결함 검출비를 고려한 소프트웨어의 품질 향상에 관한 연구

최규식
건양대학교 의공학과

A Study on the S/W Quality Improvement, Considering Fault Detection Rate

Gyu Shik Che
Dept. of BM, Konyang University

요 약

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. 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.모델들이 가중 산술, 가중 기하, 또는 가중 조화평균의 개념을 적용하여 어떻게 유도되는가를 기술한다. 그 외에도, 이러한 3개의 가중치 평균에 근거하여 유사산술의 관점으로부터 좀더 일반적인 NHPP 모델을 제안한다. 상기 3개 평균 외에 변환의 파라미터 계열을 포함한 좀더 일반적인 변환을 공식화한다.

1. 서론

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. 실제로는 FDR이 테스트 팀의 기술력, 프로그램의 크기, 소프트웨어의 테스트성에 강력하게 의존한다. 소프트웨어 개발 프로젝트[20]에 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 결함검출 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 결함예측 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. 신뢰도 모델링

TE-기초의 수학적표현은 다음과 같다.

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

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

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

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

이고, 경계조건 m(0)=0을 이용하여 방정식(2)를 풀면

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

이다.

A. TE 함수

(1)의 첫 번째 부분 은 TE 함수이다. 소프트웨어의 테스트/디버깅 단계에서는 평가 가능한 TE, 예를 들면 테스트케이스의 수, 인력, CPU시간과 같은 것이 소모된다. 그래서, 인력의 자원소모나 할당은 여러 분포로 모델화할 수 있다.

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

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

2) 웨이블형 TE함수 : 많은 논문들에 의하면 TE는 테스트 단계 전 기간에 걸쳐서 일정하지 않다는 것이다. 사

실상 순간적인 TE가 테스트 수명기간 동안 감소하여 누적 TE가 한계제한치에 접근하도록 하는 것이다.

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

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

이는 3가지 경우가 있다.

(1) g(t)=β

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

(2) g(t)=β t

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

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

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

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.

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

$$w(t) = \frac{dW(t)}{dt} = \frac{NA\alpha \cdot \exp(-\alpha t)}{[1 + A \cdot \exp(-\alpha t)]^2} \quad (11)$$

B. FDR

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

$$m(t+\Delta t) = r \cdot w(t) \cdot [a - m(t)]\Delta t \quad (12)$$

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 진단하여 변화의 효과를 평가하는는 데에 도움이 된다

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.

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 테스트 단계는 집적테스트로 진행하게 되고 그리고 시스템 테스트단계로, 그리하여 프로그래머가 잔여결함을 검출하기가 더욱 더 어려워지게 된다. 이 경우에는 초기에 FDR이 증가하고 그 다음에 감소하게 된다.

3. FDR 검토

(1)을 재배치해보면 테스트시각 t에서의 잔여 결함당 FDR을 설명할 수 있다. 이는 현재의 결함 내용에 대한 결함의 검출도를 나타낸다.

$$d(t) = \frac{dm(t)}{dt} \cdot \frac{1}{a - m(t)} = r(t) \cdot w(t) \quad (13)$$

방정식(13)은 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. w(t)=일정한 경우에 대해서 d(t)=일정하며, 이는 이러한 모델들이 동차 FDR을 가짐을 가리키는 것이다.

A. 제안 1 : r(t)에 대한 상수 FDR

r(t)가 t에서 일정하면 m(t)는 일정 FDR이다.

$$r(t) = r \quad (14)$$

$$m(t) = a \cdot [1 - \exp(-r \cdot \Delta W(t))] \quad (15)$$

(13)으로부터 테스트시각 t에서의 잔여 결함당 FDR은

$$d(t) = r \cdot w(t) \quad (16)$$

B. 제안2 : 비감소 r(t)

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

케이스 2.1 :

$$r(t) = r_0 + k \cdot \frac{m(t)}{a}, \quad k > 0 \quad (17)$$

이러한 가정 하에 FDR을 설명하기 위해 선형 모델을 사용한다.

$$m(t) = a \cdot \left(1 - \frac{r_0 + k}{r_0 \cdot \exp[(r_0 + k) \cdot \Delta W(t)] + k}\right) \quad (18)$$

$$f(t) = w(t) \cdot \left(1 - \frac{k}{\exp[(r_0 + k) \cdot \Delta W(t)] + k}\right) \quad (19)$$

케이스 2.2 :

$$r(t) = r_0 + (r_f - r_0) \cdot \frac{m(t)}{a}, \quad 0 < r_0 < r_f \quad (20)$$

(20)을 (1)에 대입하면 이는 리카티 미분방정식이 되며 그 해는

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

(21)

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 :

$$r(t) = r_0 \cdot \left(1 - \frac{m(t)}{a}\right) \quad (22)$$

(22)를 (1)에 대입하여 그해는

$$m(t) = a \cdot \left(1 - \frac{1}{r_0 \cdot \Delta W(t) + 1}\right), \quad 0 < r_0 < r_f \quad (23)$$

케이스 3.2 :

$$r(t) = r_0 + k \cdot \frac{m(t)}{a}, \quad k<0 \qquad (24)$$

(24)를 (1)에 대입하면 그해는

$$m(t) = a \cdot \left(1 - \frac{r_0 + k}{r_0 \cdot \exp[(r_0 + k) \cdot \Delta W(t)] + k}\right) \qquad (25)$$

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

$$AE = \left| \frac{M_a - a}{M_a} \right| \qquad (26)$$

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

2) MSF

$$MSF = \frac{1}{k} \sum_{i=1}^{k} [m(t_i) - m_i]^2 \qquad (27)$$

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)

5. 결론

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 설명해준다. 이 기법은 광범위한 소프트웨어시스템에 쓰일 수 있다.

참고문헌

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

[2] J. D. Musa, A. Iannino, K. Okumoto, "Software Reliability : Measurement, Prediction, Application", pp230-238, 1987 Mar.

[3] S. Yamada, H. Ohtera, H. Narihisa, "Software reliability growth models with testing- efforts", IEEE Trans. Reliability, vol. R-35, pp19-23, 1986 Apr.

[4] H. Ascher, H. Feigold, "Repairable Systems Reliability : Modeling, Inference, Misconceptions, and Their Causes", 1984, Marcel Dekker

[5] K. Okumoto, A. L. Goel, "Optimum release time for software systems based on reliability and cost criteria", J. System software, vol. 1, pp315-318, 1980.

[6] S. Yamada, S. Osaki, "Cost-reliability optimal release policies for software systems", IEEE Trans. on Reliability, vol. R-34, pp422-424, 1985 Dec.

[7] Rong-Huei Hou, Sy-Yen Kuo, Yi-Ping Chang, "Optimal release policy for hyper-geometric distribution software-reliability growth model", IEEE Trans. on Reliability, vol.45, pp646-651, 1996 Dec.

[8] Hiroshi Ohtera, Shigeru Yamada, "Optimum Software-Release Time Considering an Error-Detection Phenomenon during Operation", IEEE Trans. on Reliability, vol. 39, no.5, pp596-599, 1990 Dec.

[9] Shigeru Yamada, Shunji Osaki, "Cost-Reliability Optimal Release Policies for Software Systems", IEEE Trans. on Reliability, vol. R-34, no.5, pp422-424, 1985 Dec.