소프트웨어 내에 잠입한 에러에 의한 불완전 디버깅을 고려한 소프트웨어 신뢰도성장모델

Software Reliability Growth Models for an Imperfect Debugging with Induced Software Faults

이재기(J.K. Lee) 네트워크시험팀 책임연구원
이경호(K.H. Lee) 네트워크시험팀 책임연구원, 팀장
박권철(K.C. Park) 네트워크전략연구부 책임연구원, 부장

소프트웨어의 신뢰성을 정량적으로 평가하는 데 있어서 소프트웨어 개발 프로세스의 시험단계나 사용자의 운용단계에 처한 동적 환경상태에서 소프트웨어 고장발생기능 현상을 기술한 소프트웨어 신뢰도성장모델을 많이 제안하고 있다. 대다수의 모델이 발생된 소프트웨어 고장의 발생원인에 대한 완전한 수정을 요구하는 완전 디버깅 환경을 가정하고 있다. 그러나 실제 개발자가 디버깅 작업을 수행할 때 완전한 수정이 불가능하기 때문이다. 다시 말해서 여러 소프트웨어 개발자가 경험한 이러한 디버깅 작업을 행하는 경우에는 결함을 제거하는 데 한계가 있기 때문에 수정 작업시 새로운 결함이 삽입되는 경우가 많다. 즉, 결함 수정은 불완전 환경에 처한다. 본 논문에서는 결함 수정시 신규 결함의 삽입 가능성을 고려하고 불완전 디버깅 환경에 대한 소프트웨어 신뢰도 성장모델을 제안한다. 소프트웨어 동작 환경 하에서 발생된 소프트웨어 고장과 시험 전 소프트웨어 내의 고유 결함에 의한 고장과 동작중에 랜덤하게 삽입된 결함에 의해 발생되는 고장 등 2종류의 결함을 고려하여 비동차 포아송과정(NHPP)에 의한 소프트웨어 고장발생 현상을 기술한다. 또한 소프트웨어 신뢰성 평가에 유용한 정량적인 척도를 도출하고 실측 데이터를 이용하여 적용한 결과를 제시하고 기존의 모델과의 적합성을 비교, 분석한다.

I. 서 론

소프트웨어 개발관리에 필요한 관리지표는 품질, 비용 및 최적 납기(optimal release)이다. 즉, 개발자원의 적정성을 표현하면 품질관리, 비용관리 및 공정관리를 효율적으로 추진하는 것이 중요하다. 그 중에서도 품질관리는 효율적으로 실시할 때 생산성을 확보할 수 있으며, 중요한 관리활동으로서 다양한 소프트웨어 산업에 이용된다. 또한 소프트웨어 제품의 품질특성의 짐작으로 소프트웨어 신뢰성이 매우 중요하고 이러한 기술을 공유시 이 기술을 적용하여 효율적인 개발을 추진하는 데 있어서 관리기술의 필요는 필수불가결하다.

본 논문에서는 소프트웨어 신뢰성 관리기술 분야의 소프트웨어 신뢰성 측정 및 평가법에 대해 논한다. 여기서 소프트웨어 개발 공정의 하나인 시험공정에서 삽입된 소프트웨어 결함이나 오류에 의한 고장발생 현상이나 고장발견사상을 고려한 소프트웨어 신뢰도성장모델(Software Reliability Growth Model: SRGM)을 고려한다[1]-[3]. 여기서 예상되는 경로로 소프트웨어의 동작이 수행되지 않는 경우를 고장으로 정의한다. 이러한 모델을 적용시 소

프트웨어 신뢰성 목표치의 달성시기 및 추진상황을 예상할 수 있고 소프트웨어 배포시 예상되는 품질을 판단할 수 있으며, 시험 종료 후 운용단계에 돌입할 최적배포시기 결정 및 개발비용을 고려한 최적배포시기 등을 정할 수 있다[4],[5].

본 논문에서는 소프트웨어 개발의 최종 단계에서 실시되는 시험과 운용단계에 발생한 고장은 소프트웨어 가동 전·후(즉, 시험 전·후)에 소프트웨어 내에 잠재하고 있는 고유 결함에 의해 발생되는 고장과 시험중(운용중)에 랜덤하게 삽입되는 결함에 의해 발생하는 소프트웨어 고장 등 2종류로 구분하여 가정한다. 시험에 의해 검출된 고장의 수정은 불완전하게 수행되는 불완전디버깅(Imperfect Debugging: ID) 환경[2],[6]을 고려할 때 의미가 있으며, 기존의 모델들은 이러한 실제 상황이 잘 고려되지 않았다.

기존에 제안된 불완전디버깅을 고려한 모델은 디버깅작업의 완전성, 즉 결함의 수정이 성공 또는 실패에 대한 사상만을 다루었고[7],[8] 디버깅시 새로 삽입되는 고장의 가능성을 고려한 것은 미미하다. 이러한 소프트웨어 에러 발견사상에 대해 비동차 포아송과정(Non-Homogeneous Poisson Process: NHPP)[9]을 도입시 기존 모델에 비해 적합성이 높은 소프트웨어 신뢰도성장모델을 구축한다. 한편으로는 소프트웨어 신뢰성 평가에 유용한 정량적인 평가척도를 도출하고 실제 데이터를 적용하여 모델 파라미터를 회귀분석을 활용하여 추정한다. 이러한 추정 결과에 대한 적용 예를 제시하고 기존 모델과의 적합성 비교를 한다.

II. 모델의 개념

소프트웨어 동작 환경 하에서 발생한 소프트웨어 고장은 다음과 같이 표현할 수 있다.

- 소프트웨어 내에 존재하는 결함에 의해 발생하는 고장
- 불완전 디버깅에 의해 삽입된 결함에 의해 발생되는 고장

1개의 소프트웨어 고장은 하나의 결함에 의해 발생되는 경우, 발생된 소프트웨어 고장의 원인과 결함은 위의 2가지 경우로 구분되어야 한다. 첫번째 소프트웨어 고장발생률은 운용시간 t 마다 검출된 고유 결함의 2종류의 시간적 경향($a_i(t)$, (i=1, 2))을 고려하여야 한다. 2번째 소프트웨어 고장발생률은 운용시간에 관한 일정한 λ(λ >0)를 적용할 수 있다. 즉 디버깅 작업에 결함이 삽입되는 것이 불명확하기 때문에 이러한 시간의 경우에는 소프트웨어 고장을 일으킬 확률이 발생하지 않아 운용시간에 관한 평균적인 일정한 λ 를 적용한 소프트웨어 고장발생률을 적용한다. 따라서 2가지 경우의 소프트웨어 고장발생 현상을 동시에 고려하는 경우 운용시간에 의한 소프트웨어 고장발생률은 (1)과 같이 표현된다.

$$h_i(t) = \lambda + a_i(t), (i = 1, 2) \qquad (1)$$

(1)에서 시간구간 [0, t]에서 발생될 총 기대 소프트웨어 고장 수(발견될 총 기대 결함 수)는 (2)와 같다.

$$H_i(t) = \lambda t + A_i(t),$$
여기에서, $A_i(t) = \int_0^t a_i(x)dx.$ (2)

본 논문에서는 (1)의 $h_i(t)$를 강도함수, (2)의 $H_i(t)$를 평균치 함수라고 한다.

NHPP에 의거한 소프트웨어 신뢰도성장모델을 (i=1, 2)에 대해 논하면

$$P_r[N(t)=n] = \frac{[H_i(t)]^n}{n!}\exp[-H_i(t)], (i=0,1,2,...) \qquad (3)$$

$$H_i(t) = \int_0^t h_i(x)dx \qquad (4)$$

(3)에서 N(t)는 시간구간[0, t]에 발생한 총 소프트웨어 고장 수를 표시하는 확률변수이며 $P_r[]$은 확률로 표시된다. 한편 고유결함에 의한 소프트웨어 고장발생 현상을 표시한 (2)의 A_i 및 a_i에 대해서는 NHPP 모델의 지수형 소프트웨어신뢰도성장모델(Exponential Software Reliability Growth Model:

ESRGM)과 S-자형 소프트웨어신뢰도성장모델(S-Shaped Software Reliability Growth Model: SSRGM)을 적용하면 (5), (6)으로 표시된다.

$$\begin{aligned} A_1(t) &= a(1-e^{-bt}), \\ a_1(t) &= abe^{-bt}\,(a,b>0) \end{aligned} \tag{5}$$

$$\begin{aligned} A_2(t) &= a[1-(1+bt)e^{-bt}], \\ a_2(t) &= ab^2te^{-bt},\,(a,b>0) \end{aligned} \tag{6}$$

여기에서 a는 초기 소프트웨어 내에 내장된 고유결함의 기대치를 의미하며, b는 고유결함에 의해 발생되는 1개 당 소프트웨어 고장률을 의미한다. 따라서 NHPP 모델의 평균치 함수는 (7), (8)로 표현된다.

$$H_1(t) = \lambda t + a(1-e^{-bt}) \tag{7}$$

$$H_2(t) = \lambda t + a[1-(1+bt)e^{-bt}] \tag{8}$$

앞에서 서술한 모델은 아래와 같은 성질을 가지며 (2), (5) 및 (6)으로부터 (9)~(11)의 관계식을 얻을 수 있다.

$$\frac{dH_i(t)}{dt} = b[c_i(t) - H_i(t)], (i=1,2) \tag{9}$$

$$c_1(t) = a + \left(\frac{1}{b}+t\right)\lambda \tag{10}$$

$$c_2(t) = a(1-e^{-bt}) + \left(\frac{1}{b}+t\right)\lambda \tag{11}$$

(10)과 (11)의 $c_i(t)$는 임의의 시각 t에 대한 소프트웨어 내에 잔존하는 결함 수를 의미하는 내장된 순간 결함 수이다.

(7)과 (8)의 평균치 함수 $H_i(t)$를 갖는 NHPP 모델은 (9)에 의해 임의의 시각에 대한 단위시간 당 발생하는 소프트웨어 고장 수는 그 시점에서 발견된 결함 수에 비례하는 소프트웨어 고장발생 현상을 나타낸다.

III. 신뢰성 평가 척도와 파라미터 추정

(7)과 (8)의 평균치 함수를 갖는 소프트웨어 신뢰도성장모델로부터 (12)와 같은 신뢰성 평가척도를 도출할 수 있다. 총 운용시간 $t(t>0)$에 대해 시간구간 $(t,\ t+x)(x>0)$에 대한 소프트웨어 고장이 발생하지 않을 조건부 확률은

$$\begin{aligned} R_i(x\,|\,t) &= P_r[N(t+x)-N(t)=0\,|\,N(t)=n] \\ &= \exp[H_i(t)-H_i(t+x)], \\ &\quad (i=1,2;t\ge 0, x\ge 0) \end{aligned} \tag{12}$$

로 주어진다. 이것을 소프트웨어 신뢰도라고 부르며, $P_r[A|B]$는 사상(事象) B가 일어날 조건 하에 사상 A가 일어날 조건부 확률을 의미한다. (12)에 (7), (8)을 대입하면

$$\begin{aligned} R_i(x\,|\,t) &= \exp[-\{\lambda x + ae^{-bt}(1-e^{-bx})\}], (i=1) \\ &= \exp[-\{\lambda x + ae^{-bt}[(1+bt)-(1+b(t+x))e^{-bx}]\}], \\ &\quad (i=2) \end{aligned} \tag{13}$$

으로 표현되고 소프트웨어 고장발생률이나 발생빈도를 표시하는 hazard rate는

$$z_i(x\,|\,t) \equiv \frac{\frac{-d}{dx}R_i(x\,|\,t)}{R_i(x\,|\,t)} = h_i(t+x) \tag{14}$$

로 주어져 소프트웨어 고장의 발생 패턴을 알려주고 있다. (14)에 (7), (8)을 적용하면

$$\begin{aligned} z_i(x\,|\,t) &= \lambda + abe^{-(t+x)}, (i=1) \\ &= \lambda + ab^2(t+x)e^{-b(t+x)}, (i=2) \end{aligned} \tag{15}$$

가 된다. (12)를 이용하여 총 운용시간 t에 대한 평균 소프트웨어 고장발생시간간격(Mean Time Between Software Failures: MTBF)을 도출하면 (16)으로 표현된다.

$$E_i(X\,|\,t) = \int_0^\infty R_i(x\,|\,t)dx, (i=1,2) \tag{16}$$

이것은 조건부 MTBF가 된다. 또 (1)의 강도함수의 역수를 순간 MTBF라고 하고 총 시험시간 t를 (2)의 평균치 함수로 나누면 누적 MTBF를 구할 수 있다[3],[9]. 이것에 대한 수식은 (17), (18)과 같이 표현된다.

$$MTBF_I(t) = \frac{1}{h_i(t)}, (i = 1,2) \quad (17)$$

$$MTBF_c(t) = \frac{t}{H_i(t)}, (i = 1,2) \quad (18)$$

시험구간[0, t_k]에 대해 발생된 총 소프트웨어 고장 수 y_k에 관해서 n개의 실측데이터(t_k, y_k) (k=1, 2, 3,..., n; $0<t_1<t_2<t_3 ...<t_n$)가 관측되었다면 잔차평방(殘差平方) 제곱의 합 S는

$$\begin{aligned} S &= \sum_{i=1}^{n} [y_k - H_i(t_k)]^2 \\ &= \sum_{k=1}^{n} [y_k - (\lambda t_k + A_i(t_k))]^2, (i = 1,2) \end{aligned} \quad (19)$$

이며 모델파라미터를 최소자승법(最小自乘法)에 의해 추정할 수 있다. NHPP 모델 파라미터의 추정법(estimation)에 대해서는 최우도법(Maximum Likelihood Estimation: MLE) 적용이 가능하다.

본 논문에서는 수치를 계산하기에 편리한 최소자승법을 채택한다[2].

여기에서 (7), (8)의 평균치함수 $H_1(t)$, $H_2(t)$를 NHPP 모델에 각각 적용하면 λ 에 대한 비선형 연립방정식을 얻을 수 있다. 이와 같이 비선형 연립방정식을 수치해석으로 파라미터 a, b 및 λ 의 최소자승추정치를 구할 수 있다.

Ⅳ. 신뢰성 평가 예의 고찰

본 장에서는 Ⅲ장에 언급된 소프트웨어 신뢰도성장모델의 적용 사례를 적용하기 위해 일차적으로 시스템시험에서 검출된 고장데이터를 이용한 비모수추정(non-parametric estimation) 방법을 이용해 최적모델의 선정과 P-P plot(Probability-Probability plot, 분위수 대 분위수 그림) 또는 Q-Q plot(Quantile-Quantile plot, 확률 대 확률 그림) 등 그래픽 판별 기법들을 도입하여 적합도 검정을 수행한다. 필연적으로 삽입되는 결함(induced defect)을 고려한 불완전 디버깅 환경 하의 소프트웨어 신뢰도성장모델의 적용 결과를 살펴 본다.

1. 고장 데이터 분석 및 모델 검증

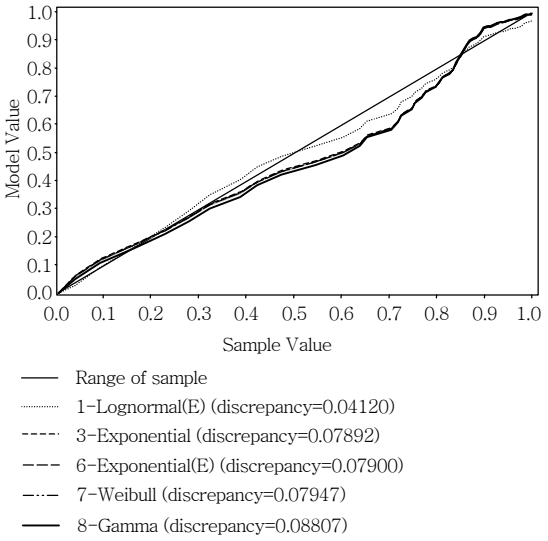
<표 1>은 시스템 시험에서 검출된 고장데이터에 대해 모델 적합도(goodness-of-fit)를 판정한 결과이다. 분석된 결과는 지수분포를 갖는 Lognormal (E) 모델이 가장 적합한 것으로 판단되었으며, 그 뒤를 이어서 지수형, 와이블형(weibull), 감마분포(Gamma distribution)를 갖는 모형으로 판단되었다. 적합도 판정에 사용된 툴은 Experfit™ [10]으로 샘플 데이터에 대해 자동으로 피팅시켜 가장 적합한 모델을 찾는다. 그밖에 모델 파라미터 추정에 수학, 통계 패키지인 Mathematica2.x[11]가 이용되었다.

P-P plot 이나 Q-Q plot은 검출된 데이터에 대한 확률분포 및 데이터 양에 대한 판단근거로 적합도 판정에 중요한 요소가 된다. 종합적인 결과를 놓고 볼 때 우리 프로젝트에 알맞은 모형은 지수분포를 갖는 exponential SRGM으로 이에 대한 연구결과가 기 발표된 바 있다[12],[13].

P-P plot 결과는 실측데이터(sample data)에 대해 확률적으로 얼마나 근사하는가를 표현해주는 geometric 기법으로 (그림 1)에서 실선(slope 1: 청

<표 1> 검출데이터에 대한 모델 적합도 판정

Model	P-P Plot		Q-Q Plot	
	Discrepancy	Rank	Discrepancy	Rank
1-Lognormal(E)	0.04120	1	0.41929	6
3-Exponential	0.07892	2	0.11532	2
6-Exponential(E)	0.07900	3	0.11514	1
7-Weibull	0.07947	4	0.11624	3
8-Gamma	0.08807	6	0.13176	5
9-Weibull(E)	0.08072	5	0.11805	4



(그림 1) Probability-Probability(P-P) Plot

색)에 대해 가장 근접한 모델이 좋은 적합도(good fit)를 나타낸다. 이 기법은 Inverse Gaussian(IG) 분포를 이용하는데 적합도 판정시 location 파라미터는 최우도 방법(maximum likelihood method)을 이용하는 것이 부적합(실측 데이터와 추정치 간의 오차가 크기 때문)하기 때문에 IG 분포를 적용할 수 없는 단점이 있다.

이상적인(esoteric) IG 분포는 시간의존데이터(processing time data)에 대해 적합한 모델로서 널리 알려진 Gamma 분포나 로그정규분포(lognormal distribution), 와이블 분포에 비해 최적의 모델을 제공한다. 이런 모델 특성은 프로젝트 진행 속도에 따라 소프트웨어의 품질이나 신뢰도가 향상되는 과정을 잘 설명해 줄 수 있기 때문이다. 이에 대한 세부 결과(내용)은 <표 1>에 나타내었다.

<표 2> 및 <표 3>은 적용된 데이터에 대한 평균, 분산, 왜도(비대칭: skewness), 로그 우도함수값에 대한 추정치이다. 왜도는 데이터의 분포를 표시하는 척도로 이 값이 non-negative인 경우는 오른쪽으로 기울고 반대인 경우는 왼쪽으로 많이 분포하고 있음을 말해준다. 본 결과는 오른쪽에 데이터가 약간 많이 분포하고 있음을 알 수 있다.

모델의 전반적인 적합도 측정치를 계산하기 위해

<표 2> 각 모델별 분석 결과(평균, 분산, 왜도)

Model	Mean	Variance	Skewness
Sample	15.78421	296.51715	1.88947
1-Lognormal(E)	18.11772	919.67580	9.70308
3-Exponential	15.78421	249.14124	2.00000
6-Exponential(E)	15.78421	249.38412	2.00000
7-Weibull	15.78549	248.11686	1.99359
8-Gamma	15.78421	233.55973	1.93645
9-Weibull(E)	15.78843	246.24433	1.98028

<표 3> 각 모델별 정규화된 로그우도 함수값

Model	Log-Likelihood Function	
	Normalized Value	Rank
1-Lognormal(E)	-3.78964	6
3-Exponential	-3.75901	3
6-Exponential(E)	-3.75950	5
7-Weibull	-3.75901	2
8-Gamma	-3.75770	1
9-Weibull(E)	-3.75946	4

제곱합 대신에 우도값(likelihood value)을 이용한다. 즉, 종속변수와 독립변수 간의 비선형 특성으로 인해 "가능성이 가장 높은 계수 추정치"를 얻기 위해 최우도(maximum likelihood) 절차를 거치게 된다. 다시 말해서 최소자승법(Least Squares Estimation: LSE) 대신에 발생 가능성(likelihood)을 극대화시키는 방법으로 다중회귀분석(multiple regression)에서 실측값과 예측값의 차이인 잔차(residual) 또는 오차의 제곱합과 유사한 방법이다. 로그우도(log likelihood) 값은 실제 우도값의 로그로 이값에 -2를 곱한 -2LL(-2로그우도)이 사용된다. 즉, 이 값이 적은 모델이 적합도가 높게 된다(완전 적합도는 1의 우도를 가지며 이때 -2LL은 0이다).

이 방법은 다중회귀분석에서의 결정계수(R^2)와 마찬가지로 모델의 전반적인 적합도를 나타내는 데 이용된다. 결정계수(決定係數)와의 관계는 (20)과 같이 표현된다.

$$R^2 = \frac{-2LL_{null} - (-2LL_{model})}{-2LL_{null}} \qquad (20)$$

$-2LL_{null}$: 귀무모델(null model)의 로그우도값

본 모델 적합도 판정에서는 감마, 와이블, 지수분포 순으로 판정되었다. 이러한 방법은 앞에서 언급한 그래픽 판정 방법과는 상반된 방법으로 모델의 적합성을 판정하는 데 여러 가지 방법이 이용되고 있다[14].

최적 모델 검증을 위해서 수행하는 과정중에 하나가 데이터에 대한 가설검정(hypothesis test)을 위해서 신뢰구간(confidence interval)에 따른 모델 검정이 수행되는데 대표적으로 카이 제곱 검정(Chi-Square test), A-D 검정(Anderson-Darling test), K-S 검정(Kolmogorov-Smirnov test)을 수행하여 기각여부를 결정하는 것이다.

<표 4>는 최적 모델로 선택된 신뢰성 모형에 대한 K-S 검정 결과이다. 검정에 적용된 데이터는 시스템 시험에서 139주 동안 보고된 고장중에서 각 분야 전문가의 리뷰에 의해서 문제점으로 판정되어 변경이력관리시스템에 정식 등록, 해결된 고장데이터(failure data)를 이용하였다.

(그림 2)는 고장데이터의 시간(x축)에 따른 분포함수에 대한 결과로서 신뢰구간 90%로 주어진 경우에 대해 신뢰도 모형간 차이를 나타낸 그림이다. 지수형 모형에 대해 시스템 시험 초기 15주 전후를 제외하고는 y축의 값 0에 상당히 근접해 있음을 알 수 있다. 이것은 해당 모형(로그 함수 및 지수분포, 와이블 분포함수)들에 대해 적합함을 말해주고 있다.

(그림 3)과 (그림 4)는 고장 분포 비율에 따른 분포함수와 생존함수(survival function)를 나타내었다. 그림에서 알 수 있듯이 시스템 시험 시작 후 82주 정도가 지난 시점에서 신뢰도가 목표치(소프트웨어 배포 기준)인 95%에 도달하는 것으로 추정되었다.

(그림 4)는 대표적인 회귀분석 모형(regression analysis model)인 Cox의 비례위험 모형(proportional hazard model)에 의해 분석된 결과이다. 즉, 정량적이고 정성적인 다변량 분석(multivariate analysis) 모형[14]인 Cox 모형[15]을 이용하여 시험시간(x-value)에 의존하는 순간 고장발생률에 대한 생존함수[s(x)]의 변화 추이 곡선이다.

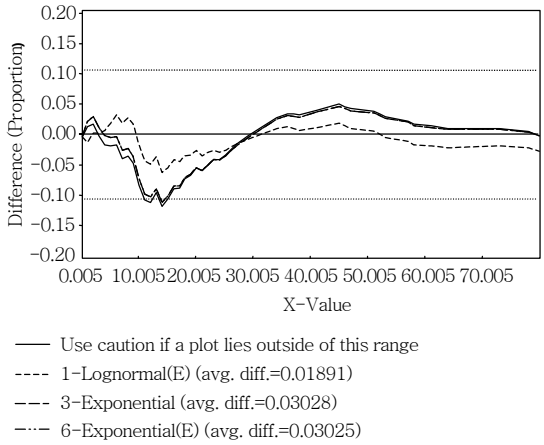
<표 4> 샘플 데이터에 대한 K-S 검정 결과

Kolmogorov-Smirnov Test with Model 1-Lognormal(E)

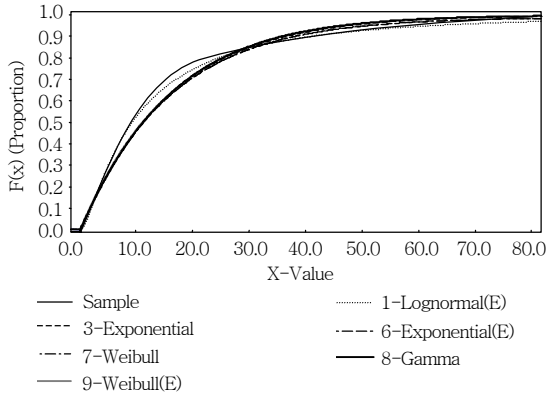
Sample size	139
Normal test statistic	0.07954
Modified test statistic	0.93782

Note: No critical values exist for this special case. The following critical values are for the case where all parameters are known, and are conservative.

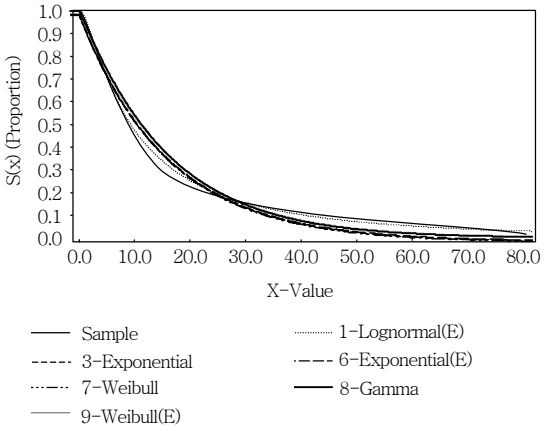
Sample size	Critical Values for Level of Significance				
	0.150	0.100	0.050	0.025	0.010
139	1.126	1.211	1.343	1.464	1.610
Reject?	No				



(그림 2) 각 신뢰도 모형별 분포함수 비교 결과



(그림 3) 비례위험 모형에 따른 분포함수



(그림 4) 각 모형별 생존함수 분석 결과

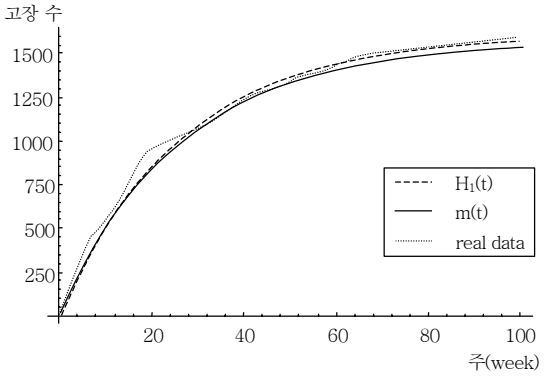
지수분포는 생존함수로서 가장 단순한 형태를 지니며, 와이블 분포는 간단하지 않는 비례적 위험함수의 가정이 성립하는 분포로 선형로그(log-linear) 모형에 속한다. 이러한 선형 로그모형에 속하는 분포들은 로그정규분포(lognormal distribution), 로그-로지스틱분포(log-logistic distribution) 모형 등이 있다. 그 중에서 실측 데이터에 가장 근접한 모형은 로그정규분포로 판정되었다(<표 1> 참조).

각 모형별 추세곡선은 지수 함수적으로 감소하는 경향을 띠며, 45주 근처에서 순간고장률이 0.1개 이하로 발생되고 있음을 (그림 4)를 통해 알 수 있다.

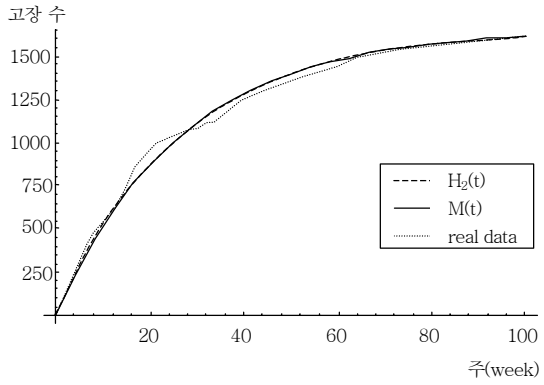
2. 삽입에러를 고려한 모델 적용 결과

여기서 적용된 데이터 그룹은 75주 동안 시험한 실측데이터(t_k, y_k) (k=1, 2, 3,..., 75)를 불완전 디버깅 환경을 고려한 2개의 모델에, 즉 (7), (8)의 평균치 함수 $H_1(t)$, $H_2(t)$를 갖는 NHPP 모델을 적용하는 경우를 고려하였다. 평균치 함수 $H_1(t)$, $H_2(t)$를 갖는 NHPP 모델을 정의하면 Newton-Raphson의 비선형연립방정식의 해를 구하면 된다. 여기서 a(총 기대잔존에러 수), λ(고장률, hazard rate) 값의 추정치를 구해 지수형 또는 S-자형(S-Shaped) 모델에 적용한다.

(그림 5)에서 지수형 소프트웨어 신뢰도성장모델의 (5)의 $A_1(t)$를 평균치함수로 하는 m(t)(모델 m이라 칭함)의 추정결과와 $H_1(t)$의 추정치 H(t) 및 실측데이터를 표시하였다. 그림에서 dash line은 $H(t)$ 추정치, 점(spot)은 실측데이터, 실선은 m(t)의 추정치를 의미한다. λ 에 의해 시험공정에서 삽입된 에러는 약 331개 정도로 추정되며, 시험공정에서 발견된 에러는 1647-331=1316개로 추정된다.



(그림 5) 모델 H_1 및 m의 추정된 평균치함수



(그림 6) 모델 H_2 및 M의 추정된 평균치함수

또 (그림 6)에 대해서는 지연 S-자형 신뢰도성장모델, (16)의 $A_2(t)$를 평균치 함수로 NHPP 모델(모델 M으로 칭함)에 적용한 결과 $M(t)$를 표시하였다.

시험에서 삽입된 결함은 302개, 시험에서 발견된 고유결함에 의한 결함 수는 1566.7-302=1264개로 추정되었다.

적용된 2개의 모델(m과 M)에 대한 세부 추정 결과는 <표 5>와 같다.

<표 5>에서 모델에 대한 MSE는 모델의 실측치와 추정치 간의 평균편차제곱을 나타내는 것으로

<표 5> 제안된 2개 모델에 대한 추정치

모델	a	b	λ	MTBF	MSE
m	1647.51	0.03747	0.017	0.269131	5.87768
M	1566.7	0.08831	0.0026	0.820921	0.139233

(21)과 같이 표현된다.

$$MSE = \frac{1}{n}\sum_{K=1}^{n}(y_k - Y_k)^2 \qquad (21)$$

Y_k : y_k 의 추정치[$m(t_k)$의 추정치를 의미]

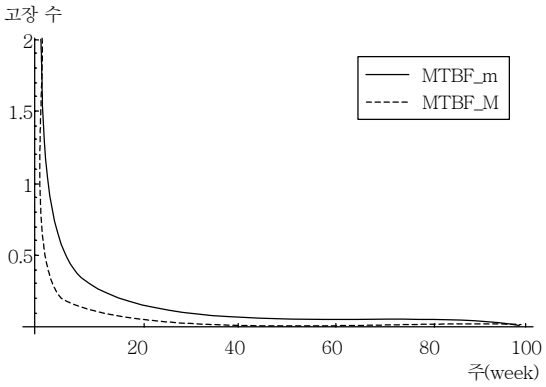
(그림 5), (그림 6)으로부터 불완전 디버깅 환경에서 랜덤하게 삽입된 결함(induced fault)을 고려하는 경우 발생된 소프트웨어 고장 수(결함 수)는 (5), (6)으로 표현된 기존의 소프트웨어 신뢰도성장모델에 대해서 수렴하는 경향을 띤다. (그림 7)은 삽입에러를 고려한 2개의 모델에 대한 평균고장 간 시간(MTBF)에 대한 추정 결과를 나타내었다.

(그림 7)은 시험시간에 따른 순간 MTBF 추정치를 나타냈는데(x축은 시간으로 week, y축은 결함수) 이것은 앞에서 서술한 (18)에 의한 추정치로 2개의 모델(m, M)에 대한 결과이며, <표 1>에 언급한 MTBF 결과는 시험 종료 시점의 (16)에 의해 계산된 조건부 MTBF를 의미한다. 이 비교결과에서는 지수형 신뢰도성장모델(m-모델)보다는 지연 S-자형(delayed S-Shaped) 신뢰도성장모델(M-모델)이 우수한 것으로 나타났다. 이것은 과거의 연구결과[13]에서도 밝혀졌으며, 대형 교환시스템 개발시 시스템시험에서 검출되는 고장발생 패턴은 결함검출과 동시에 해결된다는 지수형 신뢰도성장모델의 가정보다는 결함의 검출과정과 해결과정을 분리하여 해석하는 방법이 시스템 개발에 현실적으로 가깝다는 것을 보여준다[16].

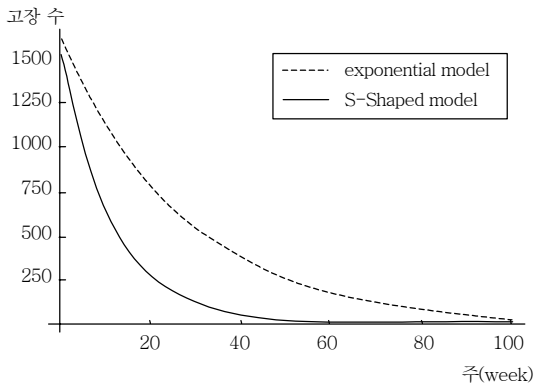
(그림 8)은 앞에서 서술한 NHPP 모델의 (7), (8)의 평균치 함수를 이용하여 추정한 소프트웨어 내에 잔존하고 있는 잔존결함(N(t))에 대한 결과이다. 시스템 시험 경과에 따른 소프트웨어 잔존결함의 추이는 모델의 특성에 따라 약간의 차이는 있으나 우리가 경험한 프로젝트 수행결과는 S-자형에 가깝고 추정결과도 지수형보다는 지연 S-자형이 더 우수한 것으로 분석되었다(empirical analysis result가 real data와 비교시 지수형보다 더 근사함).

지연 S-자형은 약 65주 근처에서 잔존에러가 5개 이내로 줄어드는 반면 지수형은 100주가 지나야 비슷한 수준으로 소프트웨어 내에 에러가 잔존하는 것으로 추정되었다. 이것은 신뢰도평가모델의 가정(assumption)에 의한 것으로 시험 전 소프트웨어 내 총 기대 잔존에러 수와 고장검출률이 지수형 모델에서 높게 추정되기 때문이다.

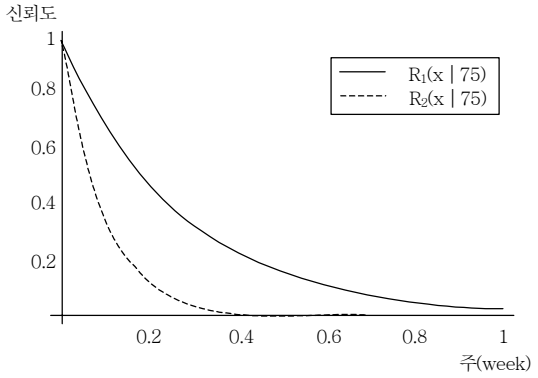
(그림 8)에서 알 수 있듯이 모델의 결함 검출률(<표 1>의 b 값 참조)이 높으면 fault에 대한 expose rate가 좋기 때문에 소프트웨어 내의 잔존결함이 줄어들어 안정된 소프트웨어를 배포(release)할 수 있기 때문이다. 그러나 실제 프로젝트 수행 전에는 시



(그림 7) 시험시간 경과에 따른 MTBF 추정치



(그림 8) 소프트웨어 잔존결함 추정치



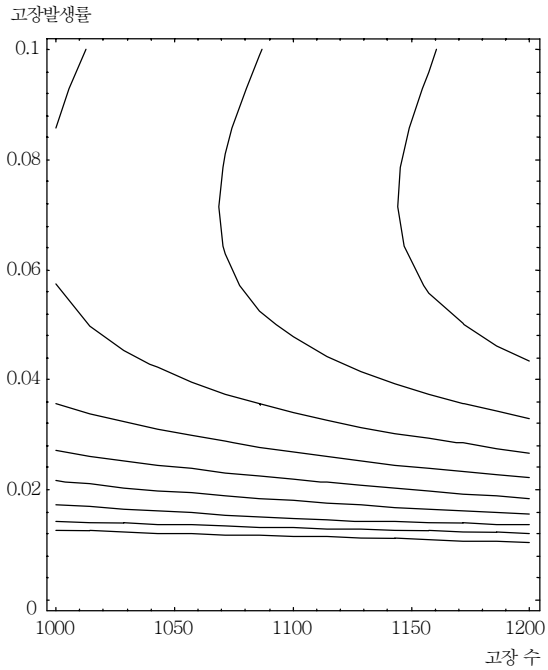
(그림 9) 소프트웨어 신뢰도 추정치 비교[R(x|75)]

험자원(시험인력, 시험도구, 개발환경 등)의 투입에 따른 시험효과의 변화에 의해 좌우될 수 있기 때문에 프로젝트 관리자는 시험진행에 따른 적절한 자원의 투입을 고려하여야 한다. 즉, 소프트웨어 신뢰도와 개발비용(시험시간이 길어지면 신뢰도는 향상되나 비용이 추가됨) 간의 trade-off 문제로서 신뢰도 목표치의 설정 및 비용을 고려한 모델을 적용하여 최소의 비용으로 최적의 신뢰성을 갖는 소프트웨어 개발방법이 적용되어야 한다. 이러한 측면(stopping rule)의 연구결과도 많이 발표되고 있다[17]-[20].

(그림 9)는 시험시작 75주 후(t=75)의 소프트웨어 신뢰도를 운용시간(x)에 따른 변화를 추정한 결과이다. (13)에 의한 소프트웨어 신뢰도 추정치를 2개의 모델에 적용하여 비교한 데이터로서 그림에서 x축은 운용시간, y축은 소프트웨어 신뢰도를 의미한다.

소프트웨어 신뢰도의 변화는 운용시간에 따라 S-자형 모델이 급속히 떨어지는 반면 지수형 모델은 완만한 곡선을 이루는 것으로 추정되었다.

(그림 10)은 시스템 시험에서 검출된 결함 데이터들을 해결해 나가는 과정에 따른 시스템의 고장발생률(b=0.074)을 기준으로 표시한 분포 곡선으로 5주 단위의 시험그룹(test group) 데이터를 이용하여 표시한 예이다. 즉 시스템 시험에서 검출, 해결된 고장데이터가 1,000~1,200개 정도로 시험시작 후 54주가 경과된 시점의 고장 발생 수 및 발생률에 대한 관계를 등고선 형태의 분포 곡선(contour plot)으로



(그림 10) 결함 해결에 따른 고장발생률 분포

나타낸 것이다.

그림을 살펴볼 때 아직도 많은 결함이 시스템 내에 존재하고 있음을 알 수 있으며, 앞에 언급한 각종 추정결과(총 결함 수 및 MTBF, 잔존 결함 수 등)들과 비교해 볼 수 있다.

V. 결 론

본 논문에서는 소프트웨어 고장발생을 소프트웨어 개발의 최종단계인 시험단계와 배포 후 운용단계에서 소프트웨어 내에 잠재하고 있는 고유결함에 의한 소프트웨어 고장발생 현상과 소프트웨어가 운용중에 불완전 디버깅에 의해 발생되는 2가지 고장발생 현상을 가정하였고 소프트웨어 신뢰도성장모델을 구축하여 신뢰성평가의 예를 보였다. 또 기존의 신뢰도성장모델과 비교하여 모델의 특징을 설명하였다.

향후 연구사항으로는 불완전디버깅 환경에서의 에러 삽입 및 제거에 대한 실험적 분석과 관측된 데이터의 계측이나 수집방법을 고려하여 모델의 타당

성을 면밀히 검토하는 것이다. 또한 모델 파라미터 추정법에 최소자승추정법을 채택한 것과 최우도추정법에 의해 추정된 결과를 비교시 관측 데이터에 대한 모델의 적합성을 높이는 것이 필요하다.

참 고 문 헌

[1] S. Bittanti, ed., "Software Reliability Modeling and Identification," Springer-Verlag, Berlin, 1998.

[2] J.D. Musa, A. Lannino, and K. Okumoto, "Software Reliability: Measurement, Prediction, Application," McGraw-Hill, New York, 1987.

[3] 山田 茂, "ソフトウェア信頼性モデレ基礎と應用," 日科技研出版社, 1994.

[4] M.L. Shooman, "Software Engineering: Design, Reliability, and Management," McGraw-Hill, New York, 1983.

[5] M. Ohba and X.M. Chou, "Does Imperfect Debugging Affect Software Reliability Growth?," *Proc. 11th Int. Conf. Software Engineering*, May. 1989, pp. 237-244.

[6] J.G. Shanthikumer, "A State and Time Dependent Error Occurrence Rate Software Reliability Model with Imperfect Debugging," *Proc. National Computer Conf.*, June 1981, pp. 311-315.

[7] 山田 茂, "ソフトウェア信頼性評價技術," HBJ出版局, 1989, pp. 73-153.

[8] J.K. Lee, S.K. Shin, S.S. Nam, and K.C. Park, "Software Reliability Prediction Incorporating Information from a Similar Project(ACE64/256)," *Electronics and Telecommunications Trends*, Vol. 15, No. 5, ETRI, Oct. 2000, pp. 94-102.

[9] 山田 茂, 得能 貢一, 井上 圭, "Optimal Release Problems with Software Reliability/Safety Based on Cost Criteria," 電子情報通信學會論文誌, Jan. 1999, pp. 64-72.

[10] 山田 茂, 大寺浩志, "ソフトウェア信頼性：理論と實踐的應用," Software Research Center SE Series, Feb. 1990, pp. 317-339.

[11] S. Yamada and S. Osaki, "Optimal Software Release Policies with Simultaneous Cost and Reliability Criteria," *European J. Operational Research*, Vol. 31, No. 1, 1987, pp. 46-51.

[12] K. Okumoto and A.L. Goel, "Optimal Release Time for Software System Based on Reliability and Cost Criteria," *J. System and Software*, Vol. 1, No. 4, 1980, pp. 315-318.

[13] 이재기외 1, "기능 블럭으로 구성된 대형 교환 소프트웨어," 대한전자공학회논문지 S편 제 23권 2호, 1998. 10., pp. 1022-1030.

[14] 이재기외 2, "고장 데이터 분석을 통한 교환 소프트웨어 특성 연구," 한국통신학회논문지 제 23권 8호, 1998. 10., pp. 1915-1925.

[15] 송혜향, 정갑도, 이원철, "생존분석," 청문각, 2002, pp. 67-105.

[16] Averill M. Law, "ExperFit User's Guide," ExperFit Software, Sep. 2002.

[17] S. Yamada and S. Osaki, "Optimal Software Release Policies with Simultaneous Cost and Reliability Criteria," *European J. Operational Research*, Vol. 31, No. 1, 1987, pp. 46-51.

[18] 이재기외 2, "신뢰성 평가 척도를 중심으로 한 교환 소프트웨어 최적배포 시기 결정 및 신뢰도 평가," 한국정보처리학회논문지 제 6권 3호, Mar. 1999, pp. 615-621.

[19] 여운승, "사회과학과 마케팅을 위한 다변량 행동 조사," 민영사, 2000, pp. 404-477.

[20] Stephen Wolfram, "Mathematica User's Guide," Wolfram Research, 2000.