한국항행학회 논문지 제14권 제1호 2010년 2월

로지스틱 테스트함수의 불완전 디버깅에 관한 연구

A Study on the Imperfect Debugging of Logistic Testing Function

최규식*, 문명호*, 양계탁**

Gyu-Shik Che*, Myung-Ho Moon* and Kye-Tak Yang**

요 약

지난 30여년간 개발소프트웨어의 잔여결함, 결함률 및 신뢰도와 같은 신뢰도 척도를 분석하기 위해 소프트웨어의 신뢰도 성장 모델이 개발되어 왔다. 이들 대부분은 개발중 검출되는 소프트웨어의 오류가 완벽하게 수정되는 것으로 가정하였다. 즉, 이들은 테스트중에 검출되는 오류가 완벽하게 제거되는 것을 가정하여 그들의 연구를 진행해왔던 것이다. 그러나 오류를 검출하는 것이 어려울 뿐만 아니라 그 과정에서 새로운 오류가 도입되기도 하기 때문에 오류를 완벽하게 제거하기는 대단히 어렵다. 따라서 본 논문에서는 그동안 가장 보편 타당한 것으로 평가되어 왔던 웨이불형과 비교하여 로지스틱 테스트 노력함수를 적용한 불완전한 소프트웨어의 테스트 노력을 제안하여 연구 검토한다.

Abstract

The software reliability growth model(SRGM) has been developed in order to estimate such reliability measures as remaining fault number, failure rate and reliability for the developing stage software. Almost of them assumed that the faults detected during testing were eventually removed. Namely, they have studied SRGM based on the assumption that the faults detected during testing were perfectly removed. The fault removing efficiency, however, is imperfect and it is widely known as so in general. It is very difficult to remove detected fault perfectly because the fault detecting is not easy and new error may be introduced during debugging and correcting. Therefore, We want to study imperfect software testing effort for the logistic testing effort which is thought to be the most adequate in this paper.

Keywords : Testing effort function, Failure rate, The mean value function, Logistic testing function, Reliability

I. 서 론

소프트웨어 개발에서 테스트 단계는 매우 중요하며 소프트웨어 개발의 핵심 부분이다. 그동안 많은 연구에서는 테스트 단계의 테스트 자원의 소모율은 일정하다고 가정하거나 또는 그러한 테스트 노력을 고려하지도 않는 것으로 간주하고 연구를 하였다. 몇몇 참고문헌에서는 역일 테스트, 테스트 노력량, 테스트 노력에

* 건양대학교 의공학과(Bioengineering Dept. in Konyang University)
** 건양대학교 정보보호학과(Information Security Dept. in Konyang University)
· 교신저자 : 최규식
· 투고일자 : 2010년 2월 10일
· 심사(수정)일자 : 2010년 2월 15일 (수정일자 : 2010년 2월 21일)
· 게재일자 : 2010년 2월 28일

의하여 검출되는 결함의 수 사이의 관계를 설명하는 소프트웨어신뢰도성장모델(software reliability growth model ; SRGM)[1]을 제안하였다. 이러한 모델들은 테스트 기간 중에 검출되는 결함들이 완벽하게 제거 및 수정되는 것을 가정한 바탕 위에서 수립되었다.

그러나, 실제로 소프트웨어 결함 디버깅은 매우 복잡한 공정이다. 테스트 환경이 고객의 환경과 동일하지 않아서 결함이 완벽하게 제거되지 않을 수도 있다.[2][3] 그러한 과정중에 새로운 결함이 도입될 수도 있다.[3] 소프트웨어 개발팀이 문제의 결함이 최종적으로 제거되기 전에 여러 번 보고된 소프트웨어 결함이라는 것을 발견하는 것도 흔한 일이다. 어떤 결함들은 고객이 현장에서 사용할 때만 나타나는 것도 있다. 그러므로, 결함 제거 효율은 소프트웨어 신뢰도 계산에서 중요한 인자이고 소프트웨어 사업관리에도 중요하다.

비록 그동안 약간의 소프트웨어 신뢰도 연구에서 불완전 디버깅 현상에 대해서 언급을 했지만 그들 대부분은 기존의 결함을 제거하는 중에 새로운 결함이 도입될 가능성에 대해서만 고려를 하였다. 그러나, 불완전 디버깅은 검출된 결함이 불완전 제거가 된 것을 의미한다. 고엘과 오꾸모또[4]는 그들의 마코프모델에서 유사한 고찰을 하고 있다. 그들은 결함 후에 잔여결함이 확률 q로 동일하게 남아 있으며, 확률 p로서 현재의 값보다 낮아진다는 것을 가정하였다. 이는 결함제거가 항상 100%는 아니라는 것이다. 크레머[5]는 불완전 결함 제거 확률(사공정)과 결함도입(생공정) 두 개를 모두 고려하여 생산공정을 소프트웨어 신뢰도 모델링에 적용하였다. 그동안의 논문들에서는 테스트중에 결함도입 가능성만을 고려했지만 최근의 논문들에 의하면 불완전디버깅[2], 결함제거효율[3] 등을 고려한 연구논문들이 발표되고 있다. 최근에는 이러한 결함검출의 불완전성 때문에 참고문헌[6]에서는 소프트웨어의 결함검출과 이를 수정하는 예측에 관한 강인한 회귀성 신경망모델을 연구하고 있으며, 참고문헌[7]에서는 소프트웨어의 거동을 동적 PRA(probabilistic risk assessment)에 접목시키는 구조형태를 만들기 위한 연구가 수행되었다.

2항에서는 문헌에 있는 기존의 테스트노력함수를 간략하게 기술하고 제안된 로지스틱 테스트노력함수를 고찰하였다. 3항에서는 로지스틱형 SRGM에 대한 불완전디버깅의 내용을 심도 있게 서술 및 분석하였다을 연구하였다. 제4항에서는 SRGM의 파라미터는 LSE와 MLE를 이용하여 산출하는 방법 치 로지스틱 테스트노력에 고나한 파라미터를 산출하는 방법을 제시하였다. 5항에서는 로지스틱테스트노력함수를 이용하여 불완전 디버깅을 할 때의 신뢰도에 미치는 영향을 참고문헌의 데이터를 이용하여 산출하였다.

Ⅱ. 테스트 노력함수

본 논문에서는 로지스틱 테스트노력함수를 가진 소프트웨어의 신뢰도가 디버깅이 불완전할 때의 신뢰도를 검토하는 방법을 제안한다. 실제 테스트 노력 데이타가 여러 가지 소모 패턴을 나타내므로 때때로 테스트 노력 비용을 지수함수나 레일레이 곡선만으로 설명하기는 어렵다. 웨이불형 곡선은 일반적인 소프트웨어 개발 환경 하에서 데이터에 잘 맞지만, 그리고 소프트웨어 신뢰도 모델링에 널리 쓰이지만 그 차수가 m>3일 때 공칭 피크현상을 가진다. 따라서 그 대안으로 로지스틱형 테스트노력 함수를 제안하는 것이다. 이 함수는 실제 프로젝트 탐사에 의해서 보고된 바와 같이 매우 정확하다. (0, t]에서의 누적 테스트 노력 소모는

$$W(t) = \frac{N}{1 + A \cdot \exp(-\alpha t)} \qquad (2.1)$$

이다.

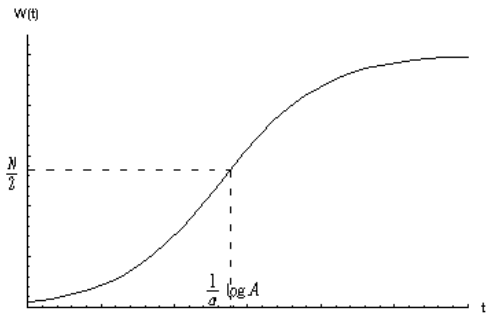

그림 2.1 로지스틱형 누적테스트노력 곡선
Fig. 2.1 Accumulative Logistic Testing

그림의 초기점에서 웨이불형 테스트 노력 함수와

비교하여 로지스틱 테스트노력 함수인 경우 $W(0) \neq 0$ 이다. 공정이 때때로 확인하기 힘든 소프트웨어 개발의 초기단계에 웨이불형 곡선과 W(t)와의 차이점이 존재한다. 그리고 적용된 테스트 노력의 양을 기록하기 위한 공식적인 계수 공정절차가 수립되지 못한 곳에서도 차이가 난다. 이 두개의 모델을 실제 결함 데이타와 잘 맞는 통계적인 테스트를 이용하여 이러한 모델 사이를 판정하는 것이 가능하다. 현재의 테스트 노력 소모량은 W(t)의 미분치로서

$$w(t) = \frac{dW(t)}{dt} = \frac{NA\alpha \cdot \exp(-\alpha t)}{[1 + A \cdot \exp(-\alpha t)]^2} = \frac{NA\alpha}{[\exp(\alpha \frac{t}{2}) + A\exp(-\frac{\alpha t}{2})]^2} \quad (2.2)$$

와 같이 표현된다. 그림에서 보는 바와 같이 이 양은 $t = \frac{1}{\alpha}\log A$ 일 때 최대치 $\frac{N\alpha}{4}$ 를 중심으로 좌우 대칭형이다.

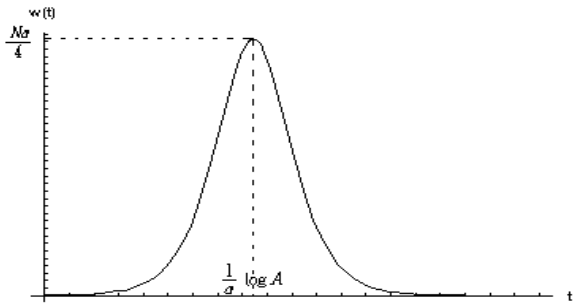

그림 2.2 로지스틱 테스트노력 소모량
Fig. 2.2 Logistic Testing Effort Consumption

본 항에서는 제안된 모델의 미분방정식에 대한 명시적인 해법도 제공하기로 한다. 결함 제거 효율과 결함 도입 현상을 포함하는 평균치함수는 아래와 같은 시스템의 미분방정식을 풀어서 구한다.[3]

$$\frac{dm(t)}{dt} = b(t)[a(t) - p \cdot m(t)] \quad (2.3)$$

$$\frac{da(t)}{dt} = \beta(t)\frac{dm(t)}{dt} \quad (2.4)$$

여기서, a(t)는 소프트웨어 초기결함 기대치의 누계 및 시각 t에서 도입되는 결함의 총기대치 합이며, p는 결함 제거 효율, β는 시각 t에서 새로운 결함이 도입될 확률이다. 개발공정을 통하여 결함의 p%가 완벽하게 제거될 수 있다는 것을 의미한다. 그러므로, (2.3)에서 m(t)는 시각 t에서 검출되는 결함의 기대치이며, 그래서 pm(t)는 성공적으로 제거하는 결함의 기대치를 명시적으로 표시한다. 기존의 모델들은 p값이 보통 1(100%)인 것으로 가정하였다.

또한, 대부분의 기존 NHPP 모델은 결함 결함비가 잔여 결함의 총수에 비례하는 것으로 가정하였다. 소프트웨어 시스템의 결함율은 어떠한 시간에 있어서도 잔여 결함의 수와 결함검출비(결함의 평균결함율로 나타냄)의 함수이다. 잔여 결함의 기대수는 아래와 같이 쓴다.

$$x(t) = a(t) - p \cdot m(t) \quad (2.5)$$

p=1일 때는 제안된 모델이 기존의 비동차포아송과정(nonhomogeneous Poisson process ; NHPP)모델로 단순화된다. 그러므로 NHPP에 근거한 신뢰도 함수는

$$R(x|t) = \exp\{-[m(t+x) - m(t)]\} \quad (2.6)$$

이며, 통계적인 예측에 의하여 현재의 결함 내용은 어느 경우이든 유한하므로 m(t)는 t에 대한 증가함수이며 m(0)=0이다. 이러한 가정 하에

$$\frac{m(t+\Delta t) - m(t)}{w(t)} = r[a - m(t)] \cdot \Delta t \ ,$$
$$r = \lim_{\Delta t \to 0}\left(\frac{m(t+\Delta t) - m(t)}{[a - m(t)]\, w(t)\Delta t}\right) \quad (2.7)$$

이며, 경계조건 m(0)=0인 조건에서 (2.7)을 풀면

$$\frac{dm(t)}{dt} + r\,w(t)m(t) = r\,a\,w(t) \ ,$$
$$m(t) = a(1 - e^{-r[W(t) - W(0)]})$$
$$= a\{1 - \exp[-r \cdot W^*(t)\}$$

$$W^*(t) = \frac{N}{1 + A \cdot \exp(-\alpha t)} - \frac{N}{1 + A} \quad (2.8)$$

이고, 결함강도는

$$\lambda(t) = \frac{dm(t)}{dt} = ar\,w(t) \cdot \exp[-rW^*(t)]$$
$$= ar\,w(t) \cdot \exp\left(-\frac{N}{1 + A \cdot \exp(-\alpha t)} + \frac{N}{1 + A}\right)$$

$$= \frac{arNA\alpha}{\left[e^{-\frac{\alpha t}{2}} + Ae^{-\frac{\alpha t}{2}}\right]^2} \cdot \exp\left[\frac{N}{1+Ae^{-\alpha t}} - \frac{N}{1+A}\right] \quad (2.9)$$

이다.

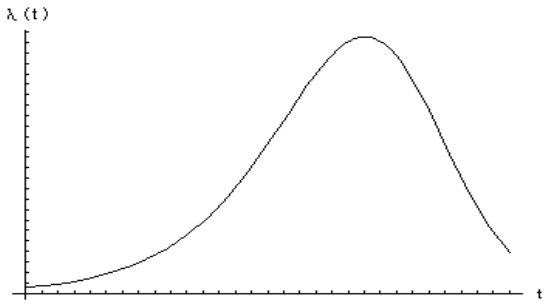

그림 2.3 결함강도함수
Fig. 2.3 Fault Intensity Function

Ⅲ. 불완전 디버깅

본 논문에서는 결함 제거 효율과 결함 도입비를 로지스틱형태의 테스트노력에 포함시키는 안을 제시한다. 결함 제거 효율은 검토자가 검토, 검사, 테스트에 의해서 제거하는 결함의 비로 정의한다. 본 항에서는 제안된 모델의 미분방정식에 대한 명시적인 해법도 제공한다. 결함 제거 효율과 결함 도입 현상을 포함하는 평균치함수는 참고문헌[4]에서 제시한 아래와 같은 시스템의 미분방정식을 풀어서 구한다.

$$\frac{dm(t)}{dt} = b(t)[a(t) - p \cdot m(t)] \quad (3.1)$$

$$\frac{da(t)}{dt} = \beta(t)\frac{dm(t)}{dt} \quad (3.2)$$

여기서, a(t)는 소프트웨어 초기결함 기대치의 누계 및 시각 t에서 도입되는 결함의 총기대치 합이며, p는 결함 제거 효율, β(t)는 시각 t에서 새로운 결함이 도입될 확률이다. 일반적으로 p≫β이며, 결함의 p%가 완벽하게 제거될 수 있다는 것을 의미한다. 그러므로, m(t)는 시각 t에서 검출되는 결함의 기대치이며, 따라서, pm(t)는 성공적으로 제거하는 결함의 기대치를 명시적으로 표시한다. 기존의 모델들은 p값이 보통 100%인 것으로 가정하였다. 결함율은 아래와 같이 표현된다.

$$\lambda(t) = m'(t) = b(t)[a(t) - pm(t)] = b(t)x(t) \quad (3.3)$$

그러므로, 평균치 함수는 아래와 같다.

$$m(t) = \int_0^t x(u)b(u)du = a\int_0^t e^{-\int_0^u [p-\beta(\tau)]b(\tau)d\tau}du \quad (3.4)$$

(3.4)에서 결함 도입 확률이 일정하다고 가정하여 β(t)=β라 하면

$$m(t) = a\int_0^t b(u)e^{-(p-\beta)\int_0^u (p-\beta)bd\tau}du = \frac{a}{(p-\beta)}[1-e^{-(p-\beta)bt}] \quad (3.5)$$

이며, 이를 이용하여 신뢰도를 구하면

$$m(t+x) - m(t) = \frac{a}{p-\beta}[1 - e^{-(p-\beta)b(t+x)}] - \frac{a}{p-\beta}[1-e^{-(p-\beta)bt}] = \frac{a}{p-\beta}e^{-(p-\beta)bt}[1 - e^{-(p-\beta)bx}] = m(x)e^{-(p-\beta)bt}$$

이므로,

$$R(x|t) = \exp[-m(x)e^{-(p-\beta)bt}] \quad (3.6a)$$
$$= \exp\left\{-\frac{a}{p-\beta}[1-e^{-(p-\beta)bx}]e^{-(p-\beta)bt}\right\}$$

이고, 원하는 목표신뢰도를 Ro라 하면

$$R_o = \exp[-m(x)e^{-(p-\beta)bT}\} \quad (3.7)$$

로 된다. 그런데,

$$R(x|0) = \exp[-m(x)]$$
$$= \exp\left\{-\frac{a}{p-\beta}[1-e^{-(p-\beta)bx}]\right\}$$

로부터

$$T = \frac{1}{(p-\beta)b}\ln\frac{\ln\frac{1}{R(x|0)}}{\ln\frac{1}{R_o}} \quad (3.8a)$$

가 된다. 여기서, T*=T은 테스트 후 목표신뢰도를 만족하여 고객에게 인도해도 좋은 시간을 말한다. 참고로 기존의 문헌에서처럼 p=1, β=0인 경우에는

$$R(x|t) = \exp[-m(x)e^{-bt}]$$
$$= \exp\{-a[1-e^{-bx}]e^{-bt}\} \quad (3.6b)$$

$$T = \frac{1}{b}\ln\frac{\ln\frac{1}{R(x|0)}}{\ln\frac{1}{R_o}} \quad (3.8b)$$

와 같이 표현된다. 본 항에서 디버깅 확률과 결함도입 확률을 도입하여 상기의 공식들을 일반화하여 다시 정리하면 다음과 같다.

- 평균치 함수

$$m(t) = \frac{a}{p-\beta}(1-e^{-B(t)}) \quad (3.9)$$

- 신뢰도

$$R(x|t) = \exp\left\{-\frac{a}{p-\beta}[1-e^{-rW(x)}]\cdot e^{-rW^*(t)}\right\}$$
$$= \exp\left\{-\frac{a}{p-\beta}[1-e^{-B(x)}]\cdot e^{-B^*(t)}\right\} \quad (3.10)$$

- 테스트 노력 함수

$$W^*(t) = \frac{1}{r}\ln\frac{a(1-e^{-rW^*(x)})}{(p-\beta)\ln\frac{1}{R_0}}$$
$$= \frac{1}{r}\ln\frac{a(1-e^{-B^*(x)})}{(p-\beta)\ln\frac{1}{R_0}} \quad (3.11)$$

이와 같은 이론을 로지스틱의 경우로 적용하면 다음과 같다.

$$B(t) = \frac{rN}{1+Ae^{-\alpha t}},\quad B(0) = \frac{rN}{1+A}$$

- 평균치함수

$$m(t) = \frac{a}{p-\beta}\cdot \left\{1-e^{-(p-\beta)rN(\frac{1}{1+Ae^{-\alpha t}}-\frac{1}{1+A})}\right\} \quad (3.12)$$

- 신뢰도

$$R(x|t) = \exp\left\{-\frac{a}{p-\beta}e^{-rN(\frac{1}{1+Ae^{-\alpha t}}-\frac{1}{1+A})}\cdot[1-e^{-rN(\frac{1}{1+Ae^{-\alpha x}}-\frac{1}{1+A})}]\right\} \quad (3.13)$$

- 발행시각

$$T = T^* = -\frac{1}{\alpha}\ln\frac{1}{A}\cdot \left[\frac{rN}{\frac{rN}{1+A}+\ln\frac{a(1-e^{-rN(\frac{1}{1+Ae^{-\alpha x}}-\frac{1}{1+A})})}{\ln\frac{1}{R_0}}}\right] \quad (3.14)$$

Ⅳ. 파라미터 산출법

위의 각 경우에서 정의된 테스트노력함수에서 파라미터 N, A, α, β는 최소자승법(LSE)으로 산출한다. MLE는 한 집합의 동시방정식을 풀어서 파라미터를 산출하며, s-신뢰 구간을 구동하는데 더 좋은 방법이다. 그러나, 그 방정식이 너무너무 복잡하므로 보통은 수치 해석적으로 푼다. LSE는 실제로 관찰/획득한 것과 예측한 것 사이의 차이를 제곱해서 더한 총값을 최소화하는 것이다. LSE는 중간 크기의 표본에 최적인 것으로 알려지고 있으며, 최적점 산출을 제공한다. 최소자승법을 적용하기 위한 산출 공식S(N, A, α)은 다음과 같다.

$$S(N,\ A,\ \alpha) = \sum_{k=1}^{n}[W_k - W(t_k)]^2 \quad (4.1)$$

Wk = (0, t] 기간동안 실제로 소요되는 누적 테스트 노력

W(tk) = 테스트 노력 함수에 의해서 산출된 누적 테스트 노력

S를 N, A, α에 관하여 미분하여 그 편미분치를 0으로 놓으면 그리고 이 항들을 재정비하여 이러한 형태의 비선형 최소 자승 문제를 푼다.

$$S(N,A,\alpha)=\sum_{k=1}^{n}\left\{W_k-\frac{N}{1+Ae^{-\alpha t_k}}\right\} \quad (4.2)$$

이를 N으로 편미분하여

$$\frac{\partial S}{\partial N}=2\sum_{k=1}^{n}\left\{W_k-\frac{N}{1+Ae^{-\alpha t_k}}\right\}\frac{1}{1+Ae^{-\alpha t_k}}=0$$

즉

$$N=\frac{\sum_{k=1}^{n}\frac{W_k}{1+Ae^{-\alpha t_k}}}{\sum_{k=1}^{n}\frac{1}{\{1+Ae^{-\alpha t_k}\}^2}} \quad (4.3)$$

을 만족하는 N값을 구한다. 그리고

$$\frac{\partial S}{\partial N}=2\sum_{k=1}^{n}\left\{W_k-\frac{N}{1+Ae^{-\alpha t_k}}\right\}\frac{e^{-\alpha t_k}}{\{1+Ae^{-\alpha t_k}\}^2}=0 \quad (4.4)$$

를 만족하는 A값을 구한다. 또한

$$\sum_{k=1}^{n}\left\{W_k-\frac{N}{1+Ae^{-\alpha t_k}}\right\}\frac{t_k e^{-\alpha t_k}}{\{1+Ae^{-\alpha t_k}\}^2}=0 \quad (4.5)$$

을 만족하는 α값을 구한다.

$W^*(t_k)=\frac{N}{1+Ae^{-\alpha t_k}}-\frac{N}{1+A}$ 에서

$$a=\frac{m_n}{1-e^{-rN(\frac{1}{1+Ae^{\alpha t_n}}-\frac{1}{1+A})}} \quad (4.6)$$

를 구한다.

$$aW^*(t_n)\phi_n=\quad=\sum_{k=1}^{n}\{(m_k-m_{k-1})\cdot \frac{-W^*(t_{k-1})\phi_{k-1}+W^*(t_k)\phi_k}{\phi_{k-1}-\phi_k} \quad (4.7)$$

에서 a를 구하게 된다. 그리고

$$aN(\frac{1}{1+Ae^{-\alpha t_n}}-\frac{1}{1+A})e^{-rN(\frac{1}{1+Ae^{-\alpha t_n}}-\frac{1}{1+A})}$$
$$=\sum_{k=1}^{n}(m_k-m_{k-1})\cdot$$
$$\left[\left\{-N(\frac{1}{1+Ae^{-\alpha t_{k-1}}}-\frac{1}{1+A})\cdot e^{-rN(\frac{1}{1+Ae^{-\alpha t_{k-1}}}-\frac{1}{1+A})}\right\}\right.$$
$$+\left\{N(\frac{1}{1+Ae^{-\alpha t_k}}-\frac{1}{1+A})\cdot e^{-rN(\frac{1}{1+Ae^{-\alpha t_k}}-\frac{1}{1+A})}\right\} \quad (4.8)$$
$$\left./\left\{e^{-rN(\frac{1}{1+Ae^{-\alpha t_{k-1}}}-\frac{1}{1+A})}-e^{-rN(\frac{1}{1+Ae^{-\alpha t_k}}-\frac{1}{1+A})}\right\}\right]$$

에서 r을 구한다.

Ⅴ. 적용 예

그림 3.1은 참고문헌[3][9]에 의해 제시된 a=142, b=0.1246, x=0.1, p=0.7, β=0.012, Ro=0.95, TLC = 500인 경우에 대해서 테스트노력이 일정하다고 가정하여 결함 제거 확률과 결함 도입 확률을 고려한 경우와 그렇지 않은 경우의 신뢰도를 비교한 그래프이다. 실선은 결함 제거가 완전한 경우이고 점선은 불완전한 경우이다. 여기서, a는 소프트웨어의 개발 초기부터 소프트웨어 내에 존재하고 있는 결함의 총 수, b는 결함검출율로서 일반적인 경우 테스트기간중의 결함검출율이다. 그리고 x는 최종결함 수정 후 다음 결함이 검출될 때까지의 시간이며, p는 결함을 완전하게 수정할 확률, β는 결함수정 중 새로운 결함이 도입될 확률, Ro는 목표신뢰도, TLC는 소프트웨어의 운영 수명이다.

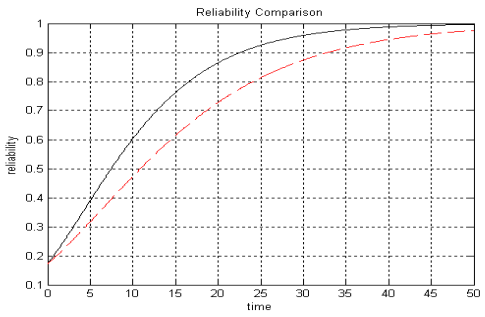

그림 3.1 신뢰도 비교
Fig. 3.1 Reliability Comparison

이 그림에 의하면 결함제거 효율이 완전하여 결함발견 즉시 수정이 완벽하고 결함 수정중 새로운 결함이 도입될 가능성이 없는 경우에는 목표신뢰도 95%에 이르는 시간이 28.4이다. 한편, 결함제거 효율을 가정하여 그 값이 70%라 하고 따라서 결함 도입 확률을 1.2%로 가정한 경우에는 목표신뢰도에 이르는 시간이 41.3으로 크게 증가한다. 따라서, 결함 제거를 완벽에 가깝도록 한다면 목표신뢰도에 이르는 시간 곧 인도시간이 크게 단축될 수 있다는 것을 알 수 있다.

VI. 결 론

소프트웨어의 개발 단계에서 테스트 및 수정단계를 거칠 때 실제로 결함 제거 효율은 통상 불완전하며, 이러한 사실은 그동안 널리 알려져 있다. 소프트웨어 결함은 그것을 찾아내는 것도 힘들지만 수정중에 새로운 결함이 도입될 수도 있기 때문에 검출된 결함이 완벽하게 제거되기는 어렵다. 따라서, 결함 제거 효율은 개발중인 소프트웨어의 신뢰도 성장이나 테스트 및 수정 비용에 영향을 크게 미친다. 이는 소프트웨어 개발의 모든 과정에서 매우 유용한 척도로서 개발자가 디버깅 효율을 평가하는데 크게 도움이 될 뿐더러, 추가로 소요되는 작업량을 예측할 수 있게 해준다. 그러므로 개발 소프트웨어의 SRGM과 비용면에서 불완전 디버깅의 영향을 연구하는 것은 매우 중요하다고 할 수 있으며, 이는 최적 인도 시각이나 운영 예산에도 영향을 줄 수 있다.

본 논문에서는 소프트웨어의 디버깅이 완전하지 않으며, 이 때문에 디버깅중 새로운 결함이 도입될 수도 있다는 제안 하에 보편적으로 사용되는 신뢰도 및 비용모델을 불완전 디버깅 범위로 확장하여 연구하였다. 그간의 기존 논문들을 참고하여 결함 제거 확률과 수정중 결함도입 확률을 고려한 이론을 전개 및 수립하였다. 이러한 알고리즘을 확인 및 실증하기 위해 그간 몇 개의 참고문헌에서 수집된 실제 데이타에 의해서 소프트웨어의 결함 제거 확률을 변화시켰을 때의 각각에 대해서 목표신뢰도에 이르는 시간, 최저 수정 비용에 이르는 시간을 계산하였다.

참 고 문 헌

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

[2] Min Xie, Bo Yang, " A study of the effect of imperfect debugging on software development cost", *IEEE Trans. on Software Eng., vol.29, no.5*, pp471-473, 2000.5

[3] X. Zhang, X. Teng, H. Pham, "considering fault removal efficiency in software reliability assessment", *IEEE Trans. on Systems, man, and cybernetics, vol.33, no.1*, pp114-120, 2003.1

[4] A.L. Goel, K. Okumoto, "A Markovian model for reliability and other performance measures of software systems", *Proc. AFIPS Conf.,* pp770-774, 1979.6

[5] W. Kremer, "Birth-death and bug counting", *IEEE Trans. on Reliability, vol.R-32, no.1*, pp37-47, 1983

[6] Q. P. Hu, M. Xie, S. H. Ng, G. Levitin, "robust recurrent neural network modeling for software fault detection and correction prediction", *Relia. Eng. and Safety, vol.92*, pp332-340, 2007.

[7] Dongfeng Zhu, Ali Mosleh, Carol Smidis, "A framework to integrate software behavior into dynamic probabilistic risk assessment", *Relia. Eng. and Safety, vol.92*, pp1733-1755, 2007.

[8] Chin-Yu Huang, Sy-Yen Kuo, "Analysis of Incorporating Logistic Testing-Effort Function into Software Reliability Modeling", *IEEE Trans. on Reliability, vol.51*, pp261-270, 2002, Sep.

[9] Min Xie, Bo Yang, " A study of the effect of imperfect debugging on software development cost", *IEEE Trans. on Software Eng., vol.29, no.5,* pp471-473, 2000.5

최 규 식 (崔圭植)

1973년 서울대학교 공과대학 전기공학과(공학사)
1983년 뉴욕공과대학 전기공학과 (공학석사)
1993년 명지대학교 전기공학과 (공학박사)
1978년 ~1993년 한국전력기술 중앙연구소 책임연구원
1993년 ~ 현재 건양대학교 의공학과 교수

관심분야 : 데이터통신. 무선랜 분야

문 명 호 (文明鎬)

숭실대학교 공과대학 전자공학과(학사)
The Catholic Univ. of America(석사)
건국대학교(박사학위)
미국 Radiation System, Inc.에서 근무
현재 : 건양대학교 정보통신학과
부교수

관심분야 : 무선통신 시스템

양 계 탁 (梁桂卓)

1984년 2월: 연세대학교 수학과 (이학사)
1986년 2월 : 연세대학교 수학과 (이학석사)
1992년 2월 : 연세대학교 수학과 (이학박사)
1993년 3월 : 건양대학교 정보보호학과 교수

관심분야 : 어플리케이션 보안, 보안 솔루션, 암호