論文99-36S-3-4

선험적 고장 데이터에 의한 TDX 계열 교환 소프트웨어의 결함 검출율 분석

(An Examination of Fault Exposure Rate of Switching Software of TDX Series from Empirical failure data)

李 在 起 *, 辛 相 權 *, 洪 性 伯 *

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

요 약

소프트웨어의 결함 검출율(FER : Fault Exposure Ratio)은 소프트웨어에 대한 시험의 효율성과 고장 당 결함 발생율(per fault hazard rate)을 제어하는데 매우 중요한 요소이다. 특히 시험이 불규칙적으로 수행될 때 고장 발견은 더욱 어려워진다. 시험이 종료되는 단계에서 소프트웨어 결함 검출율이 낮은 경우는 시험의 유효성을 기대하기 어렵기 때문이다. 일반적으로 결함 검출율(K)이 점차 높아지는 시험 종료 단계에서는 Random Test 보다는 강도 높은 실 시험이 수행되기 때문이다. 이런 가정하에 본 논문에서는 TDX 교환 소프트웨어의 결함 검출율을 추정하여 이를 기반으로 한 ATM 소프트웨어의 결함 검출율을 예측하고 또한 소프트웨어 신뢰도가 향상되어 가는 과정에 대해 논했다.

Abstract

The software fault exposure ratio is an important factor that controls the effectiveness of testing of software and the per fault hazard rate. Others, the fault exposure ratio thus presents a key challenge in our quest towards understanding the software testing process and characterizing it analytically. Especially, the failures get harder to find at random testing. In generally, because of executed effective real testing than strictly random testing at the end of system test phase. The fault exposure ratio trends too highly. Under this hypothesis : we have estimated the software fault exposure rate of the TDX switching system. Also, we show that the software fault exposure ratio in ATM switching system and the growth of software reliability.

I. 서 론

소프트웨어 결함은 주로 시스템 개발시 설계상의 에러나 코딩시 사람의 실수에 의해 발생하며, 소프트웨어 결함들은 여러 단계의 시험을 거치면서 고장의 원인 분석을 통해 검출되고 디버깅을 통해 수정된다.

소프트웨어 내에 존재하는 결함들을 시험을 통해 발견하는 고장과의 비를 결함 검출율(K로 표시, 단위는 결함/고장)이라 한다. 이는 소프트웨어 시험의 효율성을 나타내는데 있어서 주요한 요소로 취급된다. 이것은 미리 정해진 소프트웨어 신뢰도 목표치를 가용한 자원 한도 내에서 달성하기 위해 투입되는 여러 시험에 필요한 자원의 예측과 할당에 참조되며, 신뢰도 향상을 위한 관리 차원의 의사 결정이나 개발에 필요한 시간의 추정 등에 이용된다. 특히, 대형 소프트웨어인 경우 전체적인 기능의 실현이 여러 단계에 걸쳐 진행

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

接受日字:1998年9月1日, 수정완료일:1999年1月4日

됨으로써 단계별 신뢰도 목표치를 달성하기 위해서 결함의 제거를 위한 자원의 할당은 전체 소프트웨어의 개발비와 신뢰도에 많은 영향을 미치므로 결함 검출율의 연구는 매우 중요하다.

본 논문에서는 TDX 교환 소프트웨어의 개발과 시험을 통한 결함 검출에 관한 경험에서 얻은 고장자료에 대한 소프트웨어 신뢰도 성장 이론을 적용하여 결함 검출율을 정하는데 이용되는 여러 가지 개발의 특징을 나타내는 상수 값에 대해 논하고 이 결과를 이용하여 현재 개발중인 ATM 교환 시스템의 결함 검출율을 예측하고 이를 참조하여 소프트웨어 신뢰도 향상에 대해 논한다.

논문의 구성은 서론에서 소프트웨어의 결함 검출율의 중요성에 대해 논하며, 2절에서는 결함 검출율에 대한 기본 이론을 살펴본다. 그리고 3절과 4절에서 결함 검출율과 신뢰도 모형과의 관계 및 TDX 계열 교환 소프트웨어의 결함 검출율을 평가하고 5절에서 결론과 향후 연구 방향을 제시한다.

Ⅱ. 결함 검출율(Fault Exposure Ratio)

소프트웨어 결함 검출율의 추정을 위한 시험의 과정은 아래의 두 매개변수를 사용하여 표현된다. 즉, 최종적으로 검출이 예상되는 소프트웨어 내의 결함수와 시험 초기의 고장강도가 적용된다. 소프트웨어에 내재하는 전체 결함수를 ω_0 라고 하고 발견된 고장수를 ν_0 라 하면 발견된 고장에 의해 수정되는 결함수와의 비율, B는 시험에 의한 고장 발견으로 해결되는 소프트웨어의 결함 수정에 대한 비율을 의미한다. 이를 식으로 표현하면

$$B=\frac{\omega_0}{\nu_0} \tag{1}$$

로 표현된다.

ω_0 : 소프트웨어내의 기대 잔존 결함(에러)수,

ν_0 : 초기 발견 고장수

일반적으로 B<1의 형태를 띠는데, 다시 말해서 $\omega_0<\nu_0$ 로써 발견되는 고장수보다 수정되는 결함수가 적다는 것을 의미한다. 이는 순차적으로 수행되는 프로그램의 경우 하나의 동일한 고장을 유발하는 다수의 결함이 존재하여도 프로그램이 수행될 때 첫 번째 결함에 의해 고장이 발생하기 때문이다. B<1 이므로 시험을 통해 소프트웨어 결함의 수정에 어려움을 나타낸다.

시험 초기 단계에 있어서 소프트웨어에 대한 ω_0, ν_0, B는 개발이 완료된 앞선 소프트웨어에 대한 경험적인 자료에 근거하여 추정된다. B를 추정하는데 이용되는 대표적인 방법은 인위적으로 프로그램 내에 에러를 삽입하고 이를 시험을 통해 검출되는 비율로 나타내는 결함 삽입법(Error Seeding Method or Capture Recapture Method)이 있으며, 이에 대한 연구 결과가 다수 발표되었다.[1] [7] [8]

순차적으로 수행되는 프로그램이 Branch나 Loop 없이 시험에서의 단위 시간당 수행되는 빈도수를 f 라 하면 $f*\omega_0$ 는 소위 "결함빈도" 를 말하는 것으로 결함을 포함하고 있는 기계어(Machine code)의 단위 시간당 수행되는 횟수를 의미하며, 전체 프로그램이 모든 Machine code를 오직 한번 수행하는데 소요되는 시간의 단위 시간에 대한 비이다. 따라서 시험에서 발견할 수 있는 단위 시간 동안의 고장 수의 비인 고장강도는 $f*\omega_0$ 에 비례하는 것으로 가정할 수 있다. 즉,

$$\lambda_0 = K*f*\omega_0 \tag{2}$$

λ_0 : 단위시간 당 발생되는 고장 수

이때 K를 결함 당 고장수의 비로 결함 검출율이라 부른다. 식 (1), (2)에 의해

$$K=\frac{\lambda_0}{f*\omega_0}=\frac{1}{B*f}*\frac{\lambda_0}{\nu_0} \tag{3}$$

으로 표현된다.

T 시간 동안 발견된 고장 수의 강도를 λ 라 하면 (4)식과 같이 주어진다.

$$\lambda=\int\frac{N(t)dt}{T}=\frac{K}{(1/f)}\frac{N(t)}{T} \tag{4}$$

λ : 고장 강도,

$N(t)$: 시험시각 t에서의 검출이 예상되는 결함수

○ 고장(failure)과 결함(fault)의 정의

- 고장이란 프로그램이 시스템에 이식되어 동작되어질 때 요구사항에 정의된 내용과 다르게 동작하는 것을 의미하는 것으로 사용자 관점의 판단이다.

- 결함이란 프로그램이 수행될 때 고장을 유발시키는 원인이 되는 요소(defect)로 개발자 관점의 판단이다.

Ⅲ. 결함 검출율과 신뢰도 성장 모형

소프트웨어 신뢰도 성장 모형과 결함 검출율 K와의 관계는 Musa, Malaiya 등에 의해 연구되고 있다.[2] [3] 이에 대한 개념을 간단히 살펴보면 t 시간까지의 잔존 총 결함수를 $\tilde{N(t)}$ 라 하면 t에서의 잔존 결함수의 시간 변화율은 (5)식과 같이 주어진다.

$$\frac{d\tilde{N(t)}}{dt} = -\frac{K_s}{F}\frac{\tilde{N(t)}}{T_L} \tag{5}$$

$N(t)$: 시각 t에서의 기대 잔존 결함수

식 (5)에서 K_s 는 한번의 수행에서 결함을 검출하는 비율을 나타낸다. 또한 전체 프로그램의 모든 명령이 적어도 한번 그리고 꼭 한번 수행되는데 필요한 시간의 선형적 근사 시간을 T_L 이라 하면 결함 검출의 빈도수(frequency)는

$F=\frac{T_s}{T_L}$, (T_s : the average time for single execution)와 같이 주어진다. 따라서 K_s/F 는 결함 검출율을 의미한다. 지수형 소프트웨어 신뢰도 성장에 관한 t 시간 동안의 평균 고장수 $\mu(t)$는

$$\mu(t) = a(1-e^{-bt}) \tag{6}$$

a : 소프트웨어 내에 잔존하고 있는 총 결함수

b : 잔존 에러 당 결함 발견율

임을 상기하면 식 (5)의 K_s/F 는 소프트웨어 신뢰도 성장을 결정하는 파라미터 b 이다. b가 시간에 따라 변하는 경우, 즉, K가 시간에 대한 함수로 주어진 경우에도 위의 해석은 적용된다. 또 충분히 긴 시간(T) 동안의 b(T)에 대한 자료를 소프트웨어 신뢰도 성장 모델에서는 선형 근사 방법 및 log-liner 곡선으로 근사 시킨다.

소프트웨어의 FER는 [2]에 제안된 선형 수행시간 (linear execution time) 계산법이 주로 사용되며, 이를 이용하여 구한 상수값은 소프트웨어 개발조직, 개발방법을 전체적으로 특징화하고 구별하는데 이용된다.

결함 검출율의 추정에 이용되는 상수로서 K를 계산하는데 결함 검출율의 추정에 이용되는 K는 결함 당 hazard rate를 선행 수행빈도수로 정규화 함으로서 얻는다. 선형 수행 빈도란 소프트웨어 규모와 명령어 수행율의 비이다. K의 값에 대한 선행 연구 결과로는 [Musa] 에 의해 13개의 서로 다른 소프트웨어에 대해 연구 발표된 바 있으며, 대체적으로 그 값의 범위는 1.41 x 10^{-7}에서부터 10.6 x 10^{-7}까지 변화하며 평균적으로 4.2 x 10^{-7}(failures/fault)인 것으로 발표되고 있다.

소프트웨어 신뢰도 평가 모델은 시간의 변화에 대한 N(t)의 변화율과 한번의 수행에 필요한 시간, 결함을 발견해내는 비율(K)의 곱과 같다는 가정을 이용한다. 목표한 신뢰도 기준을 만족시키는 소프트웨어를 개발하기 위해 결함의 수정 과정은 결정적인 역할을 한다. 소프트웨어 결함의 상당부분이 설계와 코딩과정에서 사람의 실수에 의해 생기는 것과 마찬가지로 시험을 계획하고 수행하기 위해서 소프트웨어 개발 관리자는 어떻게 결함의 발견과 수정이 이루어지며, 결함율이 목표치보다 낮게되는 시점을 정량적으로 알 수 있어야 한다. 이런 과정은 초기 결함수와 결함 제거의 효율성이란 매개 변수에 의해 표현된다.

○ 초기 결함수

시험이 시작되기전 소프트웨어 내에 포함되어 있는 결함수를 말한다.

○ 결함제거의 효율성

결함이 보다 빨리 발견되고 수정되면 그 만큼 소프트웨어 신뢰도 수준은 높아지게 된다. 이것은 시험 방법과 결함의 분포에 의존된다. 시험의 효율성은 시험마다 달라지지만 시험 관리자는 전체적인 경향을 보기를 원한다. 이것은 결함 제거 과정의 통계적 특성을 이용한 모델로 분석된다.

소프트웨어 신뢰도 모델에 사용되는 매개변수는 소프트웨어의 정적인 특성간의 경험적, 실험적 관계를 이용하여 추정될 수 있다. 이 의미는 Goel-Okumoto (G-O) 모델에서 가장 쉽게 해석되며, 위의 식 (6)과 같이 표현된다.

FER에 영향을 주는 프로그램의 구조나 크기, 시험의 효과성을 들 수 있으며, 많은 연구에서 FER는 프로그램의 구조나 크기에는 크게 영향을 받지 않는 것으로 보고되고 있다. 그러나 시험 방법에 따른 검출비 변화를 보여주는 실제 경험적 결과가 있었으며,[3] 결

함밀도와 FER과의 관계 및 FER과 시험 단계와의 관계를 살펴보면 다음과 같다.

○ 결함밀도와 FER 과의 관계

결함의 발견 가능도는 무작위로 선택된 시험 입력에 의해 결함이 발견될 확률로 정의되며, 발견 확률이 높은 결함은 빠른 시간안에 발견되므로 시험이 진행됨에 따라 FER은 감소한다.

○ FER과 시험 단계와의 관계

시험단계를 정하는 정량화된 측도는 단순히 시간으로만 평가할 경우 먼저의 시험 효과를 반영하지 못하므로 결함밀도를 이용한다. 이는 다른 단계에서의 시험에서 대상 프로그램을 독립된 다른 것으로 취급할 수 있기 때문이다.

Ⅳ. TDX 계열 교환 소프트웨어의 결함 검출율

이 절에서는 이미 개발 완료되어 현장에서 서비스중인 TDX 교환 소프트웨어의 고장 데이터를 기초로 결함 검출율을 구하고 이를 기반으로 개발중인 ATM 교환 소프트웨어의 결함 검출율을 추정한다. 또 소프트웨어 신뢰도가 성장되어 가는 과정에 대해 논한다.

연구대상이 된 교환 소프트웨어는 음성 교환기능에 ISDN 기본 호처리 기능을 추가한 N3.3 버전과 패킷(B/D-채널 패킷 교환 기능), 공통선 신호방식(CCS No.7) 기능을 부가한 N3.4 버전으로 소프트웨어 버전의 진도에 따른 운용 및 보전 기능이 추가, 보완된 교환 소프트웨어이다.

1. TDX 교환 소프트웨어의 결함 검출율

소프트웨어 결함 검출율은 시험 과정에 따라 변화하지만 프로그램의 구조와 규모에 대한 의존성은 적은 것으로 보고되고 있다.[3] 그러면 TDX 교환 소프트웨어의 결함 검출율을 구하기 위해 앞의 식 (3)을 적용하여 조사한다. 적용되는 수치는 평균값을 이용하였으며, 이것에 대한 세부 내용을 살펴보면 시험에서 검출된 총 고장 보고수 2186건 중 소프트웨어 고장으로 인식된 1982건(고장분석 과정에서 204건은 고장이 아닌 것으로 밝혀져 기각 처리됨)을 원인 및 위치 별 고장 분석에 의해 최종 확인된 소프트웨어 결함에 대해 수정(correction)된 건수는 1651개 였다. 즉, 교환 소프트웨어의 주요 핵심 기능인 호처리, 운용, 보전, 시스템 커널(OS, DBMS) 및 데이터 처리 기능 등 각 기능 범주별로 확인된 결함 데이터는 아래 표 1과 같다.

표 1. 기능 범주별 소프트웨어 결함 데이터
Table 1. No. of fault data per categories.

기능 범주	블럭수	규모(KSLOC)	결함수
호처리	45	390.7	431
데이터처리	14	171.9	61
운용	44	315.3	665
보전	29	240.7	434
시스템커널	8	225.5	60
총 계	140	1344.1	1651

소프트웨어의 결함 밀도(fault density)는 경험 데이터(표 1 데이터 참조)로 부터 추출된 1000 라인 당 1.2 fault 정도로[5] [9] 이 데이터를 적용하여 결함수(ω)를 구하면

$$\omega = \frac{1.2 faults}{1000 SLOC} * 1,344,100 SLOC = 1,613$$

개가 되며, 식 (1)과 검출된 고장 수에 대한 결과는 $\nu = 1982$ [5] 에 의하면

$$B = \frac{\omega}{\nu} \approx \frac{1613}{1982} \approx 0.814$$ 즉, B < 1 이 된다.

TDX 계열의 교환 소프트웨어에 대한 결함 발견율은 시험이 진행될수록 시험 초기보다 떨어진다. 이 이유는 소프트웨어 내의 결함수가 점점 줄어들어 고장의 발견이 용이치 않음을 의미하며, 점진적으로 소프트웨어의 신뢰성이 높아져 가는 일반적인 경향을 띤다. 이 현상을 말해주는 TDX 교환 소프트웨어의 버전별 고장 수 분포 및 고장 발견율은 그림 1과 표 3의 b값의 변화로 알 수 있다.

한편 기계어(Machine code) 명령수 I는 CHILL (CCITT High Level Programming Language)로 구현된 소스 라인수 I_s와 한 소스 라인에 대한 기계어 라인의 변환비 평균값($Q_x = 3.65 \cong 4$)을 적용 [표 2 예시 데이터 참조] 하면 $I \approx 1,344,100 * 4$ 가 된다. 이때 프로세서 수행 빈도(f)는 MC68030™ Processor의 명령어 처리 능력인 10 MIPS를 적용하면

$$f = \frac{r}{I} = \frac{10 MIPS}{4 * 1.34 * 10^6 MC} = 1.865 CPU\text{sec},$$

γ : CPU가 처리할 수 있는 Instruction 수,
MC : Machine Code

가 되며, 식 (2)에 의해서 교환시스템의 고장강도 λ는 $\lambda = 1.2 * K * 1613$, 여기에 추정된 K값을 적용하면 고장강도는 약 5개(4.87/월)가 된다.

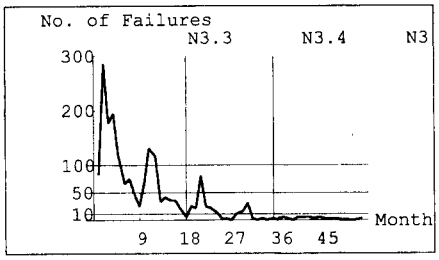

그림 1. 시험기간 동안 발견 고장 수
Fig. 1. No. of Failures during test time.

여기서 K는 프로세서의 한 사이클에 발견할 수 있는 고장율을 의미한다. 하나의 기능을 시험하는데 소요되는 프로세서의 수행시간을 계산하기 위해서 1개 기능 수행시 교환되는 메시지(IPC message : Inter Processor Communication message)의 평균 개수를 6개라 하고 수행되는 소스라인의 수를 20라인이라 하면 기계어 라인수는 480 라인이다. 따라서 평균 필요한 프로세서의 수행 시간은 $480 \times 10 \times 10^{-6} = 4.8 \times 10^{-3}$ CPUsec가 된다. 하루 평균 시험되는 항목의 수를 30개라고 하면 하루동안 수행되는 평균 프로세싱 시간은 $30 \times 4.8 \times 10^{-3} = 0.0108$, 1개월 동안 수행되는 시간은 $30 \times 0.0108 = 0.324$가 된다. 따라서 적어도 한번 그리고 꼭 한번 수행을 하기 위해 필요한 선형 수행시간은

$$T_L = \frac{I_s Q_x}{\gamma} = \frac{1334100 * 4}{10,000,000} = 0.534$$

γ : 기계어 수행율(CPU가 처리할 수 있는 명령수),
Q_x : 소스문의 평균적 기계어 번역율,
I_s : 소스 라인수

가 된다. 따라서 0.534 / 0.324 = 1.65 개월, 즉, 2개월의 기간이 전체 소프트웨어 기능에 대한 최소한 한 번의 수행을 위해 필요한 시간이다. 이 데이터는 실제 우리가 경험한 TDX 교환 소프트웨어 개발 시 소프트웨어에 대해 수행한 시험에 소요되는 시간(대략 1.5개월 정도 소요)과 근사한 것으로 나타났다.

실제 실시된 하나의 소프트웨어 버전(예, N3.3a, N3.4a, ...)에 대해 구현된 기능들이 시스템 내에서 시험되는 횟수는 대략적으로 30항목*25일*10개월≈7500회 정도로 가정된다. 실제 시험에서 검출된 고장자료를 이용하여 검출율 K를 추정할 수 가 있는데 TDX 교환 소프트웨어인 경우 각 버전별로 확인된 소프트웨어 신뢰도 성장은 S-자형 모델을 따르며, 전 개발기간에 대해서는 지수형 신뢰도 성장모델을 따르는 것으로 추정되었으며[5] 이에 대한 전체 현황은 그림 2와 같다.

Yamada et al.가 제안한 고장 발견과정과 인지과정으로 나누어 해석하는 S-자형 신뢰도 성장 모델의 평균 고장수는 (7)식과 같다.

$$\mu(t) = a(1-(1+bt)e^{-bt}) \quad (7)$$

$\mu(t)$: S-자형 모델 평균 고장수,
a : 시험전 소프트웨어내 기대 잔존 결함수,
b : 소프트웨어 결함 발견율

표 2. 소스라인 대비 기계어 변환 라인수
Table 2. Source lines vs Machine code lines.

블럭명	TDX		블럭명	ATM	
	소스 라인수	기계어 라인수		소스 라인수	기계어 라인수
asc	10929	44629	libalf	1622	4638
bpcc	14566	30789	bsif	5279	38982
csl	8938	47866	adcpf	11080	47098
dcc	24338	67098	afucf	54239	154828
dst	13216	40993	astf	15311	59608
orc	32664	115498	addbf	3782	21738
pic	7422	22225	cdgf	5003	25382
psl	2781	11164	cdmf	2929	17782
ssco	10149	43617	adecf	2599	17836
dst	13216	40993	admaf	2089	20254
총계	138,219	464,872	총계	103,933	408,146
변환비	3.363		변환비	3.927	

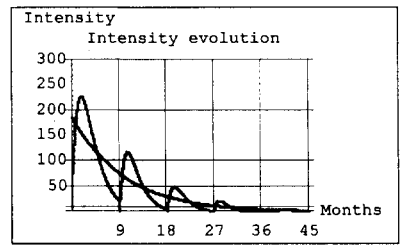

그림 2. 고장 밀도의 진전
Fig. 2. Evolution of failure intensity.

여기서 a는 시험 초기의 소프트웨어 내에 잠재하고 있는 결함의 초기값을 의미하고, b는 시험기간 동안의

결함 발견율을 의미한다. 즉, 위의 2개 파라미터는 소프트웨어 성장 곡선의 표현에 사용되는 요소로써 a의 값은 소프트웨어 개발과제에서 얻은 경험적 결과로 추정된다. b값은 시험에서 얼마나 많은 결함을 찾아낼 수 있는가를 나타내는 척도로 시험에 투입되는 인적자원과 소요시간 등 시험자원의 영향을 받으며 이를 결함 발견율이라 부른다. 이 값은 시험의 질을 표시하는 지표로도 이용된다. 모델 파라미터 a, b의 값을 추정하는데 이용되는 방법으로는 선형 근사 방법이 있으며, [2] 이에 대한 의미는 [3] [4] 에서 찾아볼 수 있다.

식 (7)을 이용하여 추정한 모델 파라미터 a, b의 값은 아래 표 3과 같다. 이때 이용되는 방법은 충분히 긴 시간 동안의 고장자료를 이용하면 log-linear 방법으로 b를 추정하고 이것을 식 (7)에 대입하여 K를 구할 수 있다. 이런 방법으로 추정한 파라미터 값은 $\overset{*}{a}=1842.04$, $\overset{*}{b}=0.1012$ 로 추정되었다. 즉, 소프트웨어 버전에 대한 총 시험기간(각 버젼별 9개월 소요)을 시험기간 T_L(36개월)에 적용하여 고장 강도를 구하면

$$\lambda(t=36)=1842.04*0.1012e^{-0.1012*36}=4.87808,$$

이때 K는 $K=\frac{4.87808}{1613*1.2}=2.52*10^{-3}$

이 된다. 따라서 버전이 시간에 따라 진전될수록 하나의 고장 발견이 고장의 원인이 되는 결함의 제거가 충분히 수행된다고 수 있다. 그러나 실제의 경우, T_L 동안에 실행되는 시험의 경우는 사용빈도에 의해 정해진 순서에 따라 결정된다. 따라서 실 시험 과정은 Zipf 법칙을 따르는 것으로 관측된다. Zipf 법칙이란 일반적 사회활동은 최단의 노력으로 최고의 효과를 보고자 하는 것으로 이 원리를 이용해 보면 최고 순위의 기능이 n번 수행된다면 i번째 순위의 기능은 i의 p 멱수(power)의 역수에 비례한다. 따라서 전체적으로 수행되는 시험의 횟수는

$n*k(1+\frac{1}{2^p}+\ldots..+\frac{1}{i^p})$ 으로 표시된다. 여기에서 k, p는 상수이다. p > 1이면 위의 급수의 상한치는 $\frac{1}{p-1}$로 주어지므로 한 사이클 동안 시험되는 전체 횟수는 위 식으로 계산되며 Musa의 FER 추정에서와 같이 임의성을 갖지 않는다. 예를 들어 p = 1.5일 때 1순위의 기능이 100번 시험된다면 순위가 5개 레벨로 정해진 시스템에서의 전체 시험 횟수는

$$100(1+\frac{1}{2^{1.5}}+\ldots+\frac{1}{5^{1.5}})=100+35+19+12+8\approx 210$$

회가 된다. 즉, 직관적인 혹은 경험적인 사실에 따라 수행되는 시험의 횟수는 기능에 따라 변화한다.

TDX 교환시스템인 경우 시험의 전체 한 사이클 동안 시행되는 시험의 횟수는 7500번 정도이나 시험 순위에 따른 전체 시험 횟수는 이보다 훨씬 더 많을 것이다. 앞의 사실에 비추어 적어도 하나의 기능이 동일 패키지 내에서 반복 시험될 횟수가 이론적으로 8~9회 정도(실제 시험 항목 수가 850 여개임)로 계산되나 실제 시험에서는 우선순위(Priority)에 따라 거의 매회 수행되는 기능이 존재하기 때문이다. 이 사실은 시스템 시험의 어려움과 시험이 진행될수록 결함 검출율이 버전별로 지속적으로 증가하는 현상을 설명하는 것으로 볼 수 있다. [표 3. 참조]

또한 Log-linear 추정 방법은 소프트웨어 신뢰도 성장 모델이 지수형임을 가정한 것으로 부분적인 고장자료에 대한 소프트웨어 신뢰도 성장은 S-자형 모델이 적용되므로 이에 대한 b값을 추정하는 연구가 필요하다.

표 3. S-자형 모델의 파라미터 a, b 추정치
Table 3. Estimated values of a and b of S-shaped model.

버전 명	a(추정치)	b(추정치)	a*(경험치)
N3.3a	1118.14	0.552831	1072
N3.3b	490.311	0.641673	480
N3.4a	188.586	0.697083	186
N3.4b	62.1272	0.937313	62

그밖에 linear regression 방법에 의한 a, b의 추정에 관한 방법도 필요하다.

2. ATM 소프트웨어 결함 검출율

4.1절에서 추정한 TDX 교환 소프트웨어에 대한 결함 검출율 K를 결함과 고장에 관한 선험(先驗) 자료로 삼아 ATM 교환소프트웨어의 신뢰도 성장의 패턴에 대해 논한다.

ATM 소프트웨어는 ATM 스위칭 기능을 위한 기본 기능을 실현한 버전을 바탕으로 하여 점차 다양하고 복잡한 기능의 추가 실현을 통해 개발중에 있다. 현 시점의 프로그램 규모는 약 85만 소스라인(SLOC : Source Line of Code)에 93개 기능 블럭이다. 따

라서 TDX 교환 소프트웨어에 대한 결함밀도 1.2/KSLOC를 적용하면 ATM 소프트웨어의 초기 결함수 ω_0는

$\omega_0 = 1.2 * \frac{850,000}{1000} = 1020$ 개로 추정된다.

이때 B값은 1020/1982 ≈ 0.515가 된다. 실제 ATM 소프트웨어의 시험에서 검출된 고장 데이터를 살펴보면 표 4와 같다.[6]

표 4. 월별 고장 발생 수
Table 4. Number of failures per month.

month	1	2	3	4	5	6	7	8	9
고장 수	15	12	42	91	69	79	34	37	34

위의 고장 데이터를 기초 자료로 지수형 신뢰도 성장 모델에 적용, 각 파라미터들을 추정(non-linear curve fitting)해보면 a = 850, b = 0.136385로 계산되며, 이에 대한 누적 고장수와 예측 고장수는 그림 3과 같다. 여기서 지수형 신뢰도 성장 모델을 채택한 이유는 대상으로 삼은 교환 소프트웨어가 개발 도중에 있으며, 시험 초기 단계의 고장 데이터를 이용하여 소프트웨어에 대한 결함수를 예측하기 때문이다.

ATM 교환시스템의 제어를 위해 채택된 TM Super SPARC(clock speed 50MHz) CPU의 명령어 처리 능력인 r = 170 MIPS와 기계어 명령수 I = 4*850,000 SLOC를 적용하면 수행 빈도 f = r/I = 170 MIPS/4*850,000 = 50 CPUsec가 된다.

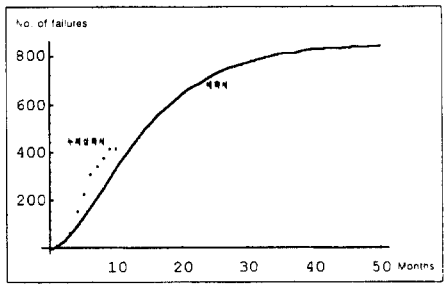

그림 3. 누적고장수와 S-자형모델 예측고장수
Fig. 3. Cumulative failures and the predicted graph of S-shaped model.

2절에서 언급한 식 (1)로 부터 ν_1를 구하면 $\nu_1 = \frac{\omega_1}{B} = 1020/0.515 = 1980.58$, 즉, ν_1 = 1980개가 되며, 또 λ_1는 지수형 신뢰도 성장 모델에서 추정한 a, b값을 이용하여 구하면 $\lambda_1 = abe^{-bt}(t=0) = 115.927$ 이 된다. 위에서 구한 B, ν_1, λ_1의 데이터 값들을 2절의 식(1), (2), (3)에 적용하여 소프트웨어 결함 검출율(K)의 값을 식(3)으로부터 구할 수 있는데,

$$K = \frac{\lambda_1}{B * f * \nu_1} = \frac{115.927}{0.515 * 50 CPUsec * 1980} = 2.27 * 10^{-3}$$

이 된다. 종합적으로 TDX 계열 교환 시스템에 대한 소프트웨어 결함 검출율의 결과를 살펴보면 표 5와 같다.

표 5. TDX 계열 교환시스템 결함 검출율
Table 5. Fault exposure rate of TDX series.

시스템 명	결함 검출율(K)
TDX 교환시스템	2.52×10^{-3}
ATM 교환시스템	2.27×10^{-3}

ATM 시스템에 대한 결함 검출율을 이용하여 고장강도(λ)를 구하면

$\lambda = 1.2 * K * 1020 = 1.2 * 2.27 * 10^{-3} * 1020 = 3.3$이 되며, 교환 시스템의 결함 검출비에 의한 고장강도를 종합하면 표 6과 같다.

표 6. 교환시스템의 고장강도
Table 6. Fault Density.

시스템 명	고장강도(월)	비 고
TDX시스템	4.8개	시험종료단계
ATM시스템	3.3개	시험초기단계

위에서 구한 결함 검출율을 이용하여 시험기간(T_L)을 구하면 하나의 기능을 수행하는데 소요되는 평균 프로세서 수행시간을 구하고 메시지 처리 개수 3개와 수행 소스 라인수 10 스텝, 1일 수행 항목수 20을 적용하면 하루 평균 프로세싱 시간은 3 x 10 x 20 x 170 MIPS = 0.102가 되며, 1개월 동안 수행되는 시간은 30 x 0.102 = 3.06으로 계산된다. 이때 한번 이상 수행하기 위해 필요한 시간 즉, 달력시간(calendar time) C_T 는

$$C_T = \frac{1.2}{3.06} = \approx 0.4 month$$

가 된다. 즉, ATM 소프트웨어의 전체 기능을 1회 시험하는데 약 2주일이 소요되며, 이때 발견되는 대부분의 고장에 대한 소프트웨어 결함 제거가 수행된다고 할 수 있다. 이와 같은 사실은 실제 시험에서 단일 패

키지의 전체 기능을 시험하는데 소요되는 시험기간인 2주(15일)와 거의 일치하고 있음을 의미한다. 교환 시스템의 소프트웨어 기능을 시험하여 결함을 검출하는데 소요되는 기간은 표 7에 나타냈다.

두 시스템간 시험기간의 차이는 개발환경 및 디버깅 환경 제공 등 시험환경의 차이와 메인 프로세서(CPU)의 프로세싱 능력 등의 하드웨어 성능 차이, 구현된 소프트웨어 기능의 다양함과 복잡성(기능 블록간 인터페이스 문제) 때문에 TDX 시스템이 다소 많이 소요되는 것으로 계산되었다.

표 7. 교환 소프트웨어 기능시험 기간
Table 7. An execution time for the software test.

소프트웨어 명	기능시험 기간(월)	비 고
TDX Software	1.65	CPU 클럭 25MHz 디버깅환경 열악
ATM Software	0.4	DOS 환경지원 CPU 클럭50MHz

두 시스템간의 구현된 소프트웨어의 커다란 차이점은 TDX 교환 시스템의 운용자 정합(MMI : Man Machine Interface) 기능이 시스템 내에 구현된 반면에 ATM 교환 시스템은 HMI(Human Machine Interface) 기능 및 TMN(Telecommunication Management Network) 관련 기능이 시스템 외부에 별도의 기능으로 구현되어 시스템과 인터페이스 되고 있는 것이 두드러진 특징이다.

ATM 시스템의 시험기간은 추가되는 기능과 구현되고 있는 프로그램의 규모가 점차 증가되는 추세에 있는 개발중인 시스템이므로 향후 그 기간은 점차 늘어날 것으로 예상된다.

V. 결 론

과거의 경험적인 고장 데이터로부터 소프트웨어에 대한 결함 검출율을 구하고 이를 토대로 새로운 시스템인 ATM 교환 시스템의 결함 검출율을 예측해 보았다. 위의 2개 시스템의 결함 검출율은 TDX 교환 시스템이 ATM 시스템보다 다소 높은 것으로 나타났는데 그 이유는 TDX 교환 시스템이 시험의 최종 단계의 현장 배포 시기에 수행된 고장 데이터인 반면에 ATM 교환 시스템의 경우는 개발중인 시스템으로서 소프트웨어 시험이 점차 진행되고 있는 단계이다. 즉, 구현된 기능에 대한 정확한 인지와 시험 시나리오의 숙련도 등 시험 능력이 배가되는 현장 배포 단계에서는 소프트웨어 결함 발견 능력이 최고조에 이른다. 다시 말해서 개발 단계의 소프트웨어 결함 검출율은 현장 배포시기(현장 운용단계)에 비해 다소 떨어진다. 앞에서 언급한 것과 마찬가지로 결함 검출율은 시험의 각 단계를 거치면서 소프트웨어에 대한 결함 검출율이 점점 높아져 가는 것을 표 2의 내용으로 알 수 있다. 또 실제 시험에서 검출된 고장 데이터를 토대로 지수형 신뢰도 성장 모델에 근거하여 평가한 추정치와 결함 검출율을 응용한 예측치를 상호 비교함으로서 유사한 교환 시스템 개발시 대상 소프트웨어의 초기 고장밀도(Initial fault density)로부터 정확한 결함 검출율을 추정할 수 있는 방법이 좀더 연구되어야 한다.

초기 소프트웨어에 대한 결함 검출율은 소프트웨어 신뢰도 성장 과정을 추적하는데 매우 중요한 요소로 각 시험 단계에서의 초기 고장강도와 함께 소프트웨어 신뢰도 및 시험기간 등을 결정할 수 있는 중요한 파라미터이다. 다시 말해서 소프트웨어 신뢰도 분석의 정확성을 향상시킬 수 있으며, 이러한 접근 방법은 신뢰도 모형의 대표적인 Time base model과 Coverage model의 정확성을 높일 수 있는 방법으로 활용될 수 있다.

참 고 문 헌

[1] Martin L. Shooman, "Software Engineering : Design, Reliability, and Management", New York : McGraw-Hill, pp. 335- 340, 1988.

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

[3] Yashwant K. Malaiya, "An Estimation of Fault Exposure Ratio", IEEE Transaction on Software Engineering, vol. 19, no. 11, pp. 1087-1094, Nov. 1993.

[4] Li Naixm, Yashwant K. Malaiya, "Fault Exposure ratio Estimation and Application", IEEE Transaction on Software Engineering, pp. 372-381, 1996.

[5] 유재년, 이재기, "기능 블록으로 구성된 대형 소프트웨어의 신뢰도 성장", 대한전자공학회논문지, 제35권 S편 제1호, pp. 29-38, Jan. 1998
[6] 유재년, "소프트웨어 신뢰도 성장 모델의 일반형", 대한전자공학회논문지, 제35권 C편 제 5호, pp. 331-336, May. 1998.
[7] Norio MURATA and Masato OH-MINAMI, "Quality Control for Large Scale Software", IEEE/Elsevier Science Publishers B.V.(North-Holland), pp. 1043-1046, 1984.
[8] Masato et al. "Capture & Recapture 법에 대한 잠재 버그의 추정법과 그 응용", 정보처리학회 소프트웨어 공학 연구회 자료 19-1, Japan, July 1981
[9] 이재기외, "고장데이터 분석을 통한 교환 소프트웨어 특성 연구", 한국통신학회논문지, 제 23권 8호, 1998. 9

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) 第 36卷 第 3號 參照
주관심분야는 Software Integration & Test

辛 相 權(正會員) 第 36卷 第 3號 參照
주관심분야는 Software Integration & Test