유사 프로젝트(ACE64/256)로부터 얻은 경험 데이터에 의한 소프트웨어 신뢰도 예측

Software Reliability Prediction Incorporating Information from a Similar Project(ACE64/256)

이재기(J.K. Lee) 시스템종합팀 선임연구원
신상권(S.K. Shin) 시스템종합팀 연구원
남상식(S.S. Nam) 시스템종합팀 책임연구원, 팀장
박권철(K.C. Park) 교환기술연구부 책임연구원, 부장

시험기간 동안 수집된 고장 데이터를 이용하여 소프트웨어 신뢰도를 예측할 수 있는 모델은 많으나 이 예측 방법은 정확하지 못하며, 특히 초기 시험 단계에서는 더욱 더 부정확하여 예측자들은 이러한 소프트웨어 신뢰도 모델의 적용을 주저한다. 한편 소프트웨어 신뢰도 성장 모델은 유사 프로젝트나 개발 초기에 얻은 정보를 가지고는 신뢰도 예측 데이터로 활용이 불가능하다. 예를 들면 최근의 소프트웨어 시스템들은 항시 유사 프로젝트들로부터 활용이 가능한 일련의 정보와 동일 응용 영역의 초기 또는 최신의 정보들이 변경, 개선되기 때문이다. 본 논문에서는 유사한 프로젝트로부터 얻은 공통의 데이터들을 활용하여 소프트웨어 신뢰도를 예측할 수 있는 방법들을 제안한다. 특히 일반적으로 사용되고 있는 Goel-Okumoto(G-O) 모델이나 고장 검출률을 이용하거나 시험 데이터를 활용하는 방법 등을 이용하여 모델 파라미터를 추정하고 실제 프로젝트 수행중에 얻어진 각종 결과를 토대로 해서 Numerical Algorithm이 아닌 통계적인 관점의 분석 결과와 MLE(Maximum Likelihood Estimation) 추정 방법 등을 동원하여 초기에 우리 프로젝트에 맞는 정확한 소프트웨어 신뢰도 평가 방법을 제안하였다.

I. 서 론

실제 사용되고 있는 소프트웨어 신뢰도 평가 방법은 시험기간 동안 신뢰도 모델을 적용하는데 예를 들면 Musa(1987)나 Lyu(1996)가 제안한 모델로 정확한 신뢰도 평가를 위해서 오랜 시스템 시험기간 동안 수집된 대량의 고장 데이터가 요구된다[1, 2]. 이러한 통계적 방법은 실제로 여러 가지 문제를 안고 있다.

소프트웨어 개발자는 가능한 한 용이한 방법의 신뢰도 모델 적용과 품질 계획에 알맞은 평가 모델에 관심이 많게 된다. 일반적인 신뢰도 평가 모델은 단지 특수한 소프트웨어에 대해 소프트웨어 시스템의 특정 버전의 소프트웨어에 대한 고장 데이터를 이용한 신뢰도 평가를 제공할 뿐이다.

조직 내에서 유사 프로젝트로부터 얻은 경험 데이터는 실제 소프트웨어 신뢰도 예측에 정확을 기할 수 있기 때문이다. 왜냐하면 최근의 대형 시스템은 동일 환경에서 초기 버전 뿐만 아니라 개발 완료시점까지 동일한 방법으로 변경, 개발되기 때문이다. 비록 두 개의 시스템이 다르다 할지라도 유사한 방법으로 개발되기 때문에 일부 정보는 신뢰도 예측에 유용한 데이터로 사용될 수 있다. 이러한 특수 경우는 동일한 개발 프로세스 즉 설계 및 사용언어 등을

포함한 시험 방법, 개발환경, 응용 소프트웨어 등 두 개의 시스템을 위한 동일 정보를 프로젝트에 적용해 볼 필요가 있다.

본 논문에서는 이러한 경험적인 데이터를 이용하여 초기의 신뢰도 예측이나 좀더 정확한 예측을 하기 위한 방법을 제안한다. 또한 대형 교환 시스템인 TDX-ATM 교환 시스템 개발 프로젝트에서 얻은 정보들을 정리하고 체계적인 소프트웨어 신뢰도의 추정을 위해 시스템 시험기간 동안 수집된 고장 데이터를 분석하고 통계적 방법을 이용하여 신뢰도가 성장되어 가는 과정을 추적, 적용하였다.

초기 개발 시스템(ACE64)의 모델 파라미터 추정 결과를 최종 개발 시스템인 ACE256 시스템에 적용하여 최종 시스템 개발에 대한 예측을 해보았다. 이러한 접근 방법은 1973년 Littlewood와 Verrall이 제안한 전형적인 경험 모델(Traditional Bayesian Model: TBM)이 아닌 실제 검출 데이터를 활용한 사례 연구이다[3]. 이 방법은 1991년 Xie의 소프트웨어 신뢰도 성장 모델 등 여러 가지 모델들이 있으나 우리의 프로젝트에 쉽게 적용해 볼 수 있는 G-O 모델[G-O 79]을 채택하였다[4-6]. 그 이유는 본 모델이 초기 프로젝트에서 얻어진 데이터를 활용할 수 있고 신뢰성 데이터에 대한 투명한 해석이 가능하기 때문이다. 즉, 모델의 두 개 파라미터인 초기 고장 수와 시험에서 수집된 고장 검출률(Fault Rate: FR)을 활용할 수 있기 때문이다. 소스 라인 수에 대한 초기 고장 수와의 관계 등을 이용할 수 있는데 이러한 매트릭스(matrics)는 소프트웨어 고장 수를 추정하는 데 좋은 방법은 되지 못한다. 이런 경우 두 시스템의 고장 검출률을 동일 시험 방법과 툴, 환경을 이용하여 추정할 수 있다. 즉, 초기 개발 시스템에서 얻은 정보를 사용하여 안정적이며 손쉽게 추정을 할 수 있다. 더욱이 초기 신뢰성 평가에 대한 가용 데이터 없이 유사 프로젝트로부터 얻은 경험 데이터 정보로 소프트웨어 신뢰도 추정이 가능하다.

본 논문의 구성은 II장에서 초기 개발 시스템인 ACE64 시스템의 시험 데이터에 대해 논하고 소프트웨어 신뢰도 모델 중 가장 널리 이용되고 있는 G-O 모델을 이용하여 각종 모델 파라미터들을 추정한다. 또 1985년 Yamada and Osaki, Bondi and Simonitti(95), Nara(95) 등이 제안한 모델 파라미터들을 활용한다[7-9]. III장에서는 후속 시스템인 ACE256 시스템에 대한 모델 파라미터 추정에 대한 이슈에 대해 논하고 두 개의 시스템에 대해 초기 개발 시스템의 초기(initial) 고장 검출률을 적용하여 고장 수를 추정한다. IV장에서 초기 고장 검출률의 적용이 없는 경우와 상호 비교 분석하고 초기 Life Cycle에 적합한 소프트웨어 신뢰도 예측 모델인 Phase-based model을 후속 시스템에 적용하여 결과를 분석한다[2]. 끝으로 향후 연구방향을 제시하고 맺는다.

II. 초기 시스템의 G-O 모델 적용

소프트웨어 시스템은 조직 내에서 동일한 개발 방법과 툴을 이용하여 개발되고 있다. 즉 새로운 시스템에 이전에 개발한 시스템의 버전을 개선하거나 변형하여 개발되는 것이 일반적인 방법이며, ACE64 및 ACE256 시스템도 위와 동일한 방법으로 추진되었다. 특히 초기 프로젝트에서 사용된 시험 정보는 신규 개발 시스템에 그대로 적용되었다. 이러한 접근 방법은 시간을 단축시키고 쉽게 신규 프로젝트에 적용할 수 있기 때문이다.

초기 시스템에 G-O 모델을 적용하여 얻어진 정보는 후속 개발 시스템의 신뢰도 평가를 추정할 수 있는 부분 연구 방법이 되기 때문이다. 앞에서 언급한 최대우도추정(MLE) 방법에 대한 설명은 III장에서 논하기로 하고 초기 시스템에 대해 배포 전까지 행해진 73주에 대한 고장 데이터를 <표 1>에 나타내었다.

Goel-Okumoto 모델은 식(1)의 평균치 함수 값으로 표시되는 비동차 포아송 과정(Non-Homogenuous Poisson Process: NHPP)으로 간단히 표현된다(Yamada and Osaki 85).

$$\mu(t) = a(1 - e^{-bt}) \qquad (1)$$

<표 1> 초기 개발 시스템(ACE64)의 주별 고장 검출 수

주	고장 수	주	고장 수	주	고장 수	주	고장 수	주	고장 수
1	6	16	1	31	20	46	0	61	0
2	7	17	1	32	2	47	0	62	1
3	5	18	3	33	10	48	2	63	1
4	33	19	4	34	7	49	0	64	0
5	15	20	6	35	9	50	1	65	1
6	11	21	9	36	9	51	0	66	0
7	11	22	7	37	2	52	1	67	0
8	6	23	12	38	3	53	0	68	0
9	9	24	5	39	0	54	1	69	0
10	5	25	4	40	2	55	1	70	1
11	4	26	6	41	6	56	0	71	1
12	4	27	9	42	6	57	0	72	0
13	4	28	12	43	0	58	1	73	1
14	4	39	2	44	1	59	1		
15	5	30	11	45	0	60	0		

이 모델에서 파라미터 a는 소프트웨어 내에 잠재하고 있는 초기 고장 수(혹은 결함 수)를 의미하며, b의 값은 시험을 진행해감에 따라 신뢰도가 성장되어 가는 과정의 고장(결함) 검출률을 의미한다. 이때 고장 밀도에 대한 함수는 식(2)로 표현된다.

$$\lambda(t) = abe^{-bt} \tag{2}$$

위의 모델에서 각 파라미터들은 단위시간 간격으로 발생되는 고장들을 가지고 최대우도추정 방법을 사용해 산출하며, 관측 시간대별로 측정된 고장 수를 가지고 우도 함수로 표시하면 식(3)과 같다.

$$L(n_1, n_2, \ldots\ldots n_k) = \prod_{i=1}^{k} \frac{[\mu(t_i)-\mu(t_{i-1})]^{ni}}{n_i!} * e^{[-\mu(t_i)-\mu(t_{i-1})]} \tag{3}$$

방정식 (3)의 양변에 로그를 취하여 정리하면 식(4)를 얻는다.

$$\ln L = \sum_{i=1}^{k} \{n_i \ln[(t(x_i) - \mu(t_{i-1})] - [\mu(t_i) - \mu(t_{i-1})] - \ln n_i!\} \tag{4}$$

이것을 다시 G-O 모델에 대해서 미분하면 a, b 파라미터를 구할 수 있으며, 식(5)와 같이 표현된다.

$$\frac{\partial \ln L}{\partial b} = \sum_{i=1}^{k} \{\frac{n_i}{a} + e^{-bt_i} - e^{-bt_{i-1}}\} = 0,$$

$$\frac{\partial \ln Ll}{\partial b} = \sum_{i=1}^{k} (\frac{n_i}{e^{-bt_{i-1}} - e^{-bt_i}} - a)(t_i e^{-bt_i} - e^{-bt_{i-1}}) = 0 \tag{5}$$

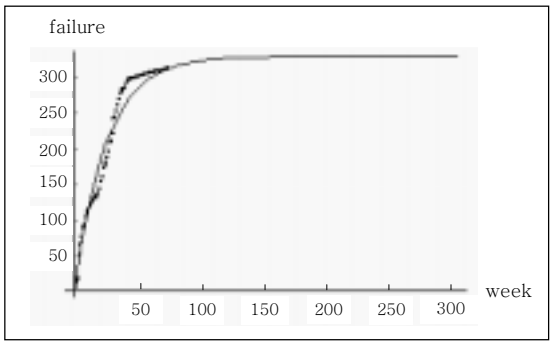
방정식(5)를 풀면 식(6)을 얻는다.

$$a = \frac{\sum_{i=1}^{k} n_i}{1-e^{-bt_k}},$$

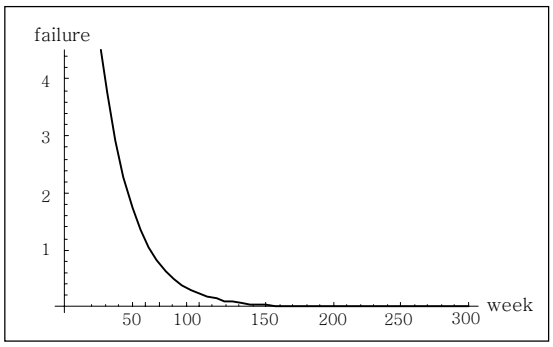
$$\sum_{i=1}^{k} (\frac{n_i}{e^{-bt_{i-1}} - e^{-bt_i}} - \frac{\sum_{i=1}^{k} n_i}{1-e^{-bt_k}})(t_i e^{-bt_i} - t_{i-1} e^{-bt_{i-1}}) = 0 \tag{6}$$

식(6)은 비선형으로 일반해(analytic solution)를 구할 수 없으며 이를 수치적(numerically)으로 풀면 된다.

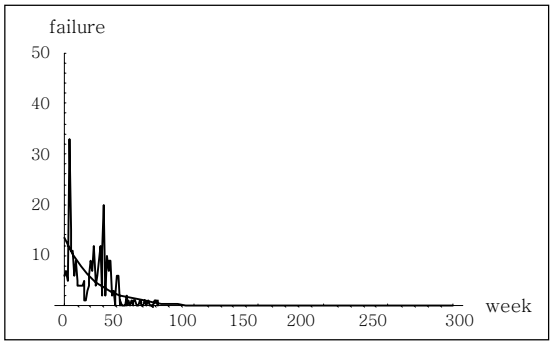
<표 1>의 고장 데이터에 대해 a, b 값을 구하면 a = 329.136, b = 0.0404832를 얻는다. 또 각 주별 누적 고장 수(cumulative failures), 고장밀도(failure intensity), 고장 발견 및 해결(correction and detected failure)에 대한 추이를 분석하면 (그림 1)~(그림 3)과 같다.



(그림 1) 주별 누적 고장 수 추정



(그림 2) 주당 고장 밀도 추정치



(그림 3) 고장 해결 및 발견 현황

(그림 1)에서 점선은 실측 데이터이며 실선은 추정치를 의미한다.

(그림 2), (그림 3)의 x축은 단위시간(주)을 의미하며, y축은 고장 수를 의미한다.

위 모델에서 정확한 신뢰도 추정을 위해서는 대량의 고장 데이터가 필요하여 이러한 데이터의 수집이 용이하지 못하므로 주요 이슈에 대한 언급만 하고 III, IV장에서 비교 분석하기로 한다.

G-O 모델의 두 개 파라미터를 알고 있는 경우는 신뢰도 예측 뿐만 아니라 시험의 진행 여부 등을 결정할 수 있다. 그러나 이러한 정보들은 신규 프로젝트를 시작하는 시점에서 활용이 불가능해 진다. 왜냐하면 유사한 프로젝트로부터 얻은 데이터들이 일관성이 없어서 정확한 신뢰도 예측이 어려워지기 때문이다. 이러한 개념들은 III장에서 다루기로 한다.

III. 후속 개발 시스템의 초기 예측

후속 개발 시스템인 ACE256 시스템의 초기 예측 방법은 초기 시스템 개발 완료 후 진행되었으며, 33주에 걸친 시험 결과가 <표 2>와 같다.

일반적인 접근 방식은 1차 시스템과 독립적으로 데이터를 분석, 적용하는 방법이다. 그러나 초기 신뢰도 예측을 위해서는 1차 시스템의 정보를 활용하는 것이 무난하며, 이 정보를 두 개의 시스템에 모두 적용하여 G-O 모델의 a, b 파라미터를 추정해 보는 것이 필요하다.

1. 초기 고장 수(a)에 의한 추정 방법

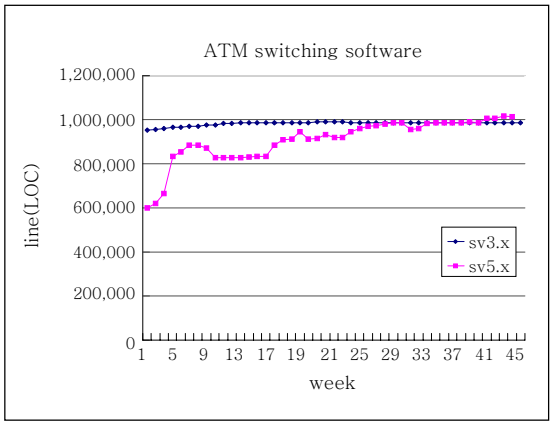
소프트웨어 내의 고장들이 발견되는 것은 모든 프로젝트에서 대단히 중요하게 여기며, 파라미터 a에 대한 연구는 소프트웨어 신뢰도 감사의 최대 정점이 된다. 특히 a는 소프트웨어 규모와도 관계가 있으며, 소스라인(Line of Code: LOC) 및 소프트웨어의 복잡도, 프로그래머의 경험 등에 많은 영향을 미친다. (그림 4)에 초기 개발 시스템 및 후속 개발 시스템의 소프트웨어(sv3.x and sv5.x) 규모를 나타내었다.

소프트웨어 버전 sv5.x는 sv3.x를 기본으로 하여 개발이 추진되었으며, 개발 초기의 일시적인 소스라인의 감소는 컴포넌트들의 통합관리와 시스템의 공통 라이브러리 통합 작업에 의한 현상이다.

소프트웨어 안에 잠재하고 있는 고장들을 추정해 내는 방법들에 대한 연구는 많이 시도되었으나 정확

<표 2> 후속 개발 시스템(ACE256)의 주별 고장 수

주	고장 수	주	고장 수	주	고장 수	주	고장 수	주	고장 수
1	3	8	8	15	6	22	1	29	5
2	1	9	7	16	13	23	1	30	16
3	0	10	7	17	8	24	3	31	2
4	2	11	3	18	19	25	1	32	8
5	4	12	1	19	8	26	2	33	4
6	5	13	7	20	7	27	2		
7	2	14	4	21	2	28	1		



(그림 4) 버전별 소프트웨어 규모

성이 떨어지는 현상이 많아 이에 대한 논란의 소지가 많았다. 이러한 이슈는 1996년 Morgan and Knafl, Ohlsson 등에 의해 다루어졌으며, 고장 수에 대한 추정치는 소프트웨어 매트릭스에 의해 추정되기도 한다[10-12].

<표 2>에서 고장 발견 수가 16주부터 18주 사이에 증가하는 추세를 보이는데 그 이유는 소프트웨어 품질을 향상하기 위한 종합시험이 실시험 환경 하에서 다양하게 치뤄졌기 때문인 것으로 분석되었다.

사용자에게 양질의 서비스를 제공하기 위한 두 차례의 상용시험(Commercial Test) 및 반복시험(Regression Test), 시스템시험이 연속적으로 수행되어 소프트웨어 내에 잠재하고 있는 미 발견 에러(Undetected error)나 시스템의 운용상에 지장이 없는 보이지 않는 결함(일명 Latent fault)을 찾아내는 데 주력하였다.

2. 고장 발견율(b)에 의한 추정 방법

소프트웨어 신뢰도 추정은 고장 발견율(b 값)을 구할 수 있다면 쉽게 해결할 수 있으며, Ⅱ장으로부터 G-O 모델에 의한 최대우도 방정식[식(5)]에 의해 a, b 값을 추정할 수 있다. 그러나 비선형 함수인 b 값을 구하는 데 수학적인 방법이 동원되어야 한다. 특히 b 값은 시험 효과를 나타내는 지수로 잘 정의된 개발 과정과 시험방법, 시험 툴을 이용하는 경우에는 동일한 값으로 추정이 예상되어 신규 프로젝트의 초기 신뢰도 평가에 이용될 수 있다. 그러나 이러한 접근 방법은 확인시험이 수반되어야 한다.

한 가지 접근 방법으로 동일한 b 값을 두 개의 시스템의 버전에 각각 적용해보고 이에 대한 비교·분석을 하여 좀더 정확한 추정을 시도하였다.

제Ⅳ장에서 이에 대한 비교 결과를 상세히 밝히고자 한다.

3. 동일 고장 발견율을 적용한 추정 결과

초기 시스템의 고장 발견율을 b_1, 후속 개발 시스템의 고장 발견율을 b_2라고 할 때 최대우도법에 의한 초기 고장 수(a_2)를 쉽게 추정할 수 있다. 즉,

$$a_2 = \frac{\sum_{i=1}^{k} n_i}{1-e^{-b_2 t_k}}, \quad (b_2 = b_1 = 0.0408832)$$

를 적용하면 ACE256 시스템에 대한 a_2 값을 추정할 수 있는데 이에 대한 결과는 <표 3>과 같다. 또한 <표 4>에 소프트웨어 내에 남아있는 고장 결과

를 추정하였다. 시험시작 20주부터 80주까지 시험하였을 경우를 가정하여 추정한 결과이다.

<표 3> 주별 초기 고장 추정치($b_2 = b_1$)

주	a_2	주	a_2
10	116.219	21	203.046
11	115.961	22	198.922
12	110.899	23	195.244
13	121.281	24	195.157
14	123.907	25	192.141
15	130.887	26	190.965
16	152.050	27	190.004
17	161.698	28	187.769
18	191.966	29	191.521
19	199.957	30	210.845
20	205.895	31	210.180

<표 4> 기대잔존 에러 수(R2) 추정($b_2 = b_1$)

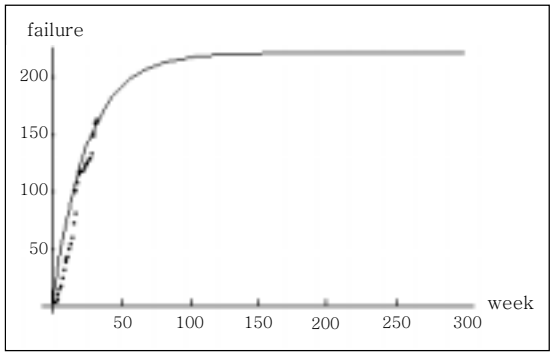
주	R2	주	R2	주	R2
20	13.8766	28	6.9445	44	1.7392
21	12.7264	29	6.3689	46	1.4628
22	11.6714	30	5.8409	48	1.2304
23	10.704	31	5.3568	50	1.0348
24	9.8166	32	4.9127	60	0.4355
25	9.0025	33	4.5055	65	0.2826
26	8.2566	40	2.4585	70	0.1833
27	7.5722	42	2.0678	80	0.0771

이 값의 의미는 소프트웨어 내에 남아있는 기대잔존 에러 수를 의미하며, 파라미터 b 값이 추정되면 고장 수 a는 가용 고장 데이터가 정리되는 즉시 구해질 수 있다. 다시 말해서 시험이 진행됨에 따라 쉽게 변화되는 것을 추정할 수 있을 것이다. 고장 발견율에 대한 추가 정보가 없는 상태의 비교는 IV장에서 논하기로 한다.

IV장에서 논하고자 하는 관점은 자주 문제가 되고 있는 비선형 방정식의 해를 구하는 일반적인 접근 방법으로서 그 값이 줄어드는 경향을 보이는 경우라 할지라도 초기 프로젝트에 적용하지 못하는 정보보다는 안정적인 것으로 간주된다. 일반적으로 고장 발견율이 감소하는 경우는 시험의 질이 떨어지는 것으로 판단되어 새로운 개념의 시험방법 도입과 원인 분석이 뒤따라야 한다. 과거의 데이터 분석 결과에 비추어 볼 때 시험 초기보다는 소프트웨어 배포 단계에 이를 때 최고치를 나타내는 것으로 분석되었으나 이는 상대적이고 항상 동일 환경 하에서 동일 기법에 의한 반복시험으로 수행되어야 하기 때문에 현실적으로 많은 어려움을 안고 있다[13].

IV. 전형적인 접근 방식과의 비교

초기 시스템에서 추정된 정보(sv3.x)를 후속 시스템 버전인 sv5.x에 적용하는 경우 G-O 모델에 적용하여 33주에 걸친 고장 수 및 발견율을 추정하면 (그림 5)와 같다. 이때 적용된 a, b 결과값은 221.14, 0.04088로 시험에서 검출한 실측 데이터인 190여 개 보다 많은 값으로 추정되었다. 이는 앞으로도 많은 에러가 소프트웨어 내에 존재하고 있는 것으로 추정되나 시험이 진행되면 점차 감소하는 것이 일반적인 경향이다.



(그림 5) 주당 누적 고장 발견 수

(그림 5)에서 y축의 값은 고장 수를 의미한다.

실제로 소프트웨어 신뢰도 모델을 성공적으로 적용하는 경우에도 많은 문제점이 발생하고 있다. 이러한 문제를 지적한 연구 결과가 발표되었으며 그 외 다수의 논문들이 발표된 바 있으나 여전히 많은 문제점을 안고 있다[10, 14]. 이러한 문제 제기는 새로운 신뢰도 평가 모델을 정립하는 계기가 되었다. 그밖에 신뢰구간(Confidence interval)에 의한 파라미터 a

에 대한 연구 결과가 '96년 Lawless에 의해 발표되었다[15]. 이 방법에 의한 분산 및 양측의 신뢰구간(two-side confidence interval)에 의한 고장 수 추정 방법은 식(7), (8)로 표현된다.

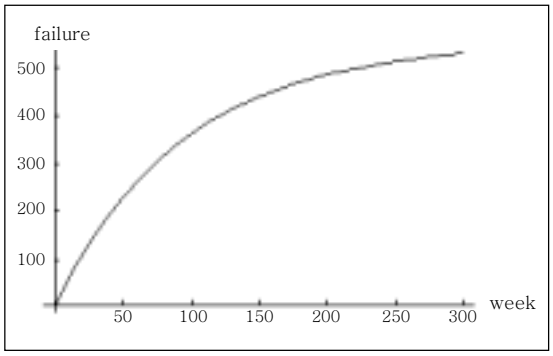
$$Var(a') = [\frac{-1}{\frac{\partial^2 \ln L}{\partial a^2}}]_{a=a'} = a'^2 / \sum_{i=1}^{k} n_i \qquad (7)$$

$$a_L = a' - Z_{\frac{\alpha}{2}} \sqrt{Var(a')}$$

그리고

$$a_U = a' + Z_{\frac{\alpha}{2}} \sqrt{Var(a')} \qquad (8)$$

여기서 $Z_{\frac{\alpha}{2}}$ 는 [100(1+a)]/2th 표준정규분포를 의미한다. 신뢰구간 95%인 경우 고장 발견율($b_2 = b_1$)을 적용한 a_2의 추정치는 (그림 6)과 같다.



(그림 6) 후속 개발 시스템의 누적 고장 수

초기 개발 시스템의 누적 고장 수 예측치인 (그림 1)과 비교해 볼 때 고장 발견비가 약간 떨어지는 것을 알 수 있다. (그림 5)로부터 초기 시스템의 발견율을 이용한 초기 고장 수를 추정하였고 후속 시스템에 대한 정확한 판단을 위해 신뢰구간을 주어 추정해 보았다. 이것의 의미는 a, b 값을 추정하는 방법의 일환으로 고장 발견율을 고정시켜 놓고 좀더 정확하게 값을 추정하고자 하는 것이다. 이렇게 추정된 데이터를 후속 개발 시스템에 대해 초기 신뢰도 평가 모델인 Phase-based model에 적용하여 상호 비교해 보고자 한다[2].

1. Phase-based(Pb) model의 의미

이 모델은 Gaffney와 Davis에 의해 1990년에 제안된 모델로 시험 및 운용중의 신뢰도를 예측하기 위해 고장 통계 데이터를 요구사항 분석, 설계, 구현 단계로 나누어 이 데이터를 이용, 신뢰도를 예측하는 모델이다. 이 모델의 의미는 다음과 같다:

- 개발에 투입되는 능력은 개발 기간 중에 발생한 고장 개수에 의존한다.
- 고장 발견 추이는 단순 증가형이다.
- 소스라인 규모에 대한 평가는 초기 개발에 투입된 노력에 활용이 가능하다. 즉, 고장 밀도는 1,000 라인 당 고장 발생 수로 표현한다.

이 모델은 1,000 라인 당 고장 발견 수와 잔존 고장 수는 각각 아래의 식(9)와 식(10)과 같이 표현된다.

$$\Delta V_t = E[e^{-B(t-1)^2} - e^{-Bt^2}], B = \frac{1}{2\tau_p^2} \qquad (9)$$

$$R_t = E * e^{-Bt^2} \qquad (10)$$

τ_p : defect discovery phase constant

t: fault discovery index
(1: 요구사항 분석, 2: 설계, 3 : 구현,
4: unit test, 5: 소프트웨어 통합,
6: 시스템시험, 7: 인수시험)

E: KLOC(kilo-line of code) 당 총 고장률

ΔV_t: discovered faults per KLOC(time t-1 to t)

R_t: remaining fault at stage t

또 Phase-based 모델의 Cumulative form은 식(11)로 나타낼 수 있다.

$$V_t = E(1 - e^{-Bt^2}) \qquad (11)$$

V_t: Number of fault per KLOC
(discovered through phase t)

2. Pb 모델 적용 결과 및 모델간 상호 비교

앞에서 언급된 고장 데이터는 주로 소프트웨어

Integration 단계에서 실시된 CR(Change Request) 시험에서 검출된 데이터이다. 이 결과는 후속 개발 시스템에 대한 소프트웨어 신뢰도 평가를 위한 Phase-based 모델의 t = 5인 경우와 6인 경우의 중간 성격을 띠어 두 개의 모델간의 비교, 분석 데이터는 우리가 수행한 프로젝트의 시스템에 대한 소프트웨어 신뢰도 예측 모델에 적용해 볼 수 있을 것이다. 우선 Phase-based 모델에 적용한 결과를 살펴보면 <표 5>와 같다.

<표 5>에 의하면 G-O 모델의 고장 발견 개수 즉, 소프트웨어 초기 고장 수가 Phase-based model에 비해 약간 높은 결과로 분석되었으나 이는 Pb 모델의 phase에 따른 fault discovery index 값을 소프트웨어 통합 모델(t = 5)로 적용한 결과로 인해 실제 시험에서 검출된 데이터는 시스템 시험 및 인수시험의 일부 데이터가 혼합된 것과의 차이 때문인 것으로 추정된다. 다시 말해서 1차 개발 시스템인 sv3.x 버전인 경우는 개발확인시험 및 상용시험이 중간에 수행되어 여기서 검출된 시스템 문제들을 일부 적용한 데이터이다. 반면 후속 개발 시스템은 기업으로 배포하기 위한 순수 시험용 버전임을 밝혀 둔다.

<표 5> 버전별 모델 적용 결과 및 비교

버전	GV_t	SLOC	V	B	E
sv3.x	329.136	97.7만	310.686	3.2873	0.318
sv5.x	221.14	83.2만	186.368	3.2873	0.224

* GV_t: Goel-Okumoto(G-O) model의 고장 추정치임.
E 값의 단위는 1,000 라인 당 고장률 수치임.

위에서 언급한 두 개의 모델은 고속으로 많은 데이터를 처리하는 교환 시스템 같은 대형 소프트웨어 시스템 개발시 초기 소프트웨어 신뢰도 평가를 위해 쉽게 적용할 수 있는 모델로서 우리의 프로젝트 성격에 맞게 지속적으로 개선이 필요한 부분이다.

V. 결론 및 향후 연구방향

최근의 시스템 개발 방법은 개발기간의 단축과 빠른 시장의 변화에 대처하기 위해서 초기 개발 시스템의 변형이나 개선을 통하거나 유사 프로젝트에서 사용되었던 툴, 개발 방법론을 활용하기도 한다. 이러한 추세에 부응하기 위해서는 더 많은 노력과 소프트웨어 신뢰도를 정확히 예측하기 위한 많은 고장 데이터의 수집 방법이 뒤따라야 한다.

본 논문에서는 이러한 관점에서 유사 프로젝트에서 얻은 경험 데이터들을 이용하여 동일한 개발 방법을 사용한 신뢰도 측정 방법을 제시한 것이다. 비록 두 개의 시스템이 동일하진 않지만 일차 개발 시스템으로부터 얻은 정보를 활용해 후속 개발 시스템의 신뢰도를 조기에 예측해 볼 수 있는 방법으로 새로운 소프트웨어 신뢰도 예측모델의 하나라고 볼 수 있다. 이러한 정보들은 조직 내에서 시스템 개발 시 재사용될 수 있는 유용한 정보다.

우리의 접근 방법은 G-O 모델에서 사용하고 있는 두 개의 파라미터들에 대한 신뢰할 만한 수준의 물리적인 해석과 시스템 개발 초기에 적용될 수 있는 손쉽고 안정적인 신뢰도 예측 모델로 활용하는 것이다. 다양한 사용자 소프트웨어 신뢰도 모델에 대한 안정도 평가의 연구결과가 발표되었으나 하나의 데이터 set을 가지고 수식적으로 해석한 결과는 현실적으로 극히 미약한 상태다[1, 12]. 그래서 실무자들은 좀더 다른 정보들로부터 새로운 방법을 찾기 시작했으나 유사 프로젝트로부터 얻은 정보를 활용한 소프트웨어 신뢰도 평가 방법은 전무한 상태다.

우리가 시도한 방법의 핵심은 초기 개발 시스템의 정보를 활용하여 간단하면서도 쉽게 평가할 수 있고 또 업데이트 할 수 있는 방법과 전형적인 방법인 초기 프로젝트의 신뢰도를 평가할 수 있는 Phase-based model과의 비교이다. 이런 것들의 가장 중요한 점은 신뢰도가 실제 시험 초기 단계에서 검출되는 데이터를 가지고 평가된다는 사실이다. 향후 연구방향은 유사 프로젝트에서 얻은 경험 정보에 대한 철저한 분석 방법과 실제 프로젝트에 가장 적합한 모델 즉, 정확한 평가가 될 수 있는 기존 G-O 모델에 대한 초기 평가 모델의 개선 작업이 수행되어야 한다.

참 고 문 헌

[1] J.D. Musa, A. Iannino and K. Okumoto, *Software Reliability Measurement, Prediction, Application*, Mc Graw-Hill, New York, 1987.

[2] M. Lyu, *Handbook of Software Reliability Engineering*, McGraw-Hill, New York, 1996, pp. 75 – 115.

[3] B. Littlewood and J.L. Verrall, "A Bayesian Reliability Growth Model for Computer Software. Applied Statistics 22(3)," 1973, pp. 332 – 346.

[4] M. Xie, *Software Reliability Modeling*, World Scientific Publisher, Singapore. 1991.

[5] M. Zhao and M. Xie, "Robustness of Software Release Time," Int'l. Proc. of the 4th Int'l Symp. on Software Reliability Engineering, IEEE Computer Press, Silver Spring, MD, 1993, pp. 218 – 225.

[6] 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, 1979, pp. 206 – 211.

[7] P. Bondi and G. Simonetti, "Evaluating the Reliability of the Software of a Switching System with a Multivariable Model. Software Testing, Verification and Reliability 5(3)," 1995, pp. 181-202.

[8] T. Nara, M. Nakata and A. Ooishi, "Software Reliability Growth Analysis ± Application of NHPP Models and its Evaluation," Intl. Proc. of the 6th Int'l Symp. on Software Reliability Engineering, IEEE Computer Press, Silver Spring, MD, 1995, pp. 250 – 255.

[9] S. Yamada and S. Osaki, "Software Reliability Growth Modeling: Models and Applications," IEEE Transactions on Software Engineering SE-11(12), 1985, pp. 1431 – 1437.

[10] G.J. Kna, "Solving Maximum Likelihood Equations for Two-parameter Software Reliability Models Using Grouped Data," Intl. Proc. of the 3rd Int'l Symp. on Software Reliability Engineering, IEEE Computer Press, Silver Spring, MD, 1992, pp. 205 – 213.

[11] J.A. Morgan and G.J. Kna, "Residual Fault Density Prediction Using Regression Methods," Intl. Proc. of the 7th Int'l Symp. on Software Reliability Engineering, IEEE Computer Press, Silver Spring, MD, 1996, pp. 87 – 92.

[12] N. Ohlsson, M. Helander and C. Wohlin, "Quality Improvement by Identification of Fault-prone Models Using Software Design Metrics," Intl. Proc. of the 6th Int'l Conf. on Software Quality, Ottawa, Canada, 1996.

[13] J.K Lee *et al.*, "An Evolution of Reliability of Large Scale Software of a Switching System," 전자통신동향분석지, 제14권 제4호, ETRI, 1999. 8, pp. 1 – 8.

[14] S.A. Hossain and R.C. Dahiya, "Estimating the Parameters of a Non-homogeneous Poisson Process Model for Software Reliability," IEEE Transactions on Reliability R-42, 1996, pp. 604 – 612.

[15] J.F. Lawless, "Lifetime Data Analysis," Wiley, New York, 1996.