論文 98-23-8-5

고장 데이터 분석을 통한 교환 소프트웨어 특성 연구

정회원 이 재 기*, 신 상 권*, 이 영 목*

A Study on Hypothetical Switching Software Through of the Analysis of Failure Data

Jae-ki Lee*, Sang-kwon Shin*, Young-mock Lee* *Regular Members*

요 약

교환 시스템이 고장과 장애를 일으키면 서비스에 치명적인 영향을 미치게 된다. 다시 말해서 시스템을 제어하는 교환 소프트웨어의 역할은 매우 중요하다. 그렇기 때문에 정량적인 소프트웨어의 품질 평가 방법은 더욱 중요하다. 본 논문에서는 기능 블록으로 구성된 교환 소프트웨어를 시험하여 얻어진 각종 고장 데이터를 수집, 분석하고 이를 이용하여 각 버전별, 개발 전과정에 대한 소프트웨어 신뢰도를 평가해 보고, 기타 고려할 사항에 대해 논한다. 대표적인 2개의 신뢰도 모형(G-O model, S-Shaped model)을 선택하여 소프트웨어 신뢰도를 비교해 보고 품질 향상을 위한 제반 활동과 소프트웨어 개발 프로젝트에 맞는 소프트웨어 신뢰도 모형을 제시하였다.

ABSTRACT

The switching system software is large scale, real-time multi-task system which requires high reliability. The reliability assessment of large-scale software is very important for the success of software development project. For this reason, the software quality measurement is much more important. In this paper, we have learned about the software reliability, method of the analysis of failure data and estimation of software quality. To estimate the software reliability, using the failure data found during of the system test. We apply the two software reliability growth models, named Goel-Okumoto(G-O) and S-shaped model, to estimate the software reliability. Also, we compared with the results and we reviewed fully not only development cycle but validation and verification of the test data, for each software versions This paper presents a software reliability model that suitable the software development project and the activity of quality control for the switching system.

I. 서 론

현대는 급속히 변화하는 정보화 사회에 직면해 있다. 빠르게 변모하는 정보화 사회의 물결은 더 많은 소프트웨어를 요구하기에 이르렀으며, 이러한 추세는 사용자의 다양한 서비스를 제공하는 교환시스템도 예외는 아니다. 특히 고 신뢰성이 요구되는 교환 시스템의 소프트웨어는 실시간을 요구하는 기능과 신뢰성을 가져야 한다. 그렇기 때문에 개발된 교환 소프트웨어의

* 한국전자통신연구원
論文番號 : 97423-1124
接受日字 : 1997年 11月 24日

품질에 대한 정량적인 평가가 더욱 필요해 졌으며, 이런 이유 때문에 소프트웨어 개발에 있어서 신뢰성 평가는 소프트웨어 생산성을 향상시키고 확률, 통계에 근거한 수학적 모델에 의한 정량적인 평가 방법이 소프트웨어 엔지니어링 측면에서 다각도로 연구되고 있다.[3,6,7,8]

소프트웨어 신뢰성 평가가 주요 문제로 등장하는 소프트웨어 테스트 단계에서는 많은 테스트 자원이 투입되고 있다. 즉, 시험 인력(Man. Power), 테스트케이스(Test case), 시험 시간 등 이러한 자원들은 시험 진척 상황에 따라 조정될 수 있다.[11]

본 논문에서는 소프트웨어 테스트 단계에서 투입되는 자원의 변화를 대변할 수 있는 소프트웨어 신뢰도 성장 모델을 이용하여 개발된 소프트웨어의 신뢰성을 시험에서 실측된 데이터를 이용해 평가해보고 추후 소프트웨어 개발 시 고려할 사항 및 개선 사항등에 대해 논한다.

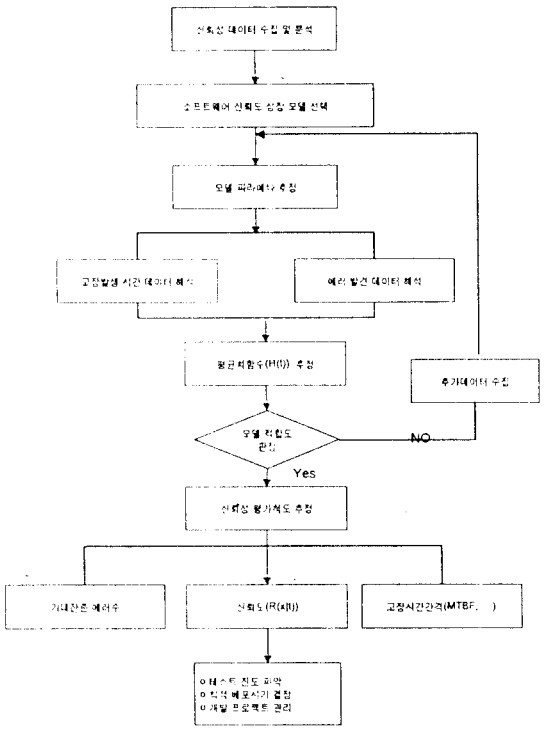
II. 소프트웨어 신뢰도 성장 모델(8)

1. 신뢰성 데이터 해석 방법

소프트웨어 신뢰도 성장 모델에 근거를 둔 신뢰성 데이터 해석 방법은 평균치 함수 H(t)에 대해 테스트 단계에서 관측된 데이터와 실측된 데이터를 적용하는 일련의 순서로 다음과 같다. 신뢰성 데이터를 수집, 분석한 후 모델을 선택하여 모델 파라메터를 추정하고 소프트웨어의 고장발생 시간 데이터 및 에러 발견 데이터를 해석하여 이를 토대로 평균치 함수를 추정한다. 평균치 함수가 구해지고 나면 모델의 적합도 판정을 거쳐 신뢰성 평가 척도를 추정, 소프트웨어 내의 기대 잔존 에러수 및 신뢰도, 고장시간간격(MTBF : Mean time between Software Failures) 등을 추정하여 소프트웨어 배포시기, 테스트 진행상황 파악, 소프트웨어 개발 프로젝트 관리등에 활용한다. 이에 대한 전반적인 처리 절차는 아래와 같다.

2. 모델의 선택

소프트웨어 개발은 초기 설계 단계부터 실현 단계를 거쳐 운용되기 전까지 모든 과정이 인위적인 작업 과정을 거치므로 소프트웨어 내에 에러가 삽입되는 것은 피할 수 없다. 그래서 소프트웨어 개발 최종 단계인 테스트 단계에서는 개발 순기(Cycle)에 따라 소



신뢰성 데이터 해석 방법

프트웨어 내에 잠재하고 있는 에러를 발견, 수정하는 작업을 반복하게 된다. 이와 같이 에러 발견 사상이나 소프트웨어 고장 발생 과정을 기술하여 테스트 단계에서의 소프트웨어 신뢰성 평가를 위한 고장 데이터 계수 과정을 거친다. 즉, 시험 시각 t 마다 발견된 총 고장 데이터를 수집, 분석하여 누적 시키는 과정을 말한다.

수집된 신뢰성 데이터를 분석하고 나면 이를 모델에 적용하게 되는데 이때 모델의 선택이 매우 중요하다. 즉 소프트웨어 개발 과정의 투명성을 보장할 수 있는 모델로 데이터의 수집이 용이하고 객관적인 데이터로 잔존 결함의 추정이 가능한 모델이 좋다. 왜냐하면 소프트웨어 신뢰성 평가 모델의 주된 용도는 개발 도중이나 개발 종료시점에서의 소프트웨어 내에 존재하고 있는 잔존 결함의 산출이기 때문이다. 다시 말해서 프로젝트 요원의 구성이나 개발 담당자의 경력, 검출 가능한 결함의 영향도를 파라메터로 하는 모델은 실제 적용과 검증이 어렵다.

고장 데이터 해석 시 적합한 모델의 선택 및 실시

한 시험 환경의 요인을 충분히 고려하여야 하는데 그 주요 고려사항은 다음과 같다.

- o 예측 타당성(Predictive validity)
  과거 데이터로부터 향후 소프트웨어 고장발생 현상이나 에러 발견사상을 예측하는 능력
- o 유효성(usefulness)
  소프트웨어 개발 프로젝트의 계획이나 관리에 있어서 소프트웨어 관리자나 개발자에 필요한 관리 지표의 추정 능력
- o 가정의 질적 수준(quality of assumption)
  실측 데이터에 대한 모델의 충실도
- o 적용성(applicability)
  타 소프트웨어의 개발환경을 고려하는 경우 등 적용범위를 확대할 필요성 고려
- o 간결성(Simplicity)
  데이터 수집, 이론의 이해, 제반 툴 화하는 용이성

등이 실제 테스트 과정에서 고려되어야 한다. 이러한 관점에서의 부분적인 연구결과 등은 이미 발표된 바 있다.[3]

3. 대표적인 신뢰도 성장 모델

시험이 진행됨에 따라 소프트웨어 내에 잠재하고 있던 에러가 발견, 수정(correction)되면서 소프트웨어의 신뢰도가 성장되어 가는 과정을 모델로 채택한 것을 신뢰도 성장 모델이라고 부르며, 이 모델은 시간 계측 모델과 개수 계측 모델로 나뉜다. 매 시험마다 얻어지는 고장 데이터를 쉽게 적용할 수 있는 개수 계측 모델의 대표적인 모델로는 NHPP(Non-Homogeneous Poisson Process) 모델을 꼽을 수 있다. 또 NHPP 모델은 지수형 신뢰도 성장 모델(ESRGM : Exponential Software Reliability growth Model)과 S-자형 신뢰도 성장모델(SSRGM : S-Shaped Software Reliability growth Model)로 나눌 수 있다.

3.1 지수형 신뢰도 성장 모델

이 모델은 "단위 시간당 결함 검출 수는 잔존 결함수에 비례한다."는 개념에서 출발한 모델로 기대 잔존 에러수 m(t)는

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

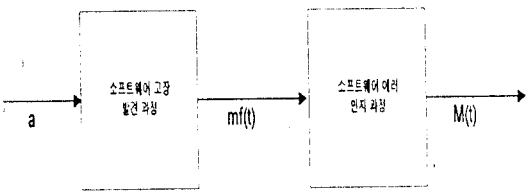
로 표현된다.

여기서 a는 최종 검출되는 기대 결함 수, b는 결함 발견율이다.

지수형 성장 모델은 Jelinski-Moranda(J-M)의 Hazard rate에 의한 모델을 개선한 것으로 고장발생시간 간격은 서로 독립적이며 소프트웨어 내에 잠재하고 있는 초기 에러는 유한하지 않다고 가정한 모델이다.

3.2 S-자형 신뢰도 성장 모델

소프트웨어 에러 발견 사상이나 고장발생 현상의 의미를 구하는데 용이한 모델로써 시험이 진행됨에 따라 에러 발견 용이성이 고려된 소프트웨어 신뢰도 성장 모델이다. 프로젝트를 수행해 오는 과정의 소프트웨어 개발관리 방법 즉, 소프트웨어 고장관리 방법과 유사한 모델로 개발된 시스템의 소프트웨어 신뢰도 평가에 적합한 모델이다.



지연 S-자형 소프트웨어
신뢰도 성장 모델

a : 시험시작 전 소프트웨어 내 잠재하고 있는 총 기대 에러 수
mf(t) : 시험시각 t 마다 발견되는 총 기대 고장 수
M(t) : 시험시각 t 마다 인지, 발견되는 총 기대 에러 수
$M(t) = a(1-(1+bt)e^{-bt})$, $(a, b>0)$, $(b = b1 + b2)$
b는 결함 발견율, b1 : 고장발견과정의 발견율, b2는 고장 인지과정의 발견율

고장의 발견 즉시 해결된다고 가정하는 G-O(Goel-Okumoto) 모델은 소프트웨어 개발 초기 단계에 많은 에러가 발견, 수정되기 때문에 계수(計數)가 용이하고 실무에 간단히 적용될 수 있기 때문에 초기 개발에 적용이 가능한 모델인 반면에 고장 발견 과정과 인지 과정에 대한 추적이 용이한 S-Shaped 모델은 개발 최종 단계인 테스트 단계에 적용하기에 적합한 모델이다.

4. 소프트웨어 신뢰도 성장 곡선

소프트웨어 신뢰도 성장 곡선은 검출한 소프트웨어 결함의 누적치를 경과 시간에 따라 그린 것으로 검출 결함 데이터로부터 잔존 결함수와 이 결함들을 검출하는데 걸리는 시간을 추정하는 지표가 된다.

소프트웨어 신뢰성 평가에 있어서 잔존 결함수 추정은 소프트웨어 개발 관리상의 중요한 항목으로 신뢰도 성장 곡선 형성의 물리적 해석을 근본으로 하는 모델과 실제 성장 곡선과의 일치를 목표로 하는 모델로 구분된다. 전자는 Goel-Okumoto가 제안한 지수형 성장 모델, Yamada가 제안한 S-자형 성장 모델, ohba - 梶川이 제안한 習熟 S-자형 성장 모델 등이 있으며, 후자는 콤펠츠(Gompertz) 곡선 모델, logistic 곡선 모델, Karunanithi가 제안한 Nural Network 모델 등이 있다.

시간계측모델이나 개수 계측 모델이 시험단계나 운용단계에서 에러 발견 사상이나 고장 발생 현상에 대한 물리적인 의미를 구하는 반면 콤펠츠 곡선모델이나 로지스틱 곡선 모델 등 경향곡선 모델은 각 곡선의 收束値를 소프트웨어내에 잠재하고 있는 총 에러수에 대해 회귀분석을 통해 추정하는 모델이다.[10]

III. 성장 모델의 기본 이론 및 평가 척도

1. 모델의 기본 개념

시스템 시험마다 발견된 소프트웨어 누적 에러를 표현하는 계수 과정을 {N(t), t > 0}라고 가정하면 이것에 대한 평균치 함수를 H(t)로 하여 소프트웨어 신뢰도 모델을 표현하면 식(1)과 같이 나타낼 수 있다.

$$P_r[N(t)=n]=\frac{[H(t)]^n e^{-H(t)}}{n!} \quad (n=0,1,2,3,\ldots) \quad (1)$$

이때 테스트에 의해 최종 발견되는 총 기대 에러수 즉, 시험 개시 전 소프트웨어 내에 잠재하고 있는 총 기대 에러수를 파라메타 a(= H(∞))로 나타낼 수 있으며, 이때 N(t)의 분포는

$$\lim_{t\to\infty} P_r[n(t)=n]=\frac{a^n e^{-a}}{n!} \quad (n=0,1,2,3,\ldots) \quad (2)$$

로 된다.

식 (2)는 시간이 무한한 경우의 시험을 수행할 때 N(t)는 평균치 a의 Poisson 분포에 따른다. 일반적으로 소프트웨어 신뢰도 성장은 테스트 시간과 그때까지 관측된 누적 에러수와의 관계를 나타낸 것을 의미한다. 다시 말해서 테스트 진행에 따른 총 기대 에러의 시간적 경향 및 변화에 의해 결정되는 곡선으로 관측된 누적 발견 에러가 지수형 성장 곡선 또는 S-자형 성장 곡선으로 나타내도록 에러 발견 사상을 기술한 모델이다.

2. 신뢰성 평가 척도

소프트웨어 신뢰성 평가에 유용한 정량적인 척도는 소프트웨어 내에 잠재하고 있는 에러수와 소프트웨어 신뢰도이다. 임의의 테스트 시각 t에서의 소프트웨어 내 에러수는 $N(t) = N(\infty) - N(t)$, 로 정의한다. 이때 N(t)의 기대치 및 분산은 $n(t) = E\{N(t)\} = a - H(t)$ = Var $\{N(t)\}$, 로 주어지며 테스트 시각 t까지의 발견된 에러수를 n이라고 하면 이때 N(t)의 조건부 분포는

$$N(t) = n$$

일 때

$$P_r[n(t)=x \mid n(t)=n]=\frac{n(t)^x e^{n(t)}}{x!} \quad (x=0,1,2,3,\ldots)$$

$$H(t)=\int_0^t h(x)\,dx$$

h(x) : 단위시간당 발견되는 결함수(순간에러 발견율)

로 되어 이 기대치는 $E\{N(t)\mid N(t) = n\} = n(t)$로 주어진다. 테스트 시각 t에서의 소프트웨어 내에 잔존하는 에러수의 분포는 평균치 n(t)의 n에 무관하게 Poisson 분포를 따르는 것을 알 수 있다.

소프트웨어 고장 발생 간격은 확률 변수 X_k (k = 1, 2, 3, ...)에 의해 정의되며, 확률 변수

$S_k = \sum_{i=1}^{k} X_i$는

k 번째 소프트웨어 고장 발생 시각을 나타낸다. k-1번째 소프트웨어 고장 발생 시각 $S_{k-1} = t$가 주어지면 X_k의 조건부 신뢰도 함수는

$$R(x \mid t) = P_r[X_k > x \mid S_{k-1} = t]$$

를 얻는다. 즉, 시험구간 (t, t+x)에서 고장이 발생하지 않을 확률을 의미한다. 평균치 함수 H(t)에 지수형 또는 S-자형 평균치 함수 식을 대입하면 각 모델의 잔존에러수 n(t)와 신뢰도 R(x|t)를 구할 수 있다. 즉,

$$n(t) = a - H(t) = Var[N(t)],$$
$$R(x \mid t) = e^{[-H(t+x)-H(t)]}$$

로 표현된다.

ㅇ 모델별 평균치 함수식

$$H(t) = m(t) = a(1-e^{-bt}), \ (a, b>0) \qquad \text{지수형}$$
$$H(t) = M(t) = a[1-(1+bt)e^{-bt}], \ (a, b>0) \quad S\text{자형}$$

3. 모델 파라메타 추정

모델 파라메타 추정은 소프트웨어 개발 과정의 테스트 단계에서 검출된 데이터를 가지고 최대 가능법(MLE : Maximum-likelihood method estimation)에 의해 가능하다. 즉 시험 단계에서 n 번(혹은 set)의 고장 데이터를 가지고 추정한다고 가정하면

$(t_i, S_i) \quad i = 1, 2, 3, ..., n, \quad (0 < t_1 < t_2 < ..... t_n)$

테스트 t_i까지 관측된 고장 데이터를 S_i라고 하면 평균치 함수 M(t)를 가진 NHPP 모델의 미지 파라메타에 대한 도함수 L은

$$L = \prod_{k=1}^{n} P_r[N(t_k) - N(t_{k-1}) = S_k - S_{k-1}]$$
$$= \prod_{k=1}^{n} \frac{M(t_k) - M(t_{k-1})^{S_k - S_{k-1}} e^{[M(t_k) - M(t_{k-1})]}}{S_k - S_{k-1}}$$

를 얻는다. 양변에 자연 대수를 취해 정리한 후 이를 편미분하면 모델의 미지 파라메타를 a, b를 얻을 수 있다.

IV. 실제 데이터 적용 예

각 버전별 시스템 시험마다 검출된 고장데이터는 표 1과 같다. 테이블에서 알 수 있듯이 시험 초기 단계에서 많은 에러가 발견되어 고쳐지고 있으며, 점차 시험이 진행됨에 따라 에러가 감소하는 경향을 그림 1, 2로 알 수 있다. 소프트웨어 신뢰성 평가에 적용된 소프트웨어 버전은 개발 초기 버전인 N3.3, 새로운 기능이 추가된 N3.4 및 현장 적용 버전인 N3.5로 구분된다.

ㅇ 대상 소프트웨어 규모 및 역할

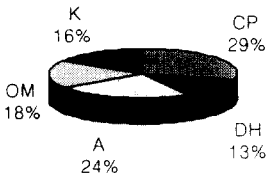
본 연구의 대상이 된 교환 소프트웨어의 규모는 134만 소스코드 라인을 갖는 대형 프로그램이며 140개의 주요 기능 블록으로 구성되어 있다. 이것은 다시 OS & DBMS 등 시스템 Kernel 부분(K)과 운용(A) 및 보전(OM) 기능, Call Processing(CP), 데이터 처리 기능 등으로 구분되며 이에 대한 전체 현황은 파이 차트와 같다.

표 1. 소프트웨어 버전별 고장 데이터
Table 1. No. of Failure data per Software Versions

시간 (월)	No. of failure (N3.3)	No. of failure (N3.4)	No. of failure (N3.5)
1	83	23	1
2	287	21	4
3	177	81	1
4	193	24	0
5	120	22	3
6	67	12	4
7	75	2	3
8	46	1	2
9	24	0	3
10	69	11	1
11	129	16	1
12	117	30	2
13	31	2	0
14	40	0	0
15	34	1	0
16	35	0(*)	0
17	20	2	1
18	5	0	0

* : 1차 배포시기

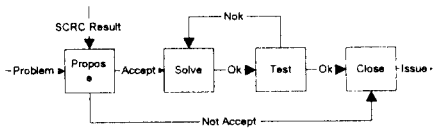
소프트웨어 기능별 규모 현황



A : Administration, CP : Call processing
OM : Maintenance, DH : Data handling
K : System Kernel

ㅇ고장 데이터 및 소프트웨어 관리

시스템 시험마다 검출된 모든 고장 보고는 소프트웨어 형상관리 위원회(SCRC : Software Change Request Committee)의 검토를 거쳐 고장관리시스템(FMS : Fault Management System)에 버전별로 등록된다. 이를 다시 하드웨어, 소프트웨어로 구분하고 소프트웨어의 수정이 필요한 고장은 소프트웨어 변경이력 관리 시스템(CRMS : Software Change Request Management System)으로 입력되어 등록된 모든 고장이 수정, 해결 완료될 때까지 추적 관리된다. 이와 같은 처리 과정은 아래와 같은 절차에 따라 수행되며, 일부 고장 데이터는 고장이 아닌 것으로 판정되어 제안과 동시에 종결처리 되었다. 이것들에 대한 비율은 총 고장 보고 1980건 중 약 10% 정도를 차지하였고 보고된 고장 중에서 고장으로 인지되어 CRMS에 등록되어 처리되고 있는 데이터는 약 1700 개 정도다. 또 각각의 세부 처리 절차는 History 파일에 저장되어 관리된다. 전반적인 소프트웨어 변경 요구에 대한 세부 단계별 처리 절차 및 내역은 생략한다.[4]



CRMS Flow Diagram

고장 데이터에 대한 버전 별 소프트웨어 블록 분포 비율 및 세부 해결 현황은 그림 1, 2와 같으며, 기능별 고장 해결 현황은 그림 3과 같다. 해결 현황을 보면 인접 기능 블록과의 연동이 많은 운용/보전 및 호처리 기능에 많은 고장이 발생하고 또 고쳐지고 있음을 그림 3을 통해 알 수 있다. 이러한 원인은 교환시스템의 주요 역할이 각종 호 처리와 주기적인 시스템 상태 관리, 감사기능(Audit Function) 등이기 때문에 이에 대한 고장은 발견 즉시 해결되어야 실시간 서비스를 제공할 수 있다.

소프트웨어 전체 규모면으로 볼 때 이 기능들이 차지하는 비율은 70% 정도이다.

Figure 1 Fault density distribution among Software Blocks

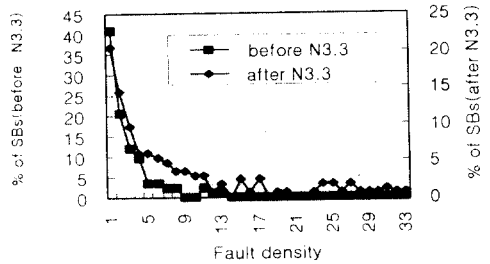


그림 1. 소프트웨어 블록내 고장발생 분포
Fig. 1 Fault density distribution among Software Blocks

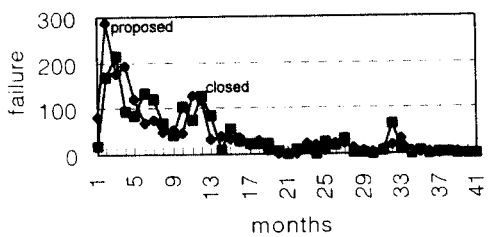


그림 2. 소프트웨어 변경 요구 및 해결 현황
Fig. 2 Evolution of the number of proposed SCRs and closed SCRs

그림 2에서 보는 바와 같이 시험 시작 10개월 후 2개월간과 30개월 후에 많은 고장이 발견되고 수정되는 경향을 보이는데 그 이유는 실용화 및 상용화를 위해 현장 적응을 위한 시도로 교환시스템의 운용 프로파일의 변화에 따라 많은 고장이 검출되고 그 시기에 수정되는 현상을 보였다.

그림 3은 주요 기능별 소프트웨어 결함 수정 현황으로 단위시간(월)별 누적 추이를 그린 것이다. 고장의 형태를 보면 운용, 보전, 호처리 순으로 결함이 많

이 발생하고 있으며, 이는 교환 시스템의 특징을 잘 말해주고 있다.

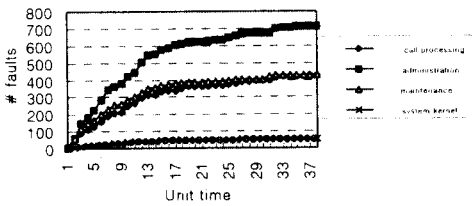
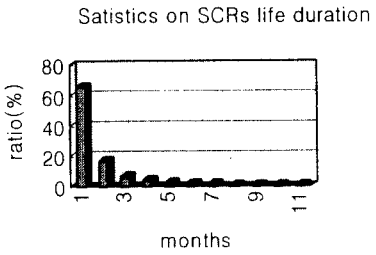


그림 3. 소프트웨어 기능별 결함 수정 누적수
Fig. 3 Cumulated number of corrected faults in Software functions

또한 소프트웨어 변경(SCR: Software Change Request)에 대한 생명 주기(Life-Cycle) 즉, 변경 요구와 종료 시점 사이의 관계를 표시한 것으로 대부분이 1개월 이내에 수정되었으나 전체의 0.5% 정도는 10개월 이상 걸린 것으로 분석되었다(아래 그림 참조).



o G-O Model에 의한 적합도 판정 예

시스템 시험 단계에서 고장 발견 사상을 소프트웨어 신뢰도 성장모델을 이용하여 기술하고 신뢰성을 평가하는 일은 매우 관심 있는 일이다. 다시 말해서 소프트웨어 신뢰성 평가에 있어서 개발관리자에 의해서 경험과 직관에 의해 평가되는 것은 애매하고 어렵다. 그래서 적절한 모델을 선택하기 위해 실측 된 데이터에 대한 모델 적합성 평가가 필요하다.

아래 표는 시스템 시험에서 검출된 실측 데이터 이용하여 각 버전별로 NHPP 모델에 대한 적합도 판정을 수행한 결과는 아래와 같다.

[N3.3]

Parameter	Estimate	Asymptotic std. Error	Asymptotic 95% Confidence interval	
			Lower	Upper
B0	1460.502008	9.8586713953	1437.1897676	1483.8142489
B1	0.160962	0.0018738456	0.1565315	0.1653934
R-Square = 0.970937 R^2 = 1-SSE/CSE, R^2: 비선형 모델의 통계량 결정계수 SSE: the variance of the full model, CSE: the variance of the mean model				

[N3.4]

Parameter	Estimate	Asymptotic std. Error	Asymptotic 95% Confidence interval	
			Lower	Upper
B0	559.2257800	5.2404657926	546.83394804	571.61761201
B1	0.2340925	0.0046798135	0.22302632	0.24515854
R-Square = 0.976044 R^2 = 1-SSE/CSE, R^2: 비선형 모델의 통계량 결정계수 SSE: the variance of the full model, CSE: the variance of the mean model				

[N3.5]

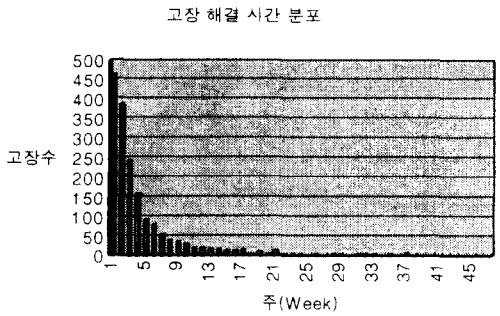
Parameter	Estimate	Asymptotic std. Error	Asymptotic 95% Confidence interval	
			Lower	Upper
B0	235.1020588	6.0394739937	220.82085745	249.38326010
B1	0.2144906	0.0111351381	0.18815992	0.24082118
R-Square = 0.906452 R^2 = 1-SSE/CSE, R^2: 비선형 모델의 통계량 결정계수 SSE: the variance of the full model, CSE: the variance of the mean model				

[Total]

Parameter	Estimate	Asymptotic std. Error	Asymptotic 95% Confidence interval	
			Lower	Upper
B0	1856.422535	2.5687703069	1848.2474237	1864.5976467
B1	0.881858	0.0045545132	0.8673628	0.8963523
R-Square = 0.997977 R^2 = 1-SSE/CSE, R^2: 비선형 모델의 통계량 결정계수 SSE: the variance of the full model, CSE: the variance of the mean model				

각 버전별로 계산된 실제 결정계수(R^2)의 값은 0과 1사이의 값을 가지며 1에 가까운 값을 가질 때 좋은 모형을 가진다. 특히 버전을 전체 통합한 데이터에 대한 결정 계수 값이 가장 좋은 결과로 계산되었는데 이 의미는 소프트웨어 개발 프로젝트 과정 중 시스템 시험단계에서 적용된 개발방법이 G-O Model에 가장 적합하다는 것을 의미한다. 그러나 개발 과정의 내용을 세밀히 관찰한 결과 각 버전별로 추진한 내용은

시험에서 검출된 문제점들이 일부 즉시 고쳐지지 않고 일정시간이 경과된 후에 해결되고 있음을 그림(고장해결 시간 분포)을 통해 알 수 있다.



위의 모델에서 시험시각 t마다 검출되는 평균 누적 고장 수는 아래의 식 (3)으로 표현된다.

$$\mu(t) = a(1 - e^{bt}) \tag{3}$$

여기에서, a, b는 모델 파라미터 들로서 a는 시간이 무한으로 이를 때의 전체 고장 수이고, b는 고장당 고장 발견 율(the fault detection rate per fault)이다. 이때 시간에 따른 고장 강도 함수는

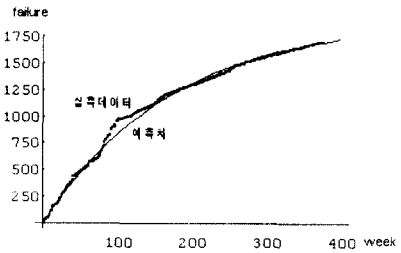
$$\lambda(t) = abe^{bt} \tag{4}$$

식 (4)로 표현되고 모델 파라미터 a, b값을 구한 후 예측된 전체 누적 고장수에 대한 추정 결과는 그림과 같다(전체 누적 고장수 참조). 전체 소프트웨어에 대한 소프트웨어 고장율에 적용된 파라미터 값은 a = 1916.87, b = 0.0057145로 계산되었다. 또한 버전별로 구한 고장율은 표 2와 같이 추정되었고 현장 배포 버전인 N3.4의 평균 순간 고장율은 b = 0.0374로 추정되었다.

ㅇ 소프트웨어 배포시기 결정 문제

앞에 언급한 Table 1의 버전별 소프트웨어 고장 데이터를 적용하여 평가된 운용시간 x 및 시험 시간 t (per week)에 관한 소프트웨어 신뢰도 [R(x|t)] 와의 관계는 그림 4, 5와 같다. 그림 4에서 현장 배포 버전인 N3.4의 시험 초기인 t = 80 및 시험 중간 단계인 t = 100일때를 살펴보면 소프트웨어가 현저히 안정되어

전체 누적 고장 수(Cumulative Failure)



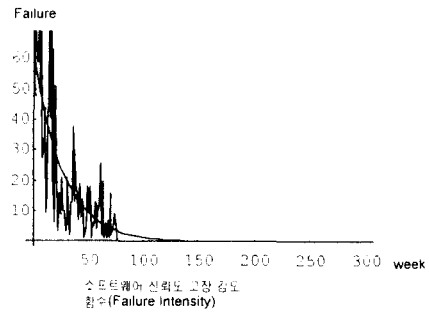


표 2. G-O Model을 적용한 소프트웨어 고장 율
Table 2. software failure rate of G-O Model

Software Version	Failure Rate(/hour)
N3.3	0.223×10^{-4}
N3.4	0.495×10^{-4}
Total	0.340×10^{-4}

감을 알 수 있다. 소프트웨어 1차 현장 배포 단계인 t = 120 주 전후에는 신뢰도 목표치(95% 이상)에 도달하고 있음을 알 수 있다.

현장 배포 단계인 t = 120 일때 소프트웨어내의 미발견 에러에 대한 기대 잔존 에러수 [n(t)] 를 추정해 보면

$$n(t) = a * e^{-bt}$$

즉,

$$n(t = 120) = 1647 * e^{-(0.0374 * 120)} = 18.36$$

개로 추정되었다. 이 에러들은 운용상에 크게 지장이 없는 에러들로 판정되었다. 이러한 에러 즉, 보이지 않는 에러(Latent Fault)는 현장배포 이후 지속적으로 운용국에서 감시, 보고되어 일부 고쳐졌다. 그림 5는 운용시간에 따른 소프트웨어 신뢰도 추정치를 나타낸다. 즉 소프트웨어를 배포(Release)하기 위한 최적 시기를 결정하는 문제로 시스템 시험 시작 후 약 120주 근처에서 현장 배포 수준인 95% 이상의 신뢰도를 갖는 것으로 평가되어 추정된 결과와 실제 소프트웨어를 현장에 배포한 시기인 30~35개월 후(Table 1 참조)와 거의 일치함을 알 수 있다.

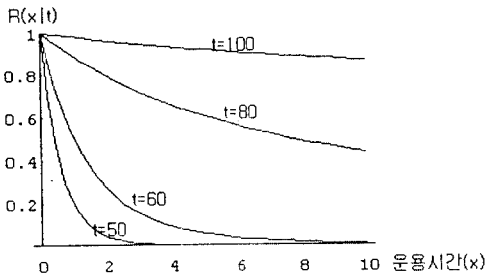


그림 4. 소프트웨어 신뢰도[R(x|t)]와 시험시각 t의 관계
Fig. 4 Software Reliability[R(x|t)] vs Test time t

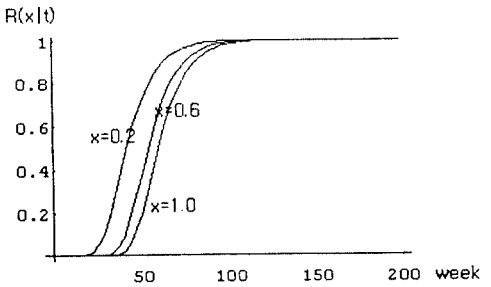


그림 5. 소프트웨어 신뢰도와 운용시간 x와의 관계
Fig. 5 Software Reliability vs Operation time x

o S-자형 모델 적용 결과

S-자형 모델 적용에 따른 N3.3 버전과 N3.4에 대한 소프트웨어 신뢰도 평가치는 표 3과 같으며, 고장인지 과정과 처리 과정을 고려한 누적 고장수 및 고장 밀도에 대한 결과는 아래 그림 6과 같다. (점선으로 표시된 곡선은 실측 데이터임)

S-자형 모델은 지연 S-자형과 시험팀의 소프트웨어에 대한 숙련도를 고려한 習熟 S-자형 모델, 테스트 능력(자원 및 인력 투입량)을 고려한 시험능력 의존형 S-자형 모델로 구분되나 본 논문에서는 S-Shaped 신뢰도 성장 모델만 취급하기로 한다. 시험시각 t 마다 발견될 총 에러수 M(t)와 고장 강도 함수 $h_M(t)$는 아래 식 (5), (6)과 같이 표현된다.

$$M(t) = a[1-(1+bt)e^{-bt}], \quad (a,\ b>0) \tag{5}$$

$$h_M(t) = ab^2 te^{-bt} \tag{6}$$

a : 시험 시작 전 소프트웨어 내에 잠재하고 있는 총 기대 에러 수

b : 1개당 소프트웨어 고장 발견 율(b1) 및 에러 인지율(b2), $(0<b2<b1<1)$

전체 고장 데이터를 적용한 결과는 a = 1559.14, b = 0.087547로 평가되어 G-O 모델보다는 순간 발생 고장율이 높은 것으로 추정되었다. 이 결과는 각 모델의 가정에 따른 차이로 밝혀졌다. 이와같은 현상은 소프트웨어 고장에 대한 비중(weight)과 고장 해결에 소요된 시간, 시험시 보고된 고장 보고에 대한 고장의 인지 여부 등 각 요소에 대한 평가치의 차이 때문인 것으로 인식된다. 이러한 제반 요소(factor)를 반영한 신뢰도 모델의 적용은 너무 복잡하기 때문에 본 논문에서는 생략한다.

시험 진행에 따라 소프트웨어 에러 발견의 난이성을 고려한 모델도 제안[7][8] 되었으나 실제 실무에 응용하는 데는 다소간 어려움이 뒤따른다. 그러나 우리가

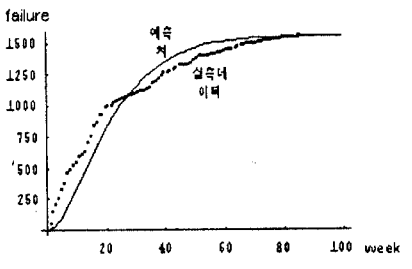
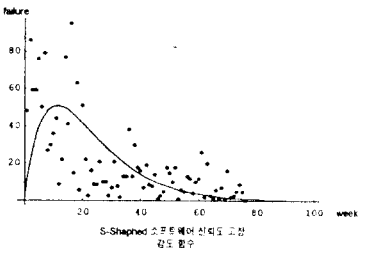


그림 6. S-자형 전체 누적 고장 수
Fig. 6 Cumulative Failure of S-Shaped

표 3. 지연 S-자형 모델을 적용한 소프트웨어 고장율
Table 3. Software Failure rate of delayed S-Shaped Model

Version Name	Failure Rate(per hour)
N3.3	0.223×10^{-3}
N3.4	0.495×10^{-4}
Total	0.340×10^{-4}



S-Shapned 소프트웨어 신뢰도 고장 함수

적용한 고장데이터는 해석 단계에서 이를 충분히 고려(시스템 시험 단계에서 고장의 등급을 고려하여 시험하고 검출된 고장 데이터를 분석, 적용 함)하였다. 근사적으로 b = b1 = b2인 경우 임의의 시간 t 마다 발견될 총 기대 소프트웨어 고장 수 mf(t)에 대해서 위의 식 (5)를 얻을 수 있다.

위의 2개 모델을 적용하여 예측한 소프트웨어 신뢰도 평가 결과는 모델의 특성에 따라 다르게 평가되었는데, 이 결과는 시험시 검출된 고장이 즉시 해결된다는 가정을 가지고 평가되는 G-O Model의 특성과 고장 발견부터 해결될 때까지의 일정한 시간이 요하는 현실을 감안한 S-자형 모델의 특성 차이이기 때문이다.

실제로 시스템 개발 과정에서 검출되는 고장들은 즉시 해결되는 고장과 일부는 즉시 고쳐지지 않고 수일 또는 수개월에 걸쳐 해결되는 고장이 있기 때문이다. 전체적으로 위의 2 모델을 적용한 결과를 비추어 볼 때 시간이 지남에 따라 소프트웨어가 안정되어가는 추세를 알 수 있고 국부적(버전별)으로 볼 때 일반적인 S-자형 모델을 따르며, 개발 전 기간을 총체적으로 살펴보면 지수형 성장모델을 따르는 것으로 밝혀졌다. 이러한 결과들은 이미 연구 조사되어 발표된 바 있다.[4, 5, 10]

결론적으로 시스템 시험 단계에서 검출된 고장데이터를 가지고 평가된 결과로서 적합한 소프트웨어 신뢰도 모델은 G-O 모델보다는 S-자형 모델이 더 현실적임을 알 수 있다. 그러나 전반적인 프로젝트 관리 측면에서의 소프트웨어 신뢰도 예측 기법으로 G-O 모델이 많이 적용되고 있다. 정확한 신뢰도 평가를 위해서는 다각적인 측면에서의 각 모델에 적용될 파라메터에 대한 더욱 연구가 필요하다.

V. 결 론

본 논문은 시스템 시험으로부터 얻어진 고장 데이터를 다각도로 분석하고 소프트웨어 신뢰도 관점에서 개발 과정을 살펴보았다. 또 우리가 행한 시험 환경 및 개발 방법을 개선하여 우리 실정에 맞는 정확한 소프트웨어 신뢰도 평가 모델을 선정하여 신뢰도를 평가해 봄으로써 개발된 교환 소프트웨어는 물론 차세대 교환 시스템 개발 시 소프트웨어 신뢰도를 좀더 정확히 평가할 수 있는 방안을 마련하였다. 그밖에 시스템 시험 기간 동안 발견되지 않은 결함(Latent Fault)이 현장 적용 기간 중에 발생되었으나 이러한 결함들은 개발 팀으로 feed-back 되어 원인 분석(Cause Analysis)과 고장 위치 분석(Semantic Analysis)을 통해 수정 되었다.

종래의 소프트웨어 개발 방법이 주로 개발 방법론에 치우친 반면에 최근에는 개발된 소프트웨어의 품질에 대한 관심이 고조됨에 따라서 소프트웨어 신뢰도 평가를 예측하기 위한 방법이 부분적으로 시도되어 왔으나 미미한 상태였다. 이러한 실정에 맞추어 좀더 다각적인 측면에서 다양한 평가 방법의 연구 및 적용이 이루어져야 겠다.

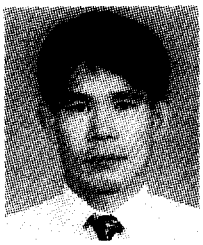
향후 연구 과제로는 고장 영향의 등급 배분을 고려한 소프트웨어 수정 규모(Corrected software size), 소프트웨어 블록(block) 규모, 고장수등 각종 소프트웨어 컴포넌트(component)들의 변경을 고려한 종합적인 소프트웨어 신뢰도 평가 방법 연구가 필요하다.

참 고 문 헌

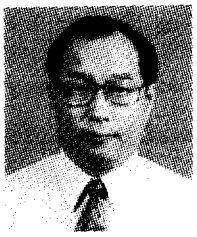
1. Mohamed Kaaniche and Karama Kanoun, "Reliability of a Commercial Telecommunications System", IEEE Trans. Software Engineering, pp. 207-211, 1996.
2. Michael R. Lyu, "Handbook of Software Reliability

Engineering", McGraw-Hill, 1995.
3. J. Musa, A. Lannino, and K. Okumoto, "Software Reliability : Measurement, Prediction, Application", McGraw-Hill, 1987.
4. 이재기외, "대형 교환기의 시스템 신뢰도 평가", '95통신학회 추계학술대회 논문집, pp. 537-540, 1995.
5. 유재년, 이재기, "교환 시스템 고장 유형 분석과 신뢰도 평가", '94 전자공학회 추계학술대회 논문집, Vol. I, pp. 495-497, 1994.
6. S. Yamada and S. Osaki, "Nonhomogeneous error detection rate Models for Software Reliability growth", in Stochastic Models in Reliability Theory, pp. 120-143, Springer-Verlag, Berlin, 1984.
7. Yamada et al, "A Software Reliability growth Model of two types of errors", R.A.I.R.O. Operations Research, Vol. 19, No. 1, pp 87-104, Feb. 1985.
8. Shigeru Yamada, Hiroshi Ohtera, "Software Reliability : Theory and Practical Application", SE series, pp. 135-196. Soft Research Center, Feb. 1990.
9. Shigeru Yamada, "소프트웨어 신뢰성 평가 기술", Integrated Libraries No. 42, pp. 79-120, HBJ Publishing, 1989.
10. 유재년, 이재기, "기능 블럭으로 구성된 대형 교환 소프트웨어의 신뢰도 성장", 대한전자공학회 논문지 제35권 2호, 1998. 1.
11. Mitsuhiro KIMURA, Shigeru Yamada, Shunji OSAKI, "A note on Software Reliability Evaluation during operation phase considering Testing-effort expenditures", 전자정보통신학회논문지, Vol. J 72-D-I No. 12, Japan, 1989.

이 재 기(Jaeki Lee) 정회원
1985년 2월 : 서울산업대학교 전자공학과 졸업(공학사).
1989년 5월 : 청주대학교 대학원 전자공학과 졸업(공학석사).
1983년 3월~현재 : 한국전자통신연구원 교환·전송연구소, 교환서비스연구부, S/W종합검증팀 선임연구원 재직중.
※주관심분야 : Software Testing, Software Quality
e-mail : jkilee@nice.etri.re.kr



신 상 권(Sangkwon Shin) 정회원
1985년 2월 : 중경공업전문대학 전자과 졸업
1988년 2월~현재 : 한국전자통신연구원 교환·전송연구소, 교환서비스연구부, S/W종합검증팀 재직중.
※주관심분야 : Software Integration & Test



이 영 목(Youngmock Lee) 평생회원
1974년 2월 : 성균관대학교 전자공학과 졸업(공학사)
1976년 8월 : 성균관대학교 대학원 전자공학과 졸업(공학석사)
1980년 8월~1982년 7월 : 미국 사우스캐롤라이나 주립대학 대학원 전기·컴퓨터공학과 수료
1984년 3월~현재 : 한국전자통신연구원 교환·전송연구소, 교환서비스 연구부, 소프트웨어 종합검증팀 선임연구원 재직중
※주관심분야 : 소프트웨어 개발방법론