論文99-36S-3-3

시험노력을 고려한 개발단계의 소프트웨어 신뢰성 평가

(Software reliability evaluation during development phase considering testing-effort)

李 在 起 *, 辛 相 權 *, 洪 性 伯 *, 尹 炳 楠 *

(J.K Lee, S.K Shin, S.B Hong, and B.N Yoon)

요 약

소프트웨어 개발과정에서는 설계 및 코딩에 의해서 프로그램 내에 많은 결함이 삽입되고 시험을 거치는 동안 결함은 발견되고 수정되어 진다. 이와 같은 반복 작업을 통해서 소프트웨어 내에 잠재하고 있는 에러들을 발견, 조치함으로써 소프트웨어 품질은 향상되고 품질에 대한 신뢰성이 높아진다. 본 논문에서는 위와 같은 소프트웨어 개발 과정을 거치는 동안에 많은 시험이 동반되고 이에 따른 시험의 여러 자원이 투입되는데 착안하여 소프트웨어 신뢰도가 성장되어 가는 과정에 수반된 시험능력을 기반으로 한 시험능력의존형 소프트웨어 신뢰도 성장모델을 살펴보고 이를 적용하여 개발중인 소프트웨어에 대한 시험능력 투입에 따른 신뢰도를 평가하였다. 또 S-Shaped Model에 의한 평가치와 비교하여 시험이 진행됨에 따라 소프트웨어 고장 발견율이 상승되는 것을 보였다.

Abstract

In the software development process such as design and coding phase, the software errors are seeded in programs but they are detected and corrected during test phase. When we find and remove the faults that are latent in the programs, the software quality and reliability are highly improved. For this, we have to invest much test effort and resources such as man power, test time so on. In this paper, we show a new approach to software reliability models using test effort from software reliability growth model. In the point of both reliability growth model and defect characterization. we apply to the test-effort dependant software reliability growth model to estimate software quality. Also we compare of S-shaped model and show that the software failure exposure ratio is growing according to test processing period.

Ⅰ. 서 론

소프트웨어 신뢰성 평가는 소프트웨어의 생산성을 향상시키는데 중요하므로 확률, 통계론에 입각한 정량적인 신뢰도 평가 방법에 대한 모델로 소프트웨어 엔지니어링 측면에서 연구되고 있다.

소프트웨어 신뢰성 평가 문제가 주요 관심사가 되는 시험 단계는 많은 시험 자원이 요구된다. 다시 말해서 시험 케이스, 수행시간, 시험요원(Man-power) 등 시험 능력이라고 불리는 시험 자원들이 대량 투입되게 된다. 이러한 시험 능력의 시간적 변동을 고려하여 달성된 소프트웨어 신뢰성 평가는 시험 공정 관리에 매우 중요하다. 이와 유사한 프로젝트 규모(또는 견적) 모델로는 소프트웨어 상세 개발 작업이나 예산(Cost), 일정(Schedule)등 효과적인 개발 계획과 정확한 개발 규모 및 개발 공수를 참조하는 모델이 있다. 즉, 소프

* 正會員, 韓國電子通信研究院
(Switching Service Department, Software Integration Team, ETRI)

接受日字:1998年9月1日, 수정완료일:1998年10月26日

트웨어 프로젝트 평가(Software Project Estimation) 모델로써 그 대표적인 모델은 Walston and felix model, COCOMO and Putnam model, Gantt-chart를 이용한 PERT/CPM(Program Evaluation and Review Technique/ Critical Path Method) model, WBS(Work Breakdown Structure) 모델 등이 있다[6].

본 논문에서는 이러한 모델 중에서 시험 능력을 고려한 시험능력 의존형 소프트웨어 신뢰도 성장모델(Testing-effort dependant Software Reliability growth model)에 근거하여 소프트웨어 개발 과정 중 시험 공정에 투입되는 시험 양의 변화를 고려한 소프트웨어 순간 고장율을 근거로 한 신뢰성 평가 방법에 대해 논한다. 그 외에 이 모델의 응용으로써 소프트웨어 최적 배포시기(Optimal Release) 문제에 대해 고찰한다.

Ⅱ. 시험능력 의존형 소프트웨어 신뢰도 성장 모델

소프트웨어 개발 과정에서는 많은 에러가 발생되고 수정되어진다. 실제 소프트웨어 개발 프로세스에서의 시험 단계는 대량의 시험 자원을 투입하여 많은 결함의 발견, 수정의 과정을 거치는데 그 이유는 개발 과정의 작업에 의해 삽입되어 지는 소프트웨어 에러의 수정 과정에서 또 다른 에러가 들어가기 때문이다. Yamada et al. 가 제안한 시험능력의존형 신뢰도 성장모델[1]에서는 에러의 발견, 수정과정에서 파생되는 새로운 에러는 무시하는 것으로 가정하고 있으나 실제 시스템 개발 과정에서는 검출된 총 에러 중 약 5 % 정도가 에러의 수정과정에서 파생된 에러가 발생하는 것을 알 수 있다[7].

1. 신뢰도 성장모델의 일반 개념

시험 단계에서 시험 시각 t 마다 검출된 에러를 개수(個數)해 나가는 과정을 비동차 포아송 과정(NHPP : Non-Homogeneous Possion Process)으로 나타내면 아래와 같은 식 (1), (2)로 나타낼 수 있다.

$$P[N(t)=n] = \{H(t)\}^n \frac{e^{[-H(t)]}}{n!} \quad (1)$$

$$H(t)=\int_0^t h(x)dx (n=0,1,2,.....) \quad (2)$$

위의 함수 H(t)는 시간구간(0, t)에서 발견된 총 기대 에러수를 나타내는 평균치 함수이고 $h(x)$는 단위시간당 발견되는 에러수에 대한 강도함수이다.

검출가능 결함수가 유한일 경우에 단위 시간당 발견될 에러수는 그 시점에서의 소프트웨어내에 잠재하고 있는 에러수에 비례한다는 가정에 의해 아래와 같은 미분 방정식으로 나타낼 수 있다.

$$\frac{dH(t)}{dt} = b(t)[a-H(t)], (b(t)>0, t\geq 0) \quad (3)$$

$b(t)$: 시험시각 t에서의 1개당 결함 발견율(소프트웨어 고장율)

a : 시험 시작전 소프트웨어내에 잠재하고 있는 총 기대 에러수

여기서 $b(t)$는 소프트웨어 내에 잔존하고 있는 결함에 대한 발견 난이도를 의미하며, 이것은 3가지 형태를 띠게 된다. 즉, 시간 t 에 대하여 감소(DFDR : decreasing fault detection rate), 증가(IFDR : increasing fault detection rate), 일정(CFDR : constant fault detection rate)한 형태의 에러 발견율을 나타낸다. b(t)가 시간 t에 관해 일정하다면 위의 식(3)을 H(t)에 관해 초기조건 [H(t = 0) = 0] 하에서 미분방정식을 풀면

$$H(t)=a(1-e[-\int_0^t b(x)dx]) \quad (4)$$

가 되며, 이때 b(t) = $b_m(t)$ = b, (b > 0)로 가정하는 경우 Goel-Okumoto가 제안한 지수형 신뢰도 성장모델의 평균치 함수는

$$H(t)=m(t)=a(1-e^{-bt}) \quad (5)$$

로 된다. Yamada et al.이 제안한 지연 S-Shaped 모델에서는 고장 발견과정과 고장 인지과정을 나누어 생각하기 때문에 평균치 함수식 및 1개당 에러 발견율은

$$H(t)=M(t)=a[1-(1+bt)e^{-bt}], (a>0, b>0), \quad (6)$$

$$b(t)=b_M(t)=\frac{b^2 t}{1+bt}, b_M(0)=0, b_M(\infty)=b \quad (7)$$

로 가정할 수 있다.

2. 시험능력의존형 신뢰도 성장 모델

시험능력의존형 모델에서는 시험시각 t에 대해 시험

능력량을 투입한 시험능력함수 $\omega(t)$를 도입하여 정리하면

$$b(t) = b_T(t) = \gamma\omega(t), (0<\gamma<1),$$

γ : 순간 고장율

로 가정하여 식(4)로부터

$$H(t) = T(t) = a(1-e^{-\gamma \cdot W(t)}),$$

$$W(t) = \int_0^t \omega(x)\,dx$$

를 얻는다. 위의 소프트웨어 신뢰도 성장 모델에서 시험시각 t 에서의 소프트웨어 안에 잠재하고 있는 총 기대 에러수 n(t)는

$$n(t) = a - H(t) \tag{8}$$

. a : 시험 전 소프트웨어 내에 잠재하고 있는 총 에러

로 나타내진다. 또 시각 t 까지 시험이 진행될 때 시간구간 (t, t+x)에 있어서 에러가 발견되지 않을 확률 즉, 소프트웨어 신뢰도 R(x|t)는

$$R(x|t) = e[-H(t+x) - H(t)\}(t>0, x>0) \tag{9}$$

로 주어진다. 실제 개발 단계의 시험 공정에서는 소프트웨어 신뢰성을 평가할 때 구체적인 신뢰도 성장모델을 정할 필요가 있는데 본 논문에서는 시험능력을 고려한 시험능력의존형 성장모델을 생각하기로 한다. 이 모델에서는 시험 공정에 투입되는 시험 자원 즉, 시험능력의 투입량은 개발 시간에 대해 레일리이(Rayleigh) 곡선을 따르는 것으로 알려져 있다[2]. 이 모델은 시험자원의 투입량에 대한 경향이나 변화를 기술하는데 융통성과 적용성이 뛰어나고 레일리이 곡선을 포괄하는 와이블(weibull) 곡선을 이용한다. 즉, 시험시각 t 에서 투입되는 시험 능력량을 $\omega(t)$라고 하면

$$\omega(t) = \alpha\beta m\, t^{m-1}\, e^{-\beta t^m},$$

$$(\alpha>0, \beta>0, m>0) \tag{10}$$

로 된다. 이것은 다시

$W(t) = \alpha(1-e^{-\beta t^m})$ 으로 정리된다. 여기서 m = 1(지수곡선) 및 m = 2(weibull 곡선) 에 대해 정리하면

$\omega(t) = \alpha\beta\, e^{-\beta t}$, $W(t) = \alpha(1- e^{-\beta t})$와

$\omega(t) = 2\alpha\beta t\, e^{-\beta t^2}$, $W(t) = \alpha(1- e^{-\beta t^2})$

α : 최종적으로 투입되는 총 시험 능력량, β : 시험능력 투입율, m : 형상 파라미터

를 얻는다. 이것을 식(1)의 비동차 포아송 과정(Non-Homogeneous Poisson Process) Model의 평균치 함수식에 대입하면 식(4)가 된다. 위에서 언급한 것과 같이 소프트웨어 내에 잠재하고 있는 에러를 발견, 수정하는데 시험 자원의 다수가 시험용 입력 데이타 즉, 테스트케이스가 실행된다. 이때 테스트케이스로부터 영향을 받은 소프트웨어 모듈 내에 여러 경로가 존재하게 되는데 시험 진행에 따른 입.출력 관계가 명확한 경우 실행된 테스트케이스의 증가에 대한 고려가 필요하게 된다. 다시 말해서 시험공간 점유율과 시험이 실시되었을 때의 발견 가능한 결함수의 관계가 정식으로 성립할 수 있다. 이와 같은 시험능력을 고려한 모델을 시험공간 의존형 신뢰도 모델(testing-domain dependent software reliability model)[3]이라 부르며, Ohtera et al. 과 Yamada et al.가 제안하였다. 이 모델은 대형 교환 소프트웨어 신뢰도 평가 방법에 적용하기에는 너무 복잡하여 적용하기가 곤란하나, 실제 소프트웨어 개발 시 기능시험에 일부 적용이 가능하다.

본 논문에서는 시험시 고장을 검출하는데 투입한 시험 능력을 고려한 시험능력의존형 신뢰도 성장 모델에 대해 생각하기로 한다.

Ⅲ. 시험능력의존형 모델에 의한 소프트웨어 신뢰성 평가 및 최적 배포 문제

시험단계를 거쳐 현장에 배포되는 소프트웨어의 신뢰성은 실시된 시험의 질과 양에 관계된다. 즉 소프트웨어 고장에 대한 순간 고장율($\gamma(s)$)는 시험 단계에서 미 발견된 잔존 에러수와 운용시간에 비례하게 되는데 식(4) 및 (6)에 의해

$$\gamma(s) \equiv \frac{d}{ds} F(s)/[1-F(s)] = pn(t)\, s^{k-1} =,$$

$$ap\, e^{-b\alpha(1-e^{-bt^m})}]\, s^{k-1} \tag{11}$$

F(s) : 배포직후 고장발생에 대한 분포함수,

p, k : 정수(형상) 파라메타

표현되며, 이것은 분포함수 F(s)가 형상 파라메타인 k의 와이블(weibull) 분포인 것을 의미한다. 이때 신뢰도 함수 R(s)와 평균고장시간(MTTF : Mean-time to software failure)을 구하면 식(12), (13)와 같이 된다.

$$R(s) = e^{-ap}[(e^{-ba(1-e^{-\beta t^m})})\, s^k/k], \tag{12}$$

$$MTTF = \Gamma(1/k)/\sqrt[k]{ap\, e^{-ba(1-e^{-\beta t^m})}\, k^{k-1}} \tag{13}$$

소프트웨어 개발에 있어서 가장 중요한 문제의 하나로써 시험을 중단하고 운용국에 배포하는 시기를 정하는 것이다. 특히 최적의 개발비용과 최대의 소프트웨어 신뢰성을 갖는 소프트웨어 개발은 실제 개발에 있어서 가장 중요한 이슈(issue)이다. 이와 같은 최적 배포시기 결정 문제는 적당한 신뢰성 평가 기준을 도입하여 시험기간과 시험에 의해 확인되는 소프트웨어 신뢰성과의 관계가 반영되도록 정식화되어 이를 최적 배포 문제라고 부른다[4, 5]. 즉, 원하는 신뢰도 목표치를 정하고 이 목표치를 달성했을 때 시험을 종료하여 소프트웨어를 사용자에게 배포하는 것이다.

운용단계에서의 신뢰도 및 MTTF의 목표치를 각각 R_0, T_0라고 가정하면 이 목표치를 달성했을 때 운용시간 $S=S_0$로 정하여 시험능력의존형 모델에 적용하고 이를 근거로 총 시험시간 즉, 배포시기 t^* 를 구하면 위의 식(12), 식(13)으로부터 각각

$$t^* = [-\ln\{\frac{1}{ab}\ln(\frac{-k\ln R_0}{pa\, S_0^{\,k}})+1\}/\beta]^{\frac{1}{m}},$$

$$t^* = -\ln\{1-\ln(\frac{k^{k-1}pa\, T_0^{\,k}}{\{\Gamma(1/k)\}^k})/ab\}/\beta$$

구할 수 있다. 아래는 실제 데이터를 적용하여 평가한 데이터로 그 결과는 다음과 같다.

시험 능력 함수 $\omega(t)$ 에 대해 파라미터 α, β를 단순회귀분석으로 정하면

$$\ln\omega(t) = \ln\alpha + \ln\beta + \ln m + (m-1)\ln t - \beta t^m$$

로 정리된다. 여기서 시험시각 t_k 마다 관측된 시험능력량 w_k 의 시험함수의 추정치 $w(t_k), (k=1,2,3,\ldots\ldots,n)$에 대해 편차 자승화에 의한 최소 자승법으로 파라미터들을 추정하면

$$S(\alpha,\beta,m) = \sum_{k=1}^{n} |\ln w_k - \ln w(t_k)|^2 =$$

$$\sum_{k=1}^{n} |\ln w_k - \ln\alpha - \ln\beta - \ln m - (m-1)\ln t_k + \beta t_k^m|^2$$

으로 정리하여 이를 편미분하면 α, β 의 값을 구할 수 있다. 이때 지수곡선 및 레일리이 곡선을 포함하는 파라메타 α, β 는

○ 지수 곡선을 포함하는 경우

$$\alpha = e[\frac{1}{n}\sum_{k=1}^{n}\ln w_R - \ln\beta + \frac{\beta}{n}\sum_{k=1}^{n} t_k],$$

$$\beta = (\sum_{k=1}^{n}\ln w_k)(\sum_{k=1}^{n} t_k) - n(\sum_{k=1}^{n} t_k.\ln w_k)/$$

$$n\sum_{k=1}^{n} t_k^2 - (\sum_{k=1}^{n} t_k)^2$$

○ Rayleigh 곡선을 포함하는 경우

$$\alpha = e[\frac{1}{n}\sum_{k=1}^{n}\ln(\frac{w_k}{t_k}) - \ln\beta + \frac{\beta}{2n}\sum_{k=1}^{n} t_k^{\,2}], \tag{14}$$

$$\beta = [2(\ln\sum_{k=1}^{n}(\frac{w_k}{t_k}))(\sum_{k=1}^{n} t_k^2) - n(\frac{w_k}{t_k})(\sum_{k=1}^{n} t_k^2\ln(\frac{w_k}{t_k}))]/$$

$$n\sum_{k=1}^{n} t_k^{\,4} - (\sum_{k=1}^{n} t_k^{\,2})^2 \tag{15}$$

으로 추정된다.

Ⅳ. 실제 데이터 적용 결과

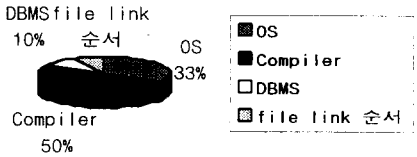
적용된 데이터는 ATM 교환 시스템의 개발 과정 중 시험단계에 투입되는 시험요원(test man power), 시험 케이스(test case & test scenario), 시험 횟수 및 시험기간 등 시험능력을 집중시켜 얻어진 소프트웨어 배포 사례이다. 소형 ATM 교환 소프트웨어(SV3.2)와 중/소규모의 상용 교환 시스템의 소프트웨어(SV3.4)에 대한 시험능력의존형 소프트웨어 신뢰도 성장모델의 적용 사례다. 적용된 사례는 소프트웨어 배포시기 및 신뢰도 목표치를 설정한 상태에서 시험자원을 적극 활용하여 얻어진 결과를 반영하였다. 본 논문에 적용한 시험능력 투입량 W_k 는 편의상 단위시간당 투입된 시험 능력을 위주로 가정한 데이터임을 밝혀둔다.

1. 대상 소프트웨어 규모 및 시스템 현황

연구 대상이 된 ATM 교환 소프트웨어의 규모는 SV3.2(Switched Virtual Connection Software

version 3.2) 버전이 약 70개의 기능 블록과 275,000 소스코드 라인으로 구성되어 있으며, SV3.4 버전인 경우는 전체 93개 기능 블록에 375,000 소스코드 라인의 규모를 이루고 있다. 표 1과 같이 소프트웨어 버전 SV3.2는 시험시작 후 10개월 경과 시 많은 에러가 검출되었는데, 이 원인은 교환시스템의 운용프로파일의 변경에 따른 테스트케이스의 다변화에 대처하지 못한 원인으로 상용화를 위한 현장 적용시험 기간이다. 또 SV3.4 버전의 시험시작 후 3개월 후는 신규 기능이 새로운 개발 환경에 통합된 시기로 이때 많은 에러가 발생, 수정되었다. 이렇게 많은 고장이 검출된 원인은 개발 환경의 차이에서 오는 개발 공수의 변화가 많았기 때문이며, 프로그램 환경 차이에 대한 이식 경험의 부족과 프로그램 이식 전 시스템의 응용프로그램에 대한 이해 부족이었다. 소프트웨어 이식 저해 요인에 대한 통계 데이터는 아래와 같다.

TDX-ATM 시스템의 소프트웨어
이식 저해 요인 분석



특히 위의 2개 소프트웨어 버전의 차이점을 호 서비스 관점에서 볼 때 교환 시스템의 핵심 기능인 호 처리 소프트웨어 중 다중 연결관리(PtMP : Point to Multi-Point connection management) 기능의 차이와 시스템 하드웨어 및 개발환경(OS) 변화였다. 이로 인해 이 기간 동안에 시험으로 발견된 고장 검출 율은 다른 버전에 비해 다소 높았다.

시험 능력 의존형 소프트웨어 신뢰도 평가 모델의 대상으로 SV3.2 버전과 SV3.4 버전을 택한 이유는 시스템 개발확인시험과 상용화의 척도를 가름하는 상용시제품시험이라는 중요한 이슈(issue)가 걸린 패키지로서 해당 소프트웨어를 개발하는 과정에 시험 능력을 집중시킬 수 있는 대표적인 케이스였기 때문이다.

2. 적용 데이터 및 신뢰도 평가

표 1의 시험능력 데이터(t_k, w_k) (k = 1, 2, 3, ...18)를 이용하여 레일리이 곡선을 포함하는 신뢰도 성장모델의 파라미터 α, β를 III절에 언급된 식 (14), (15)에 따라 추정하면

α = 71.8899, β=5.753 x 10^{-3}로 또 SV3.2 및 SV3.4 전체에 대해 α, β를 추정하면

$$\alpha = 176.327,\ \beta = 1.523 \times 10^{-3}$$

으로 추정된다.

표 1. 시험능력의존형 시험 데이터
Table 1. Test data for testing-effort.

시험시각	시험능력 투입량(w_k)		총 발견 에러수(v_k)	
t_k(월)	SV3.2	SV3.4	SV3.2	SV3.4
1	2.5	2.67	15	15
2	4.625	2.0	37	12
3	1.17	7.0	7	42
4	2.67	14.83	16	91
5	1.0	10.83	6	69
6	0.3	11.5	2	70
7	2.5	5.67	15	34
8	8.83	6.0	53	37
9	9.83	5.5	59	34
10	17.67	7.17	107	44
11	17.67	-	107	-
12	6.17		37	
13	1.67		9	
14	1.67		10	
15	5.0		29	
16	5.17		32	
17	5.17		33	
18	0.67		4	
19	-		-	

이때 시험능력 $\omega_R(t)$는

$$\omega_R(t) = 2\alpha\beta t[\ e^{-\beta t^2}] = 71.8899 \times 5.753 \times 10\text{-}3\ e^{-5.753x\,10^{-3}\,t^2}$$

로 되고 파라미터 α, β 를 시험시각 t에 대해 추정하는 경우 총 시험능력함수 W(t)와 실제 투입된 총 시험능력이 동일하다고 가정하면 이때 추정된 결과는

$$\alpha = 58.2553,\ \beta = 5.70893 \times 10^{-3}$$

이었다.

아래 그림 1은 실제 시험 능력 투입량에 따른 누적치를 보인 것이다. 이것을 토대로 시험능력 함수를 구

하면 추정된 시험 능력함수 $\omega_R(t)^*$는

$$\omega_R(t)^* = 58.2553(1- e^{[-0.00570893.t^2]}),$$

또, 추정된 평균치 함수는

$m_w(t) = a(1- e^{\gamma\omega_R(t)})$ 와 같이 된다.

여기서 γ 는 시험능력 투입량 당 고장 발견율로 S-자형 신뢰도 성장모델에서는 γ = d/b 로 표현된다,

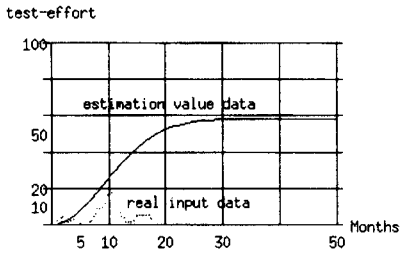


그림 1. 실측데이터와 레일리이 곡선으로 표시된 시험 능력함수 누적 추정치

Fig. 1. Estimation of real data and the predictive of graph testing-effort function to Rayleigh curve.

즉, $\gamma = \frac{d}{b}(0 \le \gamma \ge 1)$,

d : 소프트웨어 초기 고장 발견율,

b : 정상 고장 발견율

로 표현된다. 여기서 $\gamma > 1$ 인 경우는 시간에 따라 시험의 효과가 높은 것을 의미하며, $\gamma < 0$ 인 경우는 에러의 발견율이 시험 초기보다 떨어져 시험이 진행될수록 점점 소프트웨어 내에 잠재하고 있는 에러수가 줄어들어 에러의 발견이 용이하지 않은 현상을 의미한다. 이 모델에서 γ =1 이 되면 일반적인 지수형 성장모델과 같은 의미를 갖는다. 이밖에 소프트웨어의 기대 잔존 에러수 $n_W(t)$ 및 소프트웨어 신뢰도 $R_x(x|t)$는

$$n_W(t) = n_w(t) = m_w(\infty) - m_w(t) =$$

$$a(e^{-\gamma.\omega_R(t)} - e^{-\gamma.\omega_R(\infty)}), \quad (16)$$

$m_w(t)$: 시험시각 t에서의 평균치 함수

$m_{w(\infty)} = a(1 - e^{-\gamma^* a}) \neq a,$

a : 시험 시작 전 총 고장수

$$R(x|t) = e^{-a}(e^{-\gamma^* W(t)} - e^{-\gamma^* W(t+x)}) \quad (17)$$

로 표현된다. 이것은 시험마다 발견되는 고장이 존재한다는 것을 의미하며, 이 경우 이용 가능한 총 시험 능력량이 유한하다고 가정하면 $\omega(\infty) = a$ 로 가정할 수 있다.

γ는 앞의 식 (11)로부터 얻을 수 있는데 형상 파라미터(p, k)에 대한 값들은 실제 시험에서 검출된 데이터만 가지고 추정할 수 없기 때문에 이와 유사한 동종의 소프트웨어에서 경험한 경험 데이터 및 실제 field에서 운용한 운용 데이터로 해석, 추정한다[7],[8].

과거 경험 데이터를 토대로 초기 고장발견율(0.0057145) 및 정상에러 발견율(0.087547)을 적용하여 에러 발견율을 계산하면

γ = 0.06527이 된다. 즉, 위에서 밝힌 것과 같이 개발확인시험과 상용시제품시험을 거치는 과정에서 단기간에 사용자의 요구사항을 만족하기 위해 시험능력을 집중시킨 것이 해당 기간의 고장 발견에 대한 시험 효과가 높은 것으로 평가된다. 위와 같이 계산된 데이터를 이용하여 시험 능력 의존형 모델에 근거하여 현장 배포 시기인 시험시작 18개월 후(t=18) 소프트웨어내의 잔존 에러수를 평가해 보면 아래와 같이 된다.

$$n_w(t=18) = a(e^{-\gamma^*\omega_{R(t)}} - e^{-\gamma^*\omega_{R(\infty)}}),$$

a = 940.827, γ =6.527x10-2

을 적용하면 위의 식(16)에 의해 소프트웨어 내 잔존 에러수는 SV3.2 버전이 40개, SV3.4 버전이 6개로 추정되었다. 이와 같은 방법으로 소프트웨어내의 잔존 에러를 모두 제거하려면 시험을 앞으로 36개월 내지 20개월을 더 시험해야 된다는 것이다. 그러나 일부 데이터는 운용상에 크게 지장이 없는 보이지 않는 고장(latent fault)으로 확인되었고, 일부 나머지 고장들은 추후 시험 능력을 향상시켜 수정되었다.

3. S-Shaped 모델 파라미터를 이용한 고장밀도

고장 데이터 해석에 있어서 관측된 고장에 대해 고장 발생 과정과 최종 고장으로 인정하는 에러인지 과정으로 구분하여 해석, 적용하는 신뢰도 모델을 S-Shaped 모델이라 부른다[9]. 이 모델의 누적 고장수는 식 (18)과 같이 표현 된다.

$$\mu(t) = a(1-(1+bt)e^{-bt}), (a, b>0) \quad (18)$$

아래 그림 2는 시험에서 검출된 고장 데이터를 가지고 S. Yamada.가 제안한 S-Shaped 신뢰도 성장 모델

에 근거하여 추정한 버전별 고장 밀도 함수 데이터이다. 이 모델에 근거하면 시험 초기에 많은 에러가 발생되며 점차 감소되는 전형적인 통신 시스템의 고장 추세를 보인다.

즉, 시험에서 검출된 고장수를 가지고 고장 검출과정과 인지과정으로 구분하여 소프트웨어 신뢰도를 평가하는 S-shaped Model로 추정한 버전별 고장 밀도 함수로써 시험 능력 투입량의 변화에 따른 신뢰도 성장 추이와 비교해 볼 때 시험이 진행됨에 따라 소프트웨어가 점차 안정화되어 가고 있음을 알 수 있다.

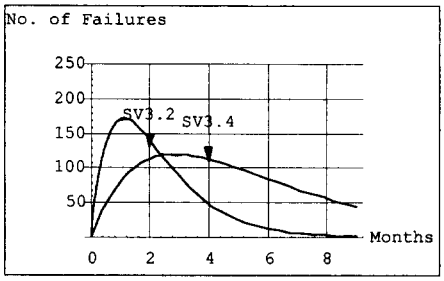


그림 2. 버전별 고장 밀도 함수
Fig. 2. Fault density Function of each version.

시험에서 검출된 고장 데이터를 이용하여 S-Shaped 모델로 추정된 결과는 표 2와 같으며, 소프트웨어 버전 SV3.2 보다 SV3.4 버전의 고장 발생율(b)이 높게 나타났다. 이와같은 현상은 시험이 진행될수록 특히 현장 배포시기가 가까울수록 고장 발견율이 높아지는데 반해 고장 발견은 어려워진다. 그 이유는 오랜 경험을 통한 시험능력이 최고조에 이르기 때문이다. 위와 같은 사실은 표 2의 데이터로 알 수 있다.

표 2. S-Shaped Model에 의한 추정치
Table 2. Estimated values a and b of S-Shaped Model.

version 명	a(추정치)	b(추정치)
SV3.2	940.827	0.347478
SV3.4	527.496	0.884934

a : 시험 전 총 고장수, b : 고장 발견율

소프트웨어 배포시기(표 1참조)인 시험 시작 후 t=18(개월)일 때 SV3.2와 SV3.4 버전에 대한 소프트웨어 신뢰도(R_1, R_2)를 구해보면

$$R_1(x|t=18)=e^{-a}(e^{-\gamma\cdot\omega_R(t)}-e^{-\gamma\cdot\omega_R(18+x)})=$$

$$e^{-940.827}(e^{-0.06527\ *W_R\ (18)}-e^{0.06527\ *W_R\ (18+x)}),$$

$$R_2(x|t)=e^{-a}(e^{-\gamma\cdot W_R\ (t)}-e^{-\gamma\cdot W_R\ (t+x)})=$$

$$e^{-527.496}(e^{-0.06527\,W_R\ (18)}-e^{-0.06527\,W_R\ (18+x)})$$

로 나타낼 수 있으며, 이때 운용시간(x)에 대한 버전별 신뢰도 추정치는 표 3과 같이 추정된다. 표 3의 결과를 놓고 볼 때 소프트웨어 버전 SV3.2 보다는 SV3.4 버전이 다소 안정하게 동작하고 있음을 보여준다.

표 3. 버전별 신뢰도 추정치
Table 3. Estimated of reliability per Version.

Version	x (운용시간)	W_R (t=18월)	R(x|t)
SV3.2	0.01	49.1114	0.95425
	0.05	49.1864	0.79209
	CROS 환경, PVC-PtMP 가능		
SV3.4	0.01	49.1114	0.97408
	0.05	49.1864	0.87749
	SROS 환경 및 H/W 재설계		

4. 소프트웨어 최적 배포 시기 결정

소프트웨어 최적 배포시기 결정 문제는 III장에서 밝힌 것과 같이 개발비용과 신뢰도와의 관계를 정하는 문제로써 원하는 신뢰도 목표치를 정하고 목표치에 도달하기까지의 최적 비용과의 관계를 정하는 문제이다. 즉, 신뢰할만한 소프트웨어를 현장에 배포하기 위해 시험을 언제 중단할 것인가를 결정하는 문제는 비용(Cost)을 고려한 소프트웨어 신뢰도 평가 방법의 최대 관심사이다.

소프트웨어 최적 배포 문제는 크게 신뢰성 평가 척도에 근거한 최적 배포시기 결정 방법과 비용(Cost) 평가 기준에 근거한 최적 배포 문제, 또 신뢰도 및 비용 평가 기준을 동시에 고려한 배포시기 문제 등 3가지로 나눌 수 있는데 본 논문에서는 신뢰도 목표치에 도달하기 위한 총 시험 시간, 다시 말해서 신뢰성 평가 척도에 근거한 최적 배포시기 결정 방법을 택해 평가해 보았다.

앞의 식 (17)에 의거 에러 발견율(r)을 적용하여 추정한 버전별 소프트웨어 신뢰도 추정치와 시험능력함수 예측치는 그림 3 및 표 3과 같다. 그림 3의 결과를 놓고 볼 때 소프트웨어 배포를 위한 신뢰도 목표치

(95% 수준)의 도달시기는 시험 시작 후 약 18개월 전후인 것으로 예측되었다. 운용시간 x=0.01 일때 각 버전별 소프트웨어 신뢰도 성장 추이 곡선은 그림 3과 같고 이에 대한 세부 데이터 추정치는 위의 표 3에 보였다.

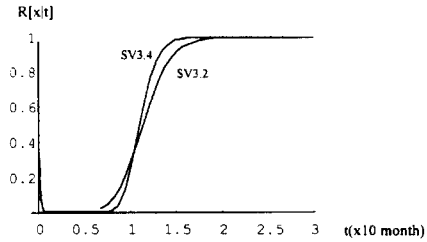


그림 3. 버전별 소프트웨어 신뢰도
Fig. 3. Software Reliability per version.

V. 결 론

본 논문은 ATM(Asynchronous Transfer Mode) 교환 시스템을 개발하는 과정중 시스템 시험 단계에서 시험능력의존형 소프트웨어 신뢰도 성장모델을 적용하여 시험마다 검출된 고장 데이터를 근거로 소프트웨어 신뢰도를 평가하였다. 또 S-Shaped Model을 이용한 평가 데이터와 비교하여 시험능력 향상에 따른 고장 발견율이 운용단계 즉, 소프트웨어 배포시기에 가까울수록 점점 상승된다는 것을 확인하였다. 다시 말해서 시험능력이 소프트웨어 배포시기 단계에 최고로 향상됨을 알 수 있다.

고장을 검출하는데 소요된 시험능력은 기존의 소프트웨어 신뢰도 성장모델에서는 잘 적용되지 못했던 부분이며 시험능력의존형 소프트웨어 신뢰도 성장모델이라는 새로운 개념의 모델을 실제 프로젝트에 적용해 봄으로 인해 소프트웨어 신뢰성 평가 분야에서 잘 다루어지지 않았던 시험능력을 고려해 보았다. 특히 시스템 개발 프로젝트 추진 시 많은 양의 시험이 수행되어 얻어진 결과물로 하나의 시스템이 완성되어 가는 과정에서 투입된 시험자원 즉, 시험능력은 소프트웨어 신뢰도 성장 평가 분야에 매우 중요한 데이터이다.

향후 연구 과제로는 다른 신뢰도 성장모델과의 비교, 평가 및 실제 운용되고 있는 유사한 교환 소프트웨어와의 신뢰성 평가 비교, 현장 운용데이터에 의한 순간 고장율을 적용한 시험능력의존형 성장모델에의 실제 적용 방안 연구, 그 외의 지속적인 경험 축적과 신뢰도 추적 등이 필요하다.

참 고 문 헌

[1] S. Yamada, H. Ohtera and H. Narihisa, "A testing-effort dependent reliability model for computer programs", Trans. IECE Japan, E69, 11, pp. 1217-1224, Nov. 1986.

[2] V. R. Basili and M. V. Zelkowitz, "Analyzing medium-scale software development", Proc. 3rd Intl. Conf. Software Engineering, pp. 116-123, 1978.

[3] S. Yamada, Dagahashi Oneo, "Introduction of Software Management : 소프트웨어 품질의 가시화와 평가법", pp.167-196, 공립출판사, Japan, Jan. 1993

[4] H. Ohtera, S. Yamada, and H. Narihisa, "An optimal release problem based on a test-effort dependent software reliability model", Trans. IEICE of Japan, vol. E71, no. 11, pp. 1140-1145, Nov. 1988.

[5] S. Yamada, H. Narihisa and S. Osaki, "Optimum release policies for a software system with scheduled software delivery time", Intl. J. System Science, vol. 15, no. 8, pp. 905-914, Aug. 1984.

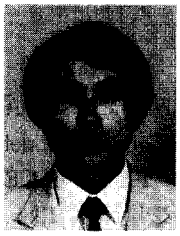
[6] S. Yamada and H. Ohtera, "소프트웨어 신뢰성 : 이론과 실천적 응용", pp. 31-46, Software Research Center SE series, Japan, Feb. 1990

[7] 이재기외, "고장데이터 분석을 통한 교환 소프트웨어 특성 연구", 한국통신학회논문지 98-8 vol. 23, no. 8, 1998. 8

[8] 유재년, 이재기, "기능블럭으로 구성된 대형 교환 소프트웨어 신뢰도 성장", 대한전자공학회논문지 제35권 S편 1호, 1998. 1

[9] S. Yamada, "소프트웨어 신뢰성 평가 기술", pp. 107-112, HBJ Integrated Library no. 42, Japan, April. 1989

저 자 소 개



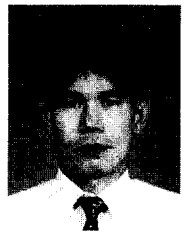
李 在 起(正會員)

第 35卷 S編 第 1號 參照

주관심분야는 Software Quality & Test, Software Reliability

洪 性 伯(正會員)

1990년 2월 연세대학교 대학원 전자공학과 졸업(공학석사). 1998년 10월 현재 한국전자통신연구원 교환. 전송연구소, 교환서비스연구부, S/W종합검증팀, S/W종합검증팀장(책임연구원) 재직중. 주관심분야는 Software Integration & Test



辛 相 權(正會員)

1985년 2월 중경공업전문대학 전자과 졸업. 1988년 2월 ~ 98년 10월 현재 한국전자통신연구원 교환. 전송연구소, 교환서비스연구부, S/W종합검증팀 재직중. 주관심분야는 Software Integration & Test

尹 炳 楠(正會員)

1996년 ~ 98년 현재 한국통신학회 통신소프트웨어 학술위원장, Asia ISDN Council(AIC) WG-1 의장, 1998년 10월 현재 한국전자통신연구원 교환.전송연구소, 교환서비스연구부, 교환서비스연구부장(책임연구원, 전산학 박사) 재직중. 주관심분야는 소프트웨어 개발방법론, ATM, 실시간 분산 스위칭, 차세대 인터넷 기술, 그룹웨어 서비스 기술