는 불변이다. 그러므로 우리는 FDR을 이러한 가능한 경향을 설명하기 위한 시간함수로 취급한다. 즉, 우리는 시변 결함 검출 함수를 가정한다. 그리고 소프트웨어 신뢰도 모델링을 할 때 이러한 두 개의 개념 TE함수와 시변 FDR을 통합하여 조합하였다.

본 논문에서는 MLE와 LSE를 이용하여 SRGM파라미터를 산출한다. 산출된 파라미터를 제안된 소프트웨어 결함 예측 모델에 취하여 예측된 모델과 다른 기존 SRGM과 비교하였다. 비교 결과로부터 우리의 분석에서는 예측된 결과가 실제 결과와 일치하는가 불일치하는가, 그 이유는 무엇인가를 결정한다. 실험적 결과에 의하면 조합된 모델이 좀더 정확한 예측치를 보여주고 있으며, 실제 시간 상황을 좀더 합리적으로 기술해주는 것을 알았다.

2 항에서는 TE 함수와 시변 FDR을 조합하는 기본 SRGM을 설명한다.

3 항에서는 기본 SRGM을 확장하여 여러 가지 시변 FDR을 고려하고 여러 모델링 제안들을 기술한다. 4항에서는 실제 관찰된 소프트웨어 고장데이타에 근거한 제안된 SRGM의 파라미터를 산출하고 평균치 함수를 그리며, 이들을 다른 기존 모델과 비교한다. 5항에서는 FDR과 신뢰도 경향을 논의한다.

2. TE-기준 소프트웨어 신뢰도 모델링

TE-기초의 수학적표현은 다음과 같다.

$$\frac{dm(t)}{dt} \cdot \frac{1}{w(t)} = r(t) \cdot [a - m(t)],\ a > 0,\ 0 < r(t) < 1$$

$$\frac{dm(t)}{dt} = w(t) \cdot r(t) \cdot [a - m(t)] \qquad (1)$$

이 SRGM(1)은 다음과 같은 가정하에 이루어졌다.

1) 결함제거공정이 NHPP이다.

2) 소프트웨어시스템이 시스템의 잔여결함을 명시함으로써 생기는 무작위 시간에서의 고장에 의존한다.

3) (t, t+Δt] 시간에 검출되는 결함의 평균치, 현재의 w(t)에 의한 dm(t)/dt는 시스템의 잔여결함 평균치에 비례한다.

4) r(t)는 시간의 함수이다. (상수가 아니다)

5) TE의 시간 종속적인 거동은 로지스틱분포나 웨이블분포로 모델화할 수 있다.
6) 고장(failure)이 발생하는시간마다 그것을 일으키는 결함이 즉각적이고도 완벽하게 제거되며, 새로운 결함이 새로이 도입되지 않는다.
7) 오류(error)를 교정하는데 드는 시간은 극히 미약하고 검출된 오류는 확실히 제거된다.

방정식(1)은 검출결함의 수에 영향을 주는 두 개의 요소를 가지고 있다. w(t), r(t)가 그것이다. r(t)가 상수이면

$$\frac{dm(t)}{dt} = w(t) \cdot r \cdot [a - m(t)] \qquad (2)$$

이고, 경계조건 m(0)=0을 이용하여 방정식(2)를 풀면

$$m(t) = a \cdot \{1 - \exp(-r \cdot [w(t) - w(0)])\} \qquad (3)$$

이다.

2.1 TE 함수

(1)의 첫 번째 부분 은 TE 함수이다. 소프트웨어의 테스트/디버깅 단계에서는 평가 가능한 TE, 예를 들면 테스트 케이스의 수, 인력, CPU시간과 같은 것이 소모된다. 그래서, 인력의 자원소모나 할당은 여러 분포로 모델화할 수 있다. 여러 연구로부터 2-A에서 보인 바와 같이 여러 형태의 TE 표현이 가능하다.

1) 일정 TE 소모 : 고전적인 SRGM의 유도에서 대부분의 연구자들은 소프트웨어 시스템의 TE(작업량)가 일정한 것으로 가정하였다.

$$w(t) = w_o \qquad (4)$$

2) 웨이블형 TE함수 : 많은 논문들에 의하면 TE는 테스트 단계 전 기간에 걸쳐서 일정하지 않다는 것이다. 사실상 순간적인 TE가 테스트 수명기간 동안 감소하여 누적 TE가 한계제한치에 접근하도록 하는 것이다. 이 분석은 합리적이라고 할 수 있는데 이는 어느 소프트웨어 회사도 소프트웨어 테스트에 무한자원을 소모하려 하지 않을 것이기 때문이다. 그러므로 우리는 TE를 웨이블함수로 설명한다.

$$W(t) = N \cdot \left[1 - \exp\left(-\int_0^t g(\tau)d\tau\right)\right] \qquad (5)$$

$$W(t) \equiv \int_0^t w(\tau)d\tau \qquad (6)$$

이는 3가지 경우가 있다.

(1) $g(t) = \beta$

그러면 $w(t) = N\beta \cdot \exp(-\beta t)$ 로서 여기에서는 지수곡선이 있고 누적 (0,t]에서의 TE는

$$W(t) = N \cdot [1 - \exp(-\beta t)] \qquad (7)$$

이다.

(2) $g(t) = \beta t$이면

$w(t) = N\beta t \cdot \exp(-\frac{\beta}{2} \cdot t^2)$로서

레일레이 곡선이고 누적 TE는

$$W(t) = N\left[1 - \exp(-\frac{\beta}{2} \cdot t^2)\right] \qquad (8)$$

이다.

(3) $g(t) = w(t) = N\beta tm \cdot t^{m-1} \cdot \exp(-\beta t^m)$

여기에는 웨이블 곡선이 있으며 누적 TE는

$$W(t) = N[1 - \exp(-\beta t^m)] \qquad (9)$$

이다.

3)로지스틱 함수 :

m>3일 경우에는 웨이블형 TE함수가 겉보기 피크 현상을 가진다. 이러한 현상은 현실적이지 못한 것으로 보이는데 그 이유는 실제 소프트웨어 개발과정에서 보편적으로 쓰이지 않기 때문이다. 그러므로, 테스트 노력 패턴을 설명하기 위해 로지스틱 TE함수를 제안하는 바이다. 또한, [24]에서는 이러한 함수가 1978-1980 프로젝트 서베이에서는 매우 정확하게 맞았다는 것을 보고하고 있다. 그러므로,

$$W(t) = \frac{N}{1 + A \cdot \exp(-\alpha t)} \qquad (10)$$

$$w(t) = \frac{dW(t)}{dt} = \frac{NA\alpha \cdot \exp(-\alpha t)}{[1 + A \cdot \exp(-\alpha t)]^2}$$

$$= \frac{NA\alpha}{\left[\exp(\frac{\alpha t}{2}) + A \cdot \exp(-\frac{\alpha t}{2})\right]^2}$$

2.2 FDR

(1)의 두 번째 부분은 FDR이다. 테스트 단계 기간동안 소프트웨어의 결함을 발견하는 비율이다. 보통은 소프트웨어 결함이 검출되었든지 아닌지에 관계 없이 프로그래머/디버거의 능력, 소프트웨어구조, 소프트웨어 개발절차의 숙련도, 프로그램 모듈간의 상호관계 여하에 의존한다.

· 테스트단계의 시초에는 검사에 의해서 많은 결함을 쉽게 검출하며, FDR은 결함 발견 효율, 결함강도, TE, 검사비율 여하에 의존한다.

· 테스트 단계의 중간쯤에는 FDR이 보통 CPU명령, 고장-결함의 관계, 코드확장인자, 역일당 정해진 CPU실행시간 등과 같은 다른인자에 의존한다.
결국, FDR은 산출 가능하다. 우리는 점검활동 절차를 추적하기 위해, 테스트 계획의 효용성을 평가하기 위해, 그리고 우리가 채택한 점검방법을 평가하기 위해 이 비율을 사용하였다.

1) 일정 비례성 : 대부분의 기존 SRGM은 [t, t+⊿t)에서 검출되는 결함의 평균치가 잔여 결함의 수에 비례하는 것으로 가정하였다.

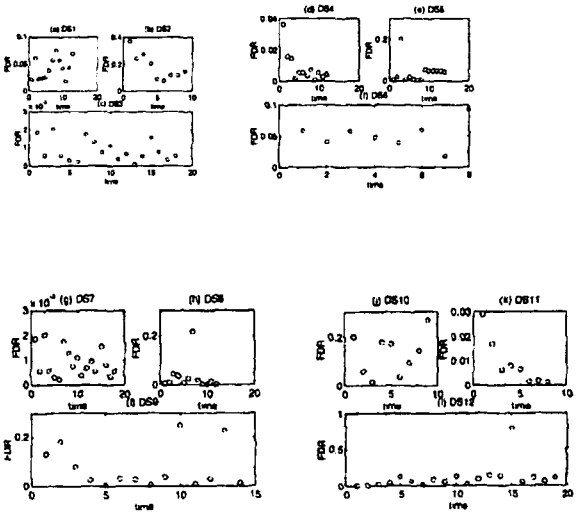
$$m(t+\Delta t)=r\cdot w(t)\cdot[a-m(t)]\Delta t \qquad (11)$$

r는 일정 비례상수이다.

2) 시변 FDR : 우리의 실험에서는 FDR이 "TE 소모당 검출되는 결함의 평균수" 또는 "특별 점검 활동에 의해서 검출되는 결함의 수"에 의해서 측정된다. 이 정보는 시스템 개발자가 점검활동을 계획하고 문제점을 진단하며 변화의 효과를 평가하는는 데에 도움이 된다. 그리고 결국은 이것이 여러 점검 활동의 비용효과분석을 지칭한다. 시간경과에 따른 FDR의 변량을 설명하기 위해 우리는 몇 가지 실제적인 테스트/디버깅 DS를 탐사해보았다. 우리는 그러한 결함 검출 절차를 분석하여 다양한 결함 검출거동을 관찰하였다. 대부분의 그룹 DS는 다음과 같은 형태를 취하고 있다.

$$(t_0,\ m_0),\ (t_1,!m_1),\ \cdots,\ (t_n,\ m_n) \qquad (12)$$

mj =tj까지 검출되는 결함의 총수



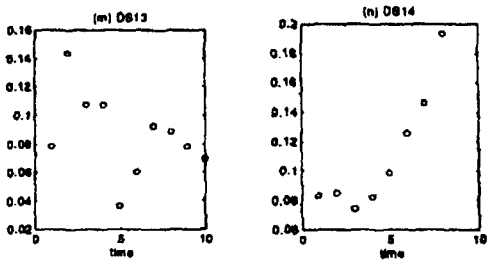


그림 1. FDR의 시간 변화

일반적으로 말해서 역일시간에 근거해서 얻어진 데이터는 시끄러운(단기 무작위) 경향이 있으며, 기존 SRGM에 있어서 부합되지 못한다고 할 수 있다. 여러 가지 시각에서 FDR을 해석하는 하나의 방법은 계산접근법을 쓰는 일이다. (3)으로부터 또 m(ti), m(ti+1)을 이용함으로써 ti와 ti+1사이의 FDR을 다음과 같이 평가한다.

$$\frac{m(t_i)}{m(t_{i+1})}=\frac{a\cdot[1-\exp(-r\cdot\exp\Delta W(t_i))]}{a\cdot[1-\exp(-r\cdot\exp\Delta W(t_{i+1}))]} \qquad (13)$$

또는

$$m(t_i)\cdot[1-\exp(-r\cdot\exp\Delta W(t_{i+1}))]$$
$$-m(t_i)\cdot[1-\exp(-r\cdot\exp\Delta W(t_i))]=0 \qquad (14)$$

방정식(14)는 수치적으로 풀을 수 있다(보통은 컴퓨터로 계산). 그림 1에서는 FDR이 여러 가지 실제적인 DS에 대해서 시간적으로 변화하는 것을 보여주고 있다.
- 그림1(a), (e), (f)에서는 시간이 경과함에 따라 FDR이 증가하는 경향이 있음을 보여준다. - 그림 1(b)-(d), (g), (k), (m)에서는 시간이 경과함에 따라 FDR이 증가하지 않는 것을 보여주고 있다.
- 그림 1((h), (i), (l)에서는 FDR이 일정한 것으로 보여주고 있다.

FDR사태를 기술함에 있어서 꼭대기도 있고 골짜기도 있음을 알수 있다. 이는 테스트 주제, 테스트 팀, 또는 테스트중인 소프트웨어의 급격한 변경으로 인하여 생기는 것으로 사료된다.

하나의 소프트웨어 테스트 공정은 유니트테스트, 집적테스트, 시스템테스트, 설치테스트를 포함하여 여러 테스트 단계로 구성된다. 소프트웨어 시스템이 매우 커서 복잡하면 프로그래머들은 소프트웨어 테스트 초기 단계에서 감사를 통하여 쉽게 그들 소프트웨어의 결함을 검출하여 제거할 수 있다. 시간이 경과함에 따라 테스트단계는 집적테스트로 진행하게 되고 그리고 시스템 테스트단계로, 그리하여 프로그래머가 잔여결함을 검출하기가 더욱 더 어려워지게 된다. 이 경우에는 초기에 FDR이 증가하고 그

다음에 감소하게 된다. (그림 1(b), (k) 또는 (m)참조) 다른 말로 말해서 결함 검출시간이 길어진다. 한편, 소프트웨어가 별로 크지 않아서 프로그램 모듈이 많지 않을 때는 프로그래머의 테스트 기술이 시간이 경과함에따라 향상되고 좀더 효과적으로 테스트를 수행할 수 있을 것이다. 따라서, FDR이 증가하는경향을 가진다.(그림1(f), (j) 또는 (n) 참조) 요건이 변경되고 새로운 형상이 추가되면 또는 디버깅 기간 동안 새로운 결함이 도입되면 이 때에도 증가하고. 어떠한 경우에도 실제프로젝트 데이터에 보인 바와 같이 FDR이 시간이 경과함에 따라 3개의 가능한 경향을 나타내는데 증가, 감소 또는 일정한 것이 그것이다. 3항에서는 여러 FDR에 대해서 소프트웨어 모델링에 대해서 연구한다.

3. FDR

우리는 어떤 특정한 FDR 표시를 연구하여 여러 제안을 하였다. 결함 검출 과제가 코딩 후에 프로그래머에 의해서 수행되기 때문에 그들은 소스 코드나 객체 코드 수행결과로 생기는 결과를 분석할 것이다. 항 2-B-2에서는 시간에 의존하는 계수가 일정 FDR 가정을 대신할수 있다는 것을 제안하였다. 결과를 해설하기 위해 FDR이 m(t)의 함수인 것으로 가정하였다. 이는 초기 결함, 검출된 결함의 수, FDR간에 어떤 관계가 있음이다. (1)을 재배치해보면 테스트시각 t에서의 잔여 결함당 FDR을 설명할 수 있다. 이는 현재의 결함 내용에 대한 결함의 검출도를 나타낸다.

$$d(t)=\frac{dm(t)}{dt}\cdot\frac{1}{a-m(t)}=r(t)\cdot w(t) \qquad (15)$$

즉,

$$r(t)=\frac{d(t)}{w(t)}$$

이다.

방정식(15)는 d(t)∝r(t)로서 r(t)↑이면 d(t)↑이고, r(t)↓이면 d(t)↓인 것을 의미한다. 그리고 잔여결함당 FDR이 현재의 w(t)의 함수임을 의미한다. 우리는 d(t)를 소프트웨어 신뢰도 성장지수로 간주한다. 대부분의 소프트웨어 신뢰도 모델들은 w(t)가 일정한 것으로 가정한다. (일부는 그것을 고려하지 않는 경우도 있고 그런 경우 r을 상수비로 설정해 놓는다.) w(t)=일정한 경우에 대해서 d(t)=일정하며, 이는 이러한 모델들이 동차 FDR을 가짐을 가리키는 것이다. 그러나, 실제 프로젝트에서 (15)는 d(t)의 거동에 대해서 좀더 정확한 결정을 내리도록 해준다. 3-A-III-C에서 여러 가지 시나리오에 대한 고찰을 한다.

A. 제안 1 : r(t)에 대한 상수 FDR

r(t)가 t에서 일정하면 m(t)는 일정 FDR이다.

$$r(t)=r \qquad (16)$$

방정식(16)에서는 테스트 기간중에 모든 결함이 동일하게 검출될 수 있다는 것을 보여준다. 이러한 가정 하에 단위 TE당 FDR은 일정하다. (16)을 (1)에 대입하여 경계조건 m(0)=0인 조건 하에서 미분방정식을 풀면

$$m(t)=a\cdot[1-\exp(-r\cdot\Delta W(t))] \qquad (17)$$

(15)로부터 테스트시각 t에서의 잔여 결함당 FDR은

$$d(t)=r\cdot w(t) \qquad (18)$$

방정식(18)은 w(t)에 의해서 d(t)가 동차이건 비동차이건 간에 d(t)가 두드러지게 나타남을 보여준다.

B. 제안2 : 비감소 r(t)

r(t)가 t에서 감소하지 않는다면 m(t)는 증가하는 결함 검출 함수를 가진다.

케이스 2.1 :

$$r(t)=r_0+k\cdot\frac{m(t)}{a},\ k>0 \qquad (19)$$

이러한 가정 하에 FDR을 설명하기 위해 선형 모델을 사용한다. (19)에서 r_0는 초기 FDR이고 k는 LSE로 계산할 수 있는 기울기(모델파라미터)이다. k는 증가하는 FDR을 추적하고 예측하는데 쓰인다. (19)를 (1)에 대입하여 미분방정식을 푼다.

$$m(t)=a\cdot\left(1-\frac{r_0+k}{r_0\cdot\exp[(r_0+k)\cdot\Delta W(t)]+k}\right) \qquad (20)$$

(15)로부터 잔여결함당 FDR은

$$f(t)=w(t)\cdot\left(1-\frac{k}{\exp[(r_0+k)\cdot\Delta W(t)]+k}\right) \qquad (21)$$

케이스 2.2 :

$$r(t)=r_0+(r_f-r_0)\cdot\frac{m(t)}{a},\ 0<r_0<r_f \qquad (22)$$

(22)를 (1)에 대입하면 이는 리카티 미분방정식이 되며 그 해는

$$m(t)=a\cdot\left(1-\frac{r_f}{r_0\cdot\exp[r_f\cdot\Delta W(t)]+r_f-r_0}\right) \qquad (23)$$

이와 유사하게 (15)로부터 시각 t에서의 잔여 결함당 FDR은

$$d(t)=w(t)\cdot r_f\left(1-\frac{r_f-r_0}{r_0\cdot\exp[r_f\cdot\Delta W(t)]+r_f-r_0}\right)$$

이며, 이는 단조증가한다. 마찬가지로 (24)는 (23)이 결함검출도가 소프트웨어 테스트공정이 진행되면서 증가하는 결함검출공정을 기술해주는 것을 의미한다.

C. 제안 3 : 비증가 r(t)

r(t)가 시각 t에서 증가하지 않으면 m(t)는 감소 FDR 함수이다. 이러한 경우는 테스트 초기에 검출이 쉬운 많은 결함을 검출하고 나중의 일부결함은 검출하기 매우 어렵다는 것을 상황을 기술해준다.

케이스 3.1 :

$$r(t)=r_0\cdot\left(1-\frac{m(t)}{a}\right) \tag{25}$$

(25)를 (1)에 대입하여 그해는

$$m(t)=a\cdot\left(1-\frac{1}{r_0\cdot\Delta W(t)+1}\right),\quad 0<r_0<r_f \tag{26}$$

(15)로부터 시각 t에서의 잔여결함당 FDR은

$$d(t)=\frac{r_0\cdot w(t)}{r_0\cdot\Delta W(t)+1} \tag{27}$$

케이스 3.2 :

$$r(t)=r_0+k\cdot\frac{m(t)}{a},\quad k<0 \tag{28}$$

(28)을 (1)에 대입하면 그해는

$$m(t)=a\cdot\left(1-\frac{r_0+k}{r_0\cdot\exp[(r_0+k)\cdot\Delta W(t)]+k}\right)$$

(29)

케이스 3.3 :

$$r(t)=r_0+(r_f-r_0)\cdot\frac{m(t)}{a},\quad r_0<r_f \tag{31}$$

(31)을 (1)에대입 하여 그해는

$$m(t)=a\cdot\left(1-\frac{r_f}{r_0\cdot\exp[r_f\cdot\Delta W(t)]+r_f-r_0},\quad r_0>r_f\right)$$

(32)

(15)로부터 시각 t에대한 잔여결함당 FDR은

$$d(t)=w(t)\cdot r_f\cdot\left(1-\frac{k}{r_0\cdot\exp[r_f\cdot\Delta W(t)]+r_f-r_0}\right)$$

(33)

D. 모델분류

표1은 항 3-A-III-C의 제안을 4개로 나누었다.

여러 TE함수를 이 4개 그룹 각각에 대해 적용하여 어떤 특수 소프트웨어 모델을 형성한다. 4개의 신뢰도 모델 즉 로지스틱, 웨이블, 레일레이, 지수의 4개 TE 모델을 생각해보기로하자. 우리의 접근방법에 근거하여 최대 16개의 모델이 있을 수 있다.

4. 실험적 연구 결과

3항의 우리 모델들의 성능을 점검하기 위해 그리고, 다른 기존 SRGM과 비교하기 위해서 우리는 우리의 모델에 3개의 DS를 적용하였다. 이러한 DS들은 표 2에 있다. 채택한 모델들을 평가하기 위한 두 개의 비교 기준은 다음과 같다.

1)AE

$$AE=\left|\frac{M_a-a}{M_a}\right| \tag{34}$$

표1 모델비교

그룹	FDR	케이스
A	$r(t)=r$	1
B	$r(t)=r_0+(r_f-r_0)\cdot\frac{m(t)}{a}$, $0<r_0<r_f$ 또는 $0<r_f<r_0$	2.2, 3.3
C	$r(t)=r_0+k\cdot\frac{m(t)}{a}$	2.1, 3.2
D	$r(t)=r_0\cdot\left(1-\frac{m(t)}{a}\right)$, $r_0>0$	3.1

Ma≡테스트 중 및 테스트 후 검출되는 결함의 총수

실제 적용에 쓸 목적으로 Ma는 소프트웨어 테스트 후 소프트웨어결함 추적을 하여 얻었다.

2) MSF

$$MSF=\frac{1}{k}\sum_{i=1}^{k}[m(t_i)-m_i]^2 \tag{35}$$

MSF의 값이 작으면 적합 에러가 줄어들고 성능이 좋다는 것을 의미한다.

잔여결함의 수를 결정하여 소프트웨어가 어느 규정된 기간동안 정상 작동할 확률을 결정하는데 도움을 주는 기타 다른 정량적인 척도로서는 다음과 같은 것이있다.

1) MF(최대결함) : 초기결함의총수, m(∞)

2) RF(시각 t에서 시스템에 잔존하고 있는 결함) : m(∞)-m(t), 소프트웨어 신뢰도에 있어서 중요한 인자이며, 유지보수 활동을 계획하는데 매우 중요한 척도이다.

3) MTTF(고장간 평균 시간)

4) SR(소프트웨어 신뢰도)

4.1 DS#1

시스템은 PL/1 데이터베이스 소프트웨어이다. (7)-(9)에서의 웨이블형 TE함수에서의 α, β, m과 (10)의 로지스틱 TE 함수의 N, A, α는 MLE와 LSE로 유도할 수 있으며, 평균치함수의 a, r_0, r_f, k도 수치적으로 계산할 수 있다. 그림 2에서는 (7)-(10)을 이용하여 평가한 것의 적합성을 보여주는 것이다. 표 3에서는 여러 TE함수, 평균치함수, 비교기준을 종합해놓았다. 우리가 제안한 소프트웨어 신뢰도 성장모델은 K-S goodness-of-fit을 통하여 평가해본 결과 5% s-significance level에서 잘 적합됨을 알 수 있다. 표 3으로부터 그룹 B의 MSE와 AE 모두 다른 그룹들의 것보다 적으며, 그리고 기존의 SRGM보다도 적다. 그러므로, 그룹 b는 좀더 나은 goodness-of-fit를 가진 것으로 볼 수 있다. 이 성능 개선은 일정한 FDR을 가정한 기존의 것 대신에 FDR 패턴을 해설하기 위해 2개의 파라미터를 이용하면 얻을 수 있다. 결함검출현상을 모델링하기 위해 몇 개의 파라미터를 추가하여 평가하는 것은 매우 지겨운 일이다. 왜냐면 좀더 복잡한 계산을 더 많이 수행해야 하기 때문이다. 그러나, 몇 개의 파라미터를 넣는다 해도 자동화하면 쉽게 계산할 수 있다., 대형 상용 소프트웨어나 안전이 극히 중요한 항공기의 경우와 같은 매우 중요한 시스템이 있어서 고도의 신뢰도를 요하는 경우에는 더욱 더 정확한 신뢰도를 위해 비용이 추가로 소요되는 것은 작은 인자이다. 우리의 모의로부터 계산된 평균치 함수의 연속곡선은 변곡점을 가지며, 즉 표3의 그룹 B에서 $r_f \gg r_0$이므로 S-형 거동을 한다.

그러한 가정 하에 유도된 소프트웨어 신뢰도 모델(22)은 여러 문헌에서 쓰였다. 종합해서 말하면 TE 함수와 시변 FDR을 결합하여 만든 조합모델(그룹B)은 실제로 본 실험에서 DS에 만족할 만하게 적합되었다.

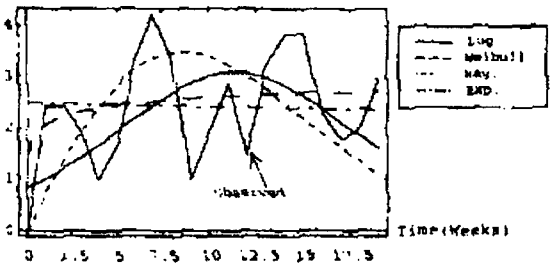


그림 2. 관찰된/예측한 TE

4.2 DS#2

이는 결함을 검출하는 패턴이다. 22일간에 걸쳐서 발견된 결함의 누적 총수는 86이고 소요된 총 디버깅 시간은 93 CPU시간이다. 모든 디버깅 데이타를 이 실험에서 사용하였다. 각각의 파라미터는 표 IV에서 보인 바와 같이 제안된 SRGM에서 MLE와 LSE로 산출하였다. 그림 3은 (8)-(10)을 이용하여 평가한 TE와 잘 적합된다. 표 IV에서는 그룹 C와 D에 대한 MSF가 다른 그룹이나 기존의 SRGM에 비하여 낮다는 것을 보여준다. 평가된 평균치 함수의 연속 곡선들은 $r_f > r_0$이기 때문에 S-형으로 보이는 변곡점을 가지고 있다. 종합해보면 TE 함수와 시변 결함 검출을 결합한 조합모델(그룹 C 또는 D)이 다른 것보다 이 DS에잘 적합된다는 것을 말해준다.

4.3 DS#3

이는 [5-6], [25]의 RADC)의 시스템 T1데이타이다.

5항 FDR 및 신뢰도경향

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

참고문헌

[1] S. Bittanti, P. Bolzem, R.Scatolini, "An introduction to software reliability modeling", vol.29, pp43-66

[2Katerina Goseva-Popstrojanova, Kishor S. Trivedi, "Failure Correlation in Software Reliability Models", IEEE Tran. on Reliability, 2000, 3, vol.49, pp37-48