대형 시스템 개발을 위한 시험능력을 고려한 소프트웨어 신뢰도 성장 모델

정회원 이 재 기*, 이 재 정**, 종신회원 남 상 식***

Software Reliability Growth Model with the Testing Effort for Large System

Jae-ki Lee*, Jae-jeong Lee**, Sang-sik Nam*** ***Regular Members***

요 약

기존에 제안된 소프트웨어 신뢰도 성장모델(SRGM)들은 결함이 발견됨과 동시에 해결되는 것을 전제로 한 완전디버깅(PD: perfect debugging)을 추구한다. 그러나 실제 프로젝트 수행시 검출된 결함(에러)들은 일정한 시간이 지난 후 해결(제거)되거나 새로운 결함이 소프트웨어 내에 삽입되는 불완전디버깅(ID: imperfect debugging) 상태에 놓이게 된다. 이러한 문제점들을 보완하기 위한 방안으로 본 논문에서는 소프트웨어의 고장을 발견/해결하는데 투입된 시험능력(test-effort)을 고려하여 이를 정형화된 모델로 발전시켜 실제 상황에 가까운 소프트웨어의 신뢰도를 평가하였다.

Key Words : SRGM, 시험능력(test-effort), 결함, 고장, 디버깅, 고장데이터, NHPP

ABSTRACT

Most of the proposed SRGMs are required to perfect debugging based on removal of defect as soon as the detection of defects in system tests. But the detected defects are corrected after few days as a fixed time or induced new fault in software under the imperfect debugging environments. Solving these problems, we discussed that the formal software reliability model considered testing-effort for the fault detection and correction of software defects, and then using this model we have estimated of the software reliability closed to practical conditions.

Ⅰ. 서 론

교환시스템, 테라 라우터 등 대형시스템 개발은 분산구조를 가지고 추진되며, 투명하고 일관성 있게 추진되어야 한다. 특히, 이러한 시스템은 사용자에게 인도되기까지 개발 기간이 2-5년 정도 걸리고 개발 인력도 200-300명 이상이 투입되는 대형 프로젝트로 복잡한 개발과정이 투명하게 조명되어야 한다.

시스템 기능들은 서브시스템들과 함께 시스템 전체의 요구사항에 부합되게 매핑(mapping)되고 구현되어야 한다. 일반적으로 이러한 대형 시스템을 지탱하는 소프트웨어는 여러 개의 서브시스템으로 구성되어 개발자에 의해 변경(modify or change), 확장(add)되어 배포된다.

요구사항의 변경(change requirement)은 오랜 시스템 개발 기간 중에 흔히 일어나며, 이러한 일들은

* 한국전자통신연구원 BcN시험기술팀(jkilee@etri.re.kr)
** 한국전자통신연구원 BcN시험기술팀(jjlee@etri.re.kr)
*** 한국전자통신연구원 BcN시험기술팀(ssnam@etri.re.kr)
논문번호 : KICS2005-04-137, 접수일자 : 2005년 4월 12일

복잡한 시스템 개발 계획이나 개발일정 조정에 따라 서브시스템별 또는 배포될 시스템에 따라 추가/삭제된다. 더 복잡한 요소는 개별적인 서브시스템별 배포를 위한 종합이나 검증에 따라 달라지며, 이러한 대형시스템은 높은 신뢰도와 품질을 보장하여야 한다.

최종 소프트웨어 배포 후 소프트웨어의 결함은 신규 소프트웨어나 소프트웨어 내에 잠재하고 있는 결함에 의해 증가하게 된다. 또한 부수적으로 많은 양의 소프트웨어에 대해 결함을 검출해 내는 회귀시험(regression test)을 수행하여야 한다.

이러한 분산 개발환경을 사용자와 개발자 측면으로 요약해보면 사용자는 경쟁 관계에 있는 시스템의 인도 시기나 높은 신뢰도를 갖는 경쟁력 있고 현실적인 가치가 있는 시스템을 인도 받기를 원할 것이다. 이러한 일은 소프트웨어 신뢰도 성장모델로 검증이 가능할 것이다. 결과적으로 사용자(구매자) 입장에서는 비용부담이나 시스템의 신뢰도와 함께 모든 신규 경쟁 제품에 대한 배포시기에 따라 빈번하게 현실성 있는 가격 경쟁력에 많은 관심을 보일 것이다.

ANSI의 정의[8]에 따르면 "소프트웨어 신뢰도는 주어진 동작 조건하에서 일정시간 동안 에러 없이 소프트웨어가 동작할 확률" 로 정의하고 있다. 소프트웨어 신뢰도 성장모델은 시험에 소요된 시간이나 사용된 소프트웨어에 대해 검출된 축적 결함수(cumulative defect)의 수학적 관계를 제공한다. 이것은 소프트웨어 운용단계에서 시험기간 동안 소프트웨어 신뢰도를 확인하는데 이용되고 있다.

소프트웨어 시험은 소프트웨어 동작 및 결과에 대해 예기치 않은 동작 등을 포함하고 있으며, 성공적인 시험은 보이지 않는 에러들의 실체를 도출하는 것을 포함한다. 결함의 위치(fault location)를 추적하거나 검출해 내는 과정을 디버깅 절차(debugging processing)라 부르며, 고장발생의 원인이나 검출 과정은 소프트웨어 신뢰도 측정이나 고장 내용의 심각성(severity)을 판단하는데 이용된다. 소프트웨어 신뢰도 모델은 소프트웨어의 품질을 평가하는 도구로 개발일정이나 시험단계에 대한 상태파악, 변경이력 감시 등 신뢰도 성능 파악에 활용된다.

고장 개수나 실행시간에 의존한 다양한 소프트웨어 신뢰도 성장모델은 이미 오래전부터 여러 연구논문이나 저서를 통해 논의되어 왔다.[5,8,10] 이러한 모델들은 소프트웨어 신뢰도나 고장의 내용을 예측하는데 이용되었으며, 동시에 소프트웨어 배포시기를 결정하는 것에도 이용되었다. 축적된 고장 경향곡선은 자연스럽게 지수형(exponential)이나 S-자형(S-Shaped)을 띠었다.[1,6,11]

현재 사용되고 있는 대표적인 신뢰도 성장모델들은 그러한 곡선중의 한 형태를 띠며, 그밖에 다른 일부 모델들도 신뢰도성장모델의 파라미터 값에 의존하는 형태를 띠고 있다. 대부분의 신뢰도 성장모델들은 고장의 발생이나 고장의 격리/제거(isolation/remove)를 구분하지 않고 있다. 그러나 실제 고장 제거 과정은 고장의 보고 후 고장의 격리(fault isolation)를 수행한다. 즉, 고장의 제거는 2단계로 수행하는데 첫 번째 단계는 고장 분석팀에 의한 고장의 격리 과정과 두 번째 단계는 개발자(프로그래머)로 구성된 또 다른 팀에 의한 고장 원인의 제거 과정으로 양 팀에 대한 자원의 배분은 반드시 관리되어야 한다.

Yamada et al.[12]에 의한 S-Shaped SRGM(Software Reliability Growth Model)은 고장발생과 제거과정을 분리하여 2단계로 나누어 시간 간격에 따른 개념을 고려하였다. 이때 결함 제거비(FRR : Fault Removal Rate)는 일정하다고 가정하였다.

본 논문에서는 분산시스템에서 아래와 같은 가정을 도입하여 시험능력을 고려한 SRGM을 개발한다.

(모델의 가정 : Assumptions)

① 시험능력은 결함의 심각성에 따라 좌우된다. 즉, 신규 추가되는 소프트웨어에서 발생되는 결함은 심각도(severity)가 다르다.

② 신규 개발되는 소프트웨어의 결함 제거비는 시험능력에 따라 해결시간이 좌우된다. 이러한 데이터는 시험시간을 설정하는 기초 자료로 활용된다.

시험단계에서의 결함 해결 비율(FER : Fault Exposure Ratio)은 프로그램 규모나 소프트웨어 시험용이성, 시험기술 등에 매우 좌우되기 때문에 시험능력이나 시간에 대한 함수로 결함 제거비(fault remove ratio)를 다룬다. 이러한 가정은 다양한 신뢰도 성장곡선을 얻을 수 있으며, 좀 더 유연한 모델로 발전시킬 수 있다. 즉, 시험능력(test-effort)과 고장 시간 간격(MTBF)의 함수를 고려한 소프트웨어 신뢰도 성장모델을 제안할 수 있다.

시험기간 동안 시험자나 수행시간등과 같은 시험자원(resource)이 투입되고 자연스럽게 시험능력의 소모에 의한 결함의 발견(detection)과 제거(remove)

가 이루어진다. 시험능력의존형(TeDM: Testing-effort dependant model) 모델에 의한 시험시간은 오래전부터 연구되어 왔으며[4,5,13], 시험능력 소모량과 시험 시간과의 관계는 지수형(Exponential) 또는 레일리히(Rayleigh) 곡선을 띠는 것으로 밝혀졌다.

지수형 곡선은 시험시간에 대해 시험능력을 일정한 비율로 사용될 때 나타나며, 그렇지 않은 경우에는 Rayleigh 곡선을 띤다. 또한 시험능력에 대한 표현이 로지스틱(Logistic) 또는 웨이블(Weibull) 함수 형태를 띠는 경우도 있다.[3,6,8,11]

본 논문에 제안된 소프트웨어 신뢰도 성장모델은 시험능력이 고려된 NHPP 모델의 가정을 기본으로 한 성장모델로 2개의 소프트웨어 결함 데이터 군(群)에 의해 신뢰도를 평가한다.

-표기(Notations)

α: total expected error in software (before start of the system test)
b: failure rate(normal failure rate)
d: initial failure rate
α: shape parameter(weibull-function)
β: scale parameter(weibull-function)
γ: $\frac{d}{b}$ (fault detection rate)
W_R: working function(for test-effort)
$x(t)$: test-effort
y_i: detection failures(at t_i)
$m(t), H(t)$: mean function
MLE: Maximum Likelihood Estimation
LSE: Least Square Estimation

Ⅱ. 시험능력 모델(Testing Effort Model)

소프트웨어 신뢰도 성장모델(SRGM)은 고장 제거에 소요된 시간에 의해 해석된다. 이미 제안된 많은 모델들이 서로 다른 가정과 시험환경에 의한 신뢰도 평가 자료를 제시하고 있다. 본 논문에서 제안한 모델은 시험능력 의존에 따라 계산된 시간을 적용한다. 시험자원(testing resources)은 모든 소프트웨어 프로젝트에 적용되는 시험의 질을 측정하는데 이용되고 있으며[9,10], 이것을 다시 세분하면 아래와 같다.

(1) 시험인력(test manpower)
(a) 고장 분석자(fault analyser)
(b) 고장 해결자(fault remover)

(2) 시험 수행시간(computing time)
(a) 단위시간(unit time or calendar time)
(b) 실행시간(execution time)

소프트웨어 시험의 핵심 기능인 시험자원(man-power)은 테스트케이스의 운용(test-case operation)과 요구규격에 의한 시험 수행 결과의 비교이다. 이때 요구규격에 위반되는 사항은 고장으로 간주되며, 개발자에 의해서 발생된 고장은 원인이 분석되어지고 제거 된다. 또한 고장에 대한 분류와 수정에 대한 소요시간이 자동으로 계산(카운트) 된다.

시험능력 투입 현황을 기술하기 위해서 지수형, 로지스틱, 레일리히, 웨이블 함수 등이 사용되어 오고 있으며, 이러한 모델들은 "시험자원을 활용한 시험능력 투입 비율"을 고려한 가정으로부터 파생된 것으로 볼 수 있다.[15,16] 즉,

$$\frac{dX(t)}{dt} = c(t)[\alpha - X(t)] \quad (1)$$

$c(t)$: time dependent rate (testing resource consumed)

초기 상태 $X(t)=0$을 적용하여 함수 (1)의 해를 구하면 아래와 같이 식 (2)를 얻는다.

$$X(t) = \alpha\left[1 - \exp\int_0^t c(k)dk\right] \quad (2)$$

$c(t)=\beta$ 일정하면

$$X(t) = \alpha(1 - e^{-\beta t}) \quad (3)$$

만약 $c(t)=\beta.t$이면 식(1)은 Rayleigh 곡선을 띠며, 식 (4)와 같이 표현된다.

$$X(t) = \alpha(1 - e^{-\beta\frac{t^2}{2}}) \quad (4)$$

Huang et al.[4]은 로지스틱 시험능력함수를 고려한 소프트웨어 신뢰도 성장모델을 제안했는데 여러 고장데이터 군(group data)을 적용하여 좋은 결과를 제공하였다. 시험기간 (0,t)에서 소요된 축적된 시험능력은 아래 식 (5)의 함수 형태를 띤다.

$$X(t) = \frac{\alpha}{1 + \beta e^{-ct}} \quad (5)$$

α, β, c: constant

Weibull 함수를 이용한 좀 더 유연하고 일반적인 시험능력함수 형태는 아래 식 (6)과 같이 표현된다.

$$X(t)=\alpha\left[1-e^{-(\frac{t}{c})^{\beta}}\right] \quad (6)$$

α, β, c constant

Ⅲ. 소프트웨어 신뢰도 성장 모델(SRGM)

근년(近年)에 일본의 Yamada et al.[14] 등은 NHPP (Non-Homogeneous Poisson Process) 모델에 근거한 SRGM을 제안하였다. 이 모델은 분산 소프트웨어의 구조를 갖는 n개의 재사용 모듈과 m개의 신규 모듈이 적용된 시스템을 가정하여 축적된 고장 데이터가 지수형 곡선을 띠는 여러 개의 소프트웨어 컴포넌트로 구성된 시스템을 고려하였다.

한편 신규 개발된 소프트웨어 컴포넌트들에 대한 축적된 고장데이터는 S-Shaped 성장 곡선을 띠는 것으로 조사되었다. 이러한 분산 개발 환경을 갖는 NHPP 모델에 근거한 성장모델의 가정하에서 신뢰도 함수를 표현하면

$$H(t)=\alpha\left[\sum_{i=1}^{n} p_i(1-e^{-b_i t})+\sum_{j=1}^{m} p_{n+j}[1-(1+b_{n+j}t)e^{-b_{n+j}t}]\right] \quad (7)$$

α 총 기대 결함수

b_i i 번째 소프트웨어 모듈에 대한 소프트웨어 고장 검출율

p_i 총 시험 투입비에 대한 weight factor

$(\Sigma p_i = 1, P_i > 0)$

3.1 모델의 가정(Model Assumptions)

시험능력의 영향은 여러 신뢰도 성장모델을 포함하며, 분산 환경을 고려한 시험능력의존형 소프트웨어 신뢰도 성장모델은 유한개의 재사용 모듈(reuse module)과 신규개발 모듈(new development module)로 구성된 소프트웨어 시스템으로 고장 간 시간(MTBF: mean time to between failures)과 고장의 분석/해결 과정을 고려하고 있다.[4,5,8,13,17] 이 모델의 세부 가정들은 아래와 같다.

-가정(Assumptions)

(a) 고장의 발견/해결 현상은 NHPP에 의해 모델링 된다.

(b) 소프트웨어는 소프트웨어에 잔존하고 있는 결함에 의해 고장을 일으킨다.

(c) 소프트웨어 시스템은 유한개의 재사용 및 신규 개발로써 서브시스템으로 구성된다.

(d) 재사용 서브시스템의 소프트웨어 신뢰도 성장은 일정하다(단, 신규 개발 서브시스템은 일정하지 않다)

(e) 각 고장에 대한 고장시간은 관측되어야 하고 고장을 해결하기 위한 시험능력은 즉시 투입되는 것으로 가정한다. 고장의 위험도가 높은 경우에는 많은 시간이 소요된다.

(f) 고장 격리 및 제거시 신규 에러가 삽입되지 않는다.(완전 디버깅을 고려함)

(g) 재사용 모듈에 대해 결함 제거 비율은 일정하다고 가정

(h) 신규개발 모듈에 대한 결함 제거 비율은 로지스틱 함수에서 얻어진 데이터에 따른다.

(i) 소프트웨어에 잔존하고 있는 에러에 의해 발생되는 결함은 일정하다고 간주

(j) 시험강도(testing intensity)에 따라 결함 격리/제거 비율은 관측된 결함수에 비례한다.

3.2 모델의 공식화(Model Formulation)

가. 재사용된 서브시스템 "P"에 관련된 고장들에 대한 모델화

고장이 발견됨과 동시에 제거되는 단순한 고장들에 대한 모델링으로 재사용된 서브시스템의 "p"의 i 번째 고장들에 대해서 1 단계 프로세스로 식 (8)과 같이 모델화 할 수 있다.

$$\frac{\frac{d}{dt}m_{ir}(t)}{x(t)}=b_i(a_i-m_{ir}(t)) \quad (8)$$

식(8)에서 모델링된 1단계 프로세스는 고장의 관측, 격리 및 제거의 과정을 나타내며, 경계조건 $m_{ir}(t=0)=0$을 적용하여 식 (8)의 미분방정식의 해를 구하면 식 (9)를 얻는다.

$$m_{ir}(t)=a_i(1-e^{-b_i X(t)}) \quad (9)$$

나. 신규개발 서브시스템 "q"의 고장들에 대한 모델화

재사용된 서브시스템과 비교시 시험능력(노력)이 많이 소요되는 신규개발 서브시스템 "q"의 j 번째 고장들에 대한 해석으로 고장분석팀(또는 시험팀)에 의해 장시간 고장 원인을 파악하고 또 고장을 제거하는데 많은 노력이 소요되는 과정으로 즉, 고장을

제거하는 과정을 기술하면 아래와 같이 2 단계로 모델링 할 수 있다.

$$\frac{\frac{d}{dt} m_{jf}(t)}{x(t)} = b_j(a_j - m_{jf}(t)) \quad (10)$$

$$\frac{\frac{d}{dt} m_{jr}(t)}{x(t)} = b_j(t)(m_{jf}(t) - m_{jr}(t)) \quad (11)$$

여기서

$$b_j(t) = \frac{b_j}{1+\eta e^{-b_j X(t)}} \quad (12)$$

2개의 단계중 첫 번째 단계는 고장의 발견과정(관측과정)을 묘사한 것으로 식 (10)으로 표현된다. 두 번째 과정은 식 (11)로 지연된 고장제거 과정을 설명한다. 이런 과정을 거치는 동안 고장 제거비(fault removal rate)는 시간에 종속되는 것으로 가정한다. 이러한 가정을 거치는 이유는 개발자 각 개인의 디버깅 능력과 밀접한 관계가 있기 때문이다.

개별 고장 제거에 대한 간파(fault removal insight)는 로지스틱 함수로 불리는 식 (12)와 같이 표현할 수 있으며, 관측된 고장으로부터 정보를 얻을 수 있다. 경계조건 $t=0, x(t)=0, m_{jf}(t=0)=0, m_{jr}(t=0)=0$를적용하여 위의 미분방정식의 해를 구하면 식 (13)을 얻는다.

$$m_{jr}(t) = a_j \frac{[1-(1+b_j X(t))e^{-b_j X(t)}]}{1+\eta e^{-b_j X(t)}} \quad (13)$$

3.3 총 고장제거 과정에 대한 모델링

제안된 모델은 p개의 재사용 모듈과 q개의 신규개발 모듈에 대한 소프트웨어 시스템으로서 NHPP에 근거하여 수학적으로 모델화 하여 평균치 함수로 나타내면 식 (14)와 같이 표현할 수 있다.

$$m(t) = \sum_{i=1}^{p} m_{ir}(t) + \sum_{j=1}^{q} m_{jr}(t),$$

$$m(t) = \sum_{i=1}^{p} a_i(1-e^{-b_i X(t)}) + \sum_{j=p+1}^{q} a_j \frac{[1-(1+b_j X(t))e^{-b_j X(t)}]}{1+\eta e^{-b_j X(t)}} \quad (14)$$

더 나아가 각각 임의의 재사용 모듈과 신규개발 모듈을 식 (14)에 적용하면 단순한 소프트웨어 시스템으로 모델화 할 수 있다. 이때 $a = a_1 + a_2 + a_3 + \cdots\cdots + a_n, (b_1 = b_2, b_3 = b_4, \cdots\cdots)$로 가정할 수 있다.

3.4 Yamada's 변형된 모델[14]

모델의 적합도 판정을 비교하기 위해 식 (14)로부터 얻어진 경험 데이터를 토대로 Yamada et al. 의 신뢰도 성장모델에 대해 평균치 함수로 다시 정리하면 아래 식 (15)와 같이 표현 할 수 있다.

$$H(t) = a[\sum_{i=1}^{n} p_i(1-e^{-b_i X(t)}) + \sum_{j=1}^{m} p_{n+j}(1-(1+b_{n+j}X(t))e^{-b_{n+j}X(t)})] \quad (15)$$

Ⅳ. 파라미터 추정(Parameter estimation)

고장시간에 대한 시험능력 함수 파라미터는 시험노력 데이터를 이용하여 추정할 수 있으며, 최적의 모델 fitting을 위해서 이 함수는 SRGM 파라미터 추정을 위해서 필요하다.

시험능력 데이터(testing-effort data)는 시험노력 $x_k(x_1 < x_2 < x_3 \cdots\cdots < x_n)$의 형태로 주어지고 소요시간은 $(0, t_j]:[j=1,2,3,4,\cdots\cdots,n]$으로 표시된다.

시험능력모델 파라미터 α, β는 최소자승법(LSE : Least Square Estimation)을 이용하여 추정할 수 있다. 즉,

$$Minimize \sum_{j=1}^{n}[X_j - \hat{X}],$$

$\hat{X}_n = X_n$(추정된 시험능력치는 실제 적용한 시험노력과 동일함)

한편 α, β 값을 알면 모델에 의한 SRGM의 파라미터는 추론과정(stochastic process)을 이용하고 NHPP 모델에 의해 기술된 최대우도추정법(MLE : Maximum Likelihood Estimation)을 통해 추정된다. 추정과정에서 추정된 α, β 값은 고정된다.

만약 모듈에서 소프트웨어 고장이 제거되면 시간 $[0, t_j]$에서 축적된 고장 제거수 y_j로 주어진다. 모듈에 대한 우도함수는 식 (16)과 같이 주어진다.

$$L(a, b, \eta | y, W) = \prod_{j=1}^{n} \frac{[m_r(\hat{X}_i) - m(\hat{X}_{i-1})]^{y_i - y_{i-1}}}{(y_j - y_{j-1})!} \cdot e^{-[m_r(\hat{X}_j) - m(\hat{X}_{j-1})]} \quad (16)$$

추정된 a, b, β 값은 위에 표현한 최대우도추정에 의해 얻어진다.

V. 모델의 검증

소프트웨어 신뢰도성장으로 기술된 제안모델에 대한 검증(verification)을 위해 시험에서 얻어진 고장데이터를 적용한다. 적용된 데이터 군(群)은 2개로 대형시스템 개발과정에서 얻어진 실 데이터(real data) 이다.

-Data set I

본 시스템은 가입자망(Access Network)의 개선을 위한 FTTH(Fiber to the Home) 실현을 위해서 개발된 핵심 장비인 s-OLT(standard optical line terminal) 시스템(순수 소스 프로그램 규모는 98만 라인)으로 멀티미디어 방송서비스 및 TDM 모듈을 활용한 음성 호처리 제공이 가능한 시스템이다. 소프트웨어 모듈은 40개 정도이며, 재사용 비율은 15% 이하로 시스템 개발시 얻어진 데이터로 약 50주 정도 실측 데이터와 추정 결과를 비교한 결과다.

-Data set II

고속 ATM cell을 스위칭해주는 대형 시스템으로 다양한 프로토콜과 가입자 타입별 처리 속도를 달리해주는 음성 및 데이터서비스를 교환해주는 전용 시스템이다. 프로그램 규모는 525만 라인 정도로 시스템 처리 용량은 80G 이다. 소프트웨어 모듈(module or function block) 수는 145개, 재사용 비율은 20% 정도이다.

고장데이터를 적용한 모델 파라미터들에 대한 추정된 결과는 표 1과 같으며, SRGM의 성능은 과거의 경험데이터에 의한 고장발견, 제거과정의 적합도(goodness-of-fit)에 의해 판단되며, 현재 또는 과거의 데이터로부터 고장 발견과정의 투명성에 대한 만족 여부의 예측으로 대변된다.[5,10]

위의 2개 데이터 군중 data set I 을 적용, 예측한 결함수 및 시험능력 투입 결과에 대한 Plot을 그림 1, 2에 보였다.(그림 1, 2에서 점선은 실측치, 실선은 추정치를 의미함)

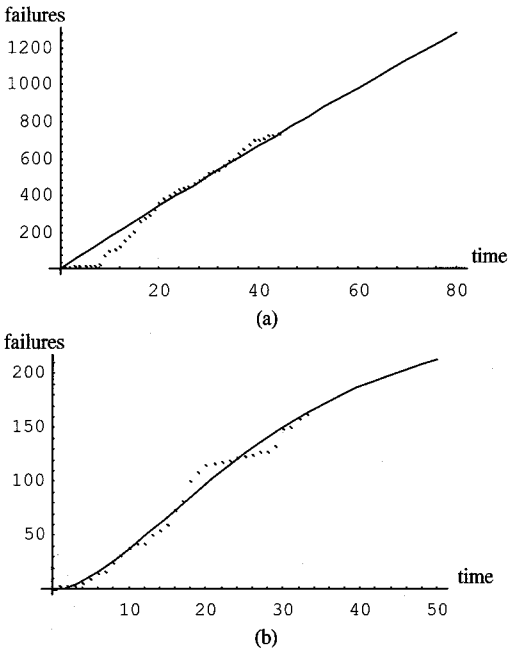


그림 1. s-OLT 시스템의 초기 고장수 및 누적 고장수 분석 결과

그림 1의 (a)는 시험시작 후 45주까지 누적 고장수(cumulative failure)에 대한 지수형 모델에 근거한 실측 및 예측치를 비교한 것이고 그림 1의 (b)는 시스템 시험의 초기 고장 데이터에 대한 S-자형 모델을 적용한 실측치 및 추정치에 대한 분석 결과다.

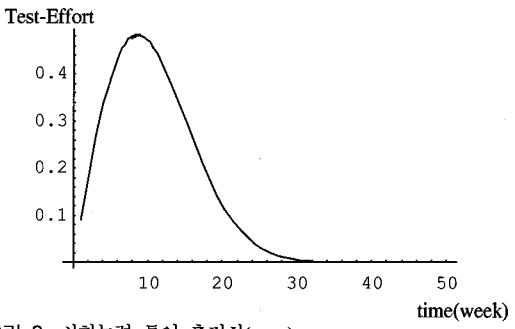


그림 2. 시험능력 투입 추정치(W_R)

그림 2에 비추어 볼 때 시험능력 투입량은 레일리히 곡선을 띠는 것으로 표현되고 있다. 또한 아래 그림 3은 1주일 단위로 시스템 시험에서 검출, 해결

표 1. 프로젝트별 모델 파라미터 추정치

project	α	β	γ	a	b
data set I	2154.81	$2.15*10^{-5}$	$4.0*10^{-2}$	249.238	0.067741
data set II	58.2553	$5.708*10^{-3}$	$6.527*10^{-2}$	940.827	0.347478

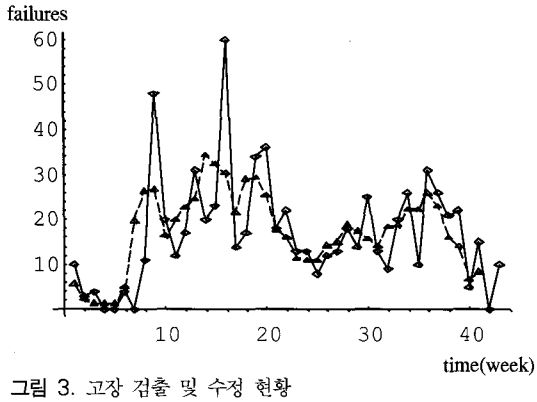


그림 3. 고장 검출 및 수정 현황

되어 가는 고장 데이터의 추이를 나타낸 것이다. 그림에서 점선은 해결 현황을 나타내며, 실선은 고장 검출수이다. 그림 2의 시험능력 투입량과 상호 비교할 때 비교적 시험능력이 많이 투입된 10주 전후에서 많은 고장이 검출되고 있음을 알 수 있다.

-적합도 판정(Goodness of fit)

모델의 적합도 판정 방법에는 여러 가지 방법이 있으며, 대표적으로 수학적인 해석 방법과 데이터 분포를 그래픽으로 판별하여 적합 여부를 판별하는 그래픽 판별법이 있다.

적용될 샘플(표본) 데이터의 수에 따라 각각의 검정방법이 적용되는데, 적용되는 대표적인 방법은 카이제곱 검정법(Chi-Square test : χ^2 -test), K-S 검정법(Kolmogorov-Smirnov test), P-P(probability-probability) Plot 검정법, Q-Q(quantitative-quantitative) Plot 검정, A-D 검정법(Anderson-Darling test) 등이 있다.

아래 기술된 방법은 표본 오차를 줄이기 위한 방법으로 추정치와 관측치 간의 오차를 줄이는 방안으로 사용되는 MSE(Mean Square Error), AIC(Akaiki Information Criterion) 기준통계량 등이 있다.

가. MSE(Mean Square Error)

예측치와 관측치 간의 차이로 모의 실험된 고장 데이터를 이용하며, 아래 식 (17)과 같이 표현된다.

$$MSE=\frac{\sum_{i=1}^{k}(\widehat{m}(t_i)-y_i)^2}{k} \qquad (17)$$

k: 관측수

MSE의 값이 적을수록 fitting 에러가 적은 것으로 간주되며, 모델의 적합도가 높다는 것을 의미한다.

나. AIC(Akaiki Information Criterion) 기준 통계량

AIC 기준 통계량은 신뢰도 모델 선택 툴로써 아래와 같이 정의된다.

$$AIC=-2\log(\text{Max. of likelihood function})+2\text{x N} \qquad (18)$$

N : 모델 파라미터 개수

AIC 기준통계량의 적은 값이 더 적합하며 예측 유효성이 뛰어나며 모델의 자유도(confidence interval)가 높다. 다시 말해서 MSE와 AIC는 모델 성능을 측정하는 metrics 으로 동일한 데이터를 적용했을 때 metrics 값이 적은 모델이 더 적합한 것으로 간주된다. 아래 테이블의 내용은 적합도를 판정하기 위해 사용된 예(표 2 참조)로 data set I에 대한 유의수준 (α=0.05)를 주어 파라미터들을 분석한 결과로써 분산분석표(Analysis of variance table : ANOVA table)에 의한 Parameter 추정치이다.

표 2. 파라미터 추정 결과{ANOVA Table}

	DoF	SoS	MeanSS	FRatio	PValue
Model	01	11.5287	11.5287	0.0727181	0.78874
Error	42	6658.65	158.539		
Total	43	6670.18			

V. 결 론

본 논문은 분산개발 환경에 대한 NHPP 모델에 근거한 새로운 신뢰도 성장모델을 제시하였다. 이 모델은 분산 소프트웨어 개발환경 하에서 일정수의 재사용 모듈과 신규개발 모듈을 적용하는 환경을 고려하여 서로 다른 형태의 에러들을 계측한 모델로써 로지스틱 함수에 의해 기술된 시험능력과 고장시간(고장 발생 간격) 및 고장제거 과정을 모델화하였다.

또 소프트웨어 개발 프로젝트에서 얻어진 실측 데이터를 이용하여 수리적으로 해석(numerical analysis), 그 결과를 보였으며, 시험능력 추정치와 시험을 통해서 발견한 고장수를 비교함으로써 제안된 모델이 프로젝트에 적합하다는 것을 부분적으로 입증하였다. 이와 같은 모델이 이론적인 바탕에 근거를 두고 소프트웨어 신뢰도를 예측, 추정하는 모델들과 차별화를 가질 수 있다. 그밖에 제안된 모델의 결과들은 유사환경에서 다른 모델과의 비교분석과 유효성 검사 등 다각도로 추가 검증이 필요하다.

참 고 문 헌

[1] S. Bittanti, P. Bolzern, E. Pedrotti and R. Scattolini, "A flexible modelling approach for software reliability growth", in Software Reliability Modelling and Identication eds., Springer Verlag, pp. 101-140, Berlin, 1998.

[2] W. D. Brooks and R. W. Motley, Analysis of Discrete Software Reliability Models, technical report (RADC-TR-80-84), Rome Air Development Center, New York, 1980.

[3] A. L. Goel and K. Okumoto, Time dependent error detection rate model for software reliability and other performance measures, IEEE Transactions on Reliability R-28(3), pp. 206-211, 1979.

[4] C. Y. Huang, S. Y. Kuo and J. Y. Chen, "Analysis of a software reliability growth model with logistic testing eort function", 8th Intl. Symp. Software Reliability Engineering, pp. 378-388, 1997.

[5] P. K. Kapur, R. B. Garg and S. Kumar, "Contributions to Hardware and Software Reliability", World Scientic, Singapore, 1999.

[6] P. K. Kapur and R. B. Garg, "A software reliability growth model for an error removal phenomenon", Software Engineering Journal 7, 291-294, 1992.

[7] T. M. Khoshogoftaar and T. G. Woodcock, "Software reliability model selection: A case study", Intl. Symp. Software Reliability Engineering, pp. 183-191, 1991.

[8] M. R. Lyu, Handbook of Software Reliability Engineering, Mcgraw Hill, 1996.

[9] 이재기, 신상권, 홍성백, 윤병남, "시험노력을 고려한 개발단계의 소프트웨어 신뢰성 평가", 대한전자공학회논문지, 제36권 S편 제3호, pp. 18-26., Mar. 1999.

[10] J. D. Musa, A. Iannino and K. Okumoto, "Software Reliability: Measurement Prediction Applications", Mcgraw Hill, New York, 1987.

[11] M. Ohba, "Software reliability analysis models", IBM J. Research and Development 28, pp. 428-443, 1984.

[12] S. Yamada, M. Ohba and S. Osaki, "S-shaped software reliability growth models and their applications", IEEE Transactions on Reliability R-33, pp. 169-175. 1984.

[13] S. Yamada and H. Ohtera, "Software reliability growth model for testing effort control", European J. Operational Research 46, pp. 343-349, 1990.

[14] S. Yamada, Y. Tamura and M. Kimura, "A software reliability growth model for a distributed development environment", Electronics and Communication in Japan, Part 3-83, pp. 1446-1453, 2000.

[15] S. Yamada, H. Ohtera, Software Reliability, SE series, Soft Research Center, Japan, 1990.

[16] S. Yamada, M. Dagahasi, Introduction of Software Management Model, Gong-nip press, Japan, 1993.

[17] 이재기, 이규욱, 김창봉, 남상식, "불완전 디버깅 환경을 고려한 소프트웨어 신뢰도 성장 모델", 한국통신학회논문지, Vol. 29, No. 6A, pp. 589-599, 2004.

이 재 기 (Jae-ki Lee) 정회원

1985년 2월 서울산업대학교 전자공학과 졸업

1989년 5월 청주대학교 전자공학과 석사

2005년 2월 공주대학교 전기 전자정보공학과 박사

1983년 3월~현재 한국전자통신연구원 책임연구원

<관심분야> 소프트웨어 신뢰도, 검증, 테스팅, BcN

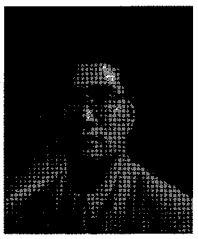
이 재 정 (Jae-jeong Lee) 정회원

2002년 2월 한밭대학교 전자공학과 졸업

1999년 3월~현재 한국전자통신연구원 연구원

<관심분야> BcN, 시스템 시험/검증

남 상 식 (Sang-sik Nam) 종신회원



1981년 2월 단국대학교 전자공학과 졸업

1983년 2월 단국대학교 전자공학과 석사

1996년 3월 단국대학교 전자공학과 박사

1999년 3월~현재 한국전자통신연구원 책임연구원

<관심분야> ATM, Signal Integrity, BcN