소프트웨어 신뢰도 정량화를 위한 검증 방법 연구

Empirical Studies of Testing Method for The Software Reliability Measurement

이재기(J.K. Lee) 시스템종합팀 선임연구원
신상권(S.K. Shin) 시스템종합팀 기술기능원
남상식(S.S. Nam) 시스템종합팀 책임연구원, 팀장

소프트웨어 검증에 사용되는 신뢰도 평가를 위한 기초 고장데이터를 수집하는 시스템 시험 방법과 소프트웨어 종합화에 고려되어야 할 사항들에 대해 제안한다. 그 외에 ATM 교환시스템 개발에서 얻은 경험을 정리하여 차세대 시스템 개발의 소프트웨어 품질 향상과 신뢰도 평가에 활용되도록 하였다.

I. 서 론

종래의 소프트웨어 시험 방법은 개발된 프로그램의 동작을 요구사항에 대한 만족 여부로 검증하는 방법을 취해 왔다. 즉, 시스템 시험에서 전체 시험대상 항목에 대한 부적합 항목의 비율로 판정하는 Nelson's Method로서 고전적인 방법이 이용되었다.[1] 이 방법은 시스템 개발 초기단계에 자주 사용되는 방법으로 널리 이용되고 있으나, 소프트웨어 개발 과정의 환경 변화나 시간의 흐름에 따른 고장 발생 및 제거 과정에 대한 해석이나 예측이 어려워 기능에 대한 검증이 충분하지 못한 방법이다.

시스템 개발 과정에서 소프트웨어 신뢰도 평가 기법을 활용하여 소프트웨어에 대한 검증을 정량화하는 것을 소프트웨어 신뢰도 측정이라고 부르며, 시스템 운용 시 고장 발생을 최소화하기 위해 시험을 통해 고장에 대한 자료를 얻고 이 데이터를 이용하여 적합한 신뢰도 모델을 선택, 분석하여 운용단계의 고장률(failure intensity)을 예측하고 이 고장률을 시스템 설계 목표치에 접근하도록 개발 과정을 제어하는 것이 신뢰도를 측도로 하는 검증의 목적이다.

신뢰도를 이용하는 소프트웨어 검증 방법은 일반적으로 종합(integration), 시험수행(test execution), 신뢰도 평가(reliability evaluation)순으로 진행되며, 시스템 개발 과정에서는 이와 같은 과정이 반복 수행된다. 또한 소프트웨어 신뢰도 측정은 개발의 마지막 단계에서 한 번의 분석을 통해 측정이 이루어지는 것이 아니라 시스템 개발 전 단계에 걸쳐 이루어진다[2-3]. 위와 같은 개발 방법을 응용하여 개발에 이용되는 소프트웨어 엔지니어링 기법을 평가하고 또 개발 과정을 확인할 수 있으며, 개발된 소프트웨어의 관리 계획을 효율적으로 수립, 시스템 엔지니어링 기술의 발전을 꾀할 수 있다.

본 논문에서는 소프트웨어 신뢰도 측정 방법을 소개하고 이 기법들을 이용하여 정량화한 소프트웨어 검증을 위한 교환시스템의 고장데이터 수집 방법 및 시험의 원리, 소프트웨어 종합화 과정에서 고려되어야 할 사항에 대해 기술한다. 제II장에서 검증기법의 소프트웨어 신뢰도에 대해 살펴보고, III장에서 교환 소프트웨어의 신뢰도 특성을, IV장에서 시스템

시험의 중요성에 대해 알아본다. 마지막 V장에서는 소프트웨어 종합과 관리 방법에 대해 언급하고 결론 및 향후 연구 방향에 대해 논하고 맺는다.

II. 검증 기법의 소프트웨어 신뢰도

대표적인 소프트웨어 신뢰도는 소프트웨어 품질 측도를 시간에 의존하는 평균고장간시간(Mean Time Between Failures: MTBF)으로 표현하고자 하는 것으로 단위시간당 발생하는 고장을 나타내는 고장률로 표현한다. 다시 말해서 "주어진 환경 하에서 일정 시간 동안 프로그램이 고장 없이 동작할 확률" 로 정의하고 있다[3].

소프트웨어 검증은 요구사항을 분석하여 개발된 기능의 동작 확인을 위해 설정된 테스트 데이터(혹은 시험항목)를 구성하고 시험을 수행하여 얻어진 자료에 대해 수학적인 모델을 이용, 통계적인 방법으로 분석되기 전까지의 짐작이다.

1. 기본 개념과 원리(Basic concept & principles)

소프트웨어 신뢰도는 주어진 환경에서의 프로그램이 동작하고 있는 상황에 따른 고장 발생 가능성에 대한 확률적 측도(measure)이므로 신뢰도를 말하고자 할 때는 시스템에 따라 결정되는 고장의 정의와 분류가 이루어져야 한다.

가. 고장의 정의와 분류

프로그램이 시스템에 이식되어 동작될 때 요구사항에 정의된 동작과 불일치하는 것을 의미하며, 고장의 분류는 크게 고정적인 고장(solid failure)과 간헐적인 고장(intermittent failure)으로 분류되고, 재현성에 따라 간헐고장과 완전고장(complete failure)으로 또 완전고장은 돌발고장(catastrofic failure)과 열화고장(degradation failure)으로 세분되고, 원인에 따른 1차 고장(primary failure)과 2차 고장(secondary failure)으로 그 외의 파급효과에 따른 단일고장(single failure)과 복합고장(combined failure), 고유결함고장(inherent weakness failure), 초과 스트레스 고장(over-stress failure), 고장의 중요도에 따른 치명고장(critical failure), 중고장(major failure), 경고장(minor failure) 등으로 고장의 분류 방법이 매우 복잡하다[4].

나. 고장과 결함(fault)의 정의

- 고장: 각종 요구사항이나 규격에 맞지 않는 경우로 사용자 관점의 판단임
- 결함: 프로그램이 실행될 때 고장을 제공하는 원인으로 주로 개발자 관점의 판단임

고장과 결함은 성격이 달라 엄격히 구분되어야 하나 구분 방법이 어렵고 시험 수행 시 프로그램 내 결함이 위치한 부분이 수행 될 때마다 외부로 나타나는 현상이 고장으로 판단되거나 프로그램 내에서의 임의의 시험 경로의 선택여부에 따라 에러가 발견되기도 하고 발견이 안 되는 경우가 있다. 그렇기 때문에 고장, 결함, 에러의 한계 구분과 프로그램 내에 잔존하는 수를 추정할 때 고장과 결함 중 어느것이 많다고 단정하기가 어렵다. 일반적인 추측은 하나의 결함에 대해 다양한 시험경로에 의해 여러 가지 고장이 발생할 수 있기 때문에 결함보다는 고장이 많다고 추정한다.

다. 고장 발생을 표현하는 시간

신뢰도 평가에 이용되는 시간의 단위는 일상시간(calendar time)과 프로그램 수행시간(execution time)이 있다.

라. 고장발생 표현 방법

고장 발생에 대한 기록은 프로그램이 수행된 시간에 근거하여 기록되며, 고장간 시간을 기록하는 방법과 단위시간 동안 발생하는 고장으로 표현하는 방법이 있다[5].

마. 고장시간 기록

고장데이터 수집에 많은 시간이 소요되며, 이 형

태의 데이터는 신뢰도 평가를 정확히 할 수 있다.

바. 고장데이터의 그룹화

단위시간 동안 발생하는 고장 수를 말하며, 데이터 수집이 용이한 반면 신뢰도 측정의 정확도가 떨어져 소프트웨어의 고장 특성을 정확히 파악하기 힘드나 전반적인 경향 파악이 용이하다.

2. 소프트웨어 신뢰도 해석에 필요한 사항

신뢰도 해석에 필요한 중요사항으로는 시스템의 형상과 동작 특성을 잘 나타낼 수 있는 운용 프로파일(operational profile) 및 모델의 변수이다. 또 시스템 개발 과정의 각 상황에 대한 검토를 가능하게 하고 시스템 개발전략 수립에 지표가 되는 고장률은 필요한 각종 자원의 관리 및 배분, 시험방법의 변경, 기능추가의 시기를 결정하는 데 도움을 준다. 이와 같이 소프트웨어 신뢰도 평가는 이러한 지표들을 추정하는 데 이용된다.

3. 대표적인 신뢰도 모델

대표적인 소프트웨어 신뢰도 평가 모델은 시간계측 모델, 개수계측 모델(error counting model), 가용도 모델(availability model), 경향곡선 모델(trend analysis model) 등이 있다. 그 중에서 교환시스템 개발에 쉽게 적용될 수 있는 개수계측 모델 중 두 가지만 소개하면 지수형 NHPP(Non-Homogeneous Poisson Process) 모델과 Logarithmic Poisson 모델이 있다[6].

가. 모델의 가정

고장은 시스템 시험이나 운용중에 시간의 경과에 따라 임의로 발생하며, 이것은 Poisson process로 표현된다. Poisson process는 $\mu(t)$, $\lambda(t)$ 로 표현된다. 이때 시간 t까지의 총 누적 고장 수는 $N(t)$ 로 표현된다. 여기서 t는 수행시간 또는 일상시간을 의미한다. 각 파라미터에 대한 의미는 아래와 같다.

- $\mu(t)$: Expected Number of cumulative failure to be experienced by time t
- $\lambda(t)$: Expected Number of failure to be experienced per unit time t
- $N(t)$: cumulative Number of failures occurred by time t
- t: Calendar time or execution time

시스템 시험이 진행되는 동안 소프트웨어에 대한 고장 발견과 수정은 동시에 수행되며, 고장률은 시간이 경과함에 따라 감소하는 모델인 비동차 포아송(NHPP) 모델이 적용된다. 그러나 운용단계의 소프트웨어 고장은 새로운 버전의 소프트웨어가 배포(release)되기 전까지는 결함이 제거되지 않기 때문에 시간이 어느 정도 지난 후에야 결함의 수정(correction)이나 새로운 기능의 추가가 이루어진다. 이 때에는 HPP 모델이 적용된다. 포아송 과정에 의한 t 시간까지 j개의 고장이 발생할 확률은 식(1)로 표현된다.

$$P[N(t)=j]=\frac{(\mu(t))^j}{j!}e^{-\mu(t)},\ j=1,2,3 \qquad (1)$$

시간의 흐름에 따라 발생하는 전체 고장 수가 유한한 것으로 보는 모델의 대표적인 예로써 발생하는 고장이 프로그램 내부에 잔존하고 있는 결함 수(혹은 고장 수)에 비례한다고 가정하면 이것은 식(2)와 같이 표현되며, 지수형 NHPP 모델을 얻는다 [1].

$$N(t)=v_0(1-e\frac{-\lambda_0 t}{v_0}) \qquad (2)$$

나. 지수형 NHPP 모델(Exponential NHPP Model 혹은 Goel-Okumoto Model)

$$\mu(t)=v_0[1-e^{\frac{-\lambda_0}{v_0}t}] \qquad (3)$$

$$\lambda(t)=\lambda_0 e^{\frac{-\lambda_0 t}{v_0}} \qquad (4)$$

λ_0 : Initial failure intensity

v_0 : Total expected failures in finite time

$\frac{\lambda_0}{v_0}$: Failure detection rate per failure

또 시간의 흐름에 따라 발생하는 고장 수가 무한하다고 보는 모델로 고장률이 기대되는 고장 수에 지수적으로 감소한다고 가정함으로써 표현되는 모델을 얻을 수 있는데 이는 대수형 포아송 실행시간 모델(logarithmic poisson model) 또는 Musa-Okumoto (M-O) model로 불린다. 이 모델에서 t는 프로그램 수행시간을 의미한다[2].

다. Logarithmic Poisson Model

$$\mu(t) = \frac{1}{\theta}\ln(\lambda_0 t + 1) \quad (5)$$

$$\lambda(t) = \frac{\lambda_0}{(\lambda_0 \theta t + 1)} \quad (6)$$

θ : Normalized failure intensity reduction rate per failures

t : Execution time

라. 모델 파라미터의 의미(Meaning of the model parameter)

위에서 언급한 두 개의 모델에 사용되는 파라미터는 소프트웨어 개발 과정을 설명할 수 있는 시험방법, 자원의 사용정도, 프로그램의 크기, 디버깅의 효율성 등을 이용하여 표시할 수 있다. 이 부분은 추후 연구가 더 필요한 부분이다.

마. 모델의 분석도구

모델의 분석은 모델 파라미터들을 실제 측정데이터를 이용하여 추정하는 것으로서 대표적인 방법은 MLE(Maximum Likelihood Estimation)와 LSE (Least Square Estimation)이 있다.

대표적인 도구로는 SAS(Statistics Analysis System)와 Mathmatica™, Mathlab™ 등이 있으며 다양한 모델 함수를 선택하고 분석하여 모델 파라미터를 추정하는 공식을 변형하여 모델 함수의 수학적인 해석을 위해 통계 패키지나 범용의 수학 패키지 등이 이용된다.

III. 교환 소프트웨어의 신뢰도 특성

교환 소프트웨어의 신뢰도 특성은 시스템 내부에서 수행되는 기능의 다양함과 광범위한 입력 상태를 갖는 운용 프로파일과 각 기능의 동작에 대한 높은 신뢰성을 들 수 있다.

1. 교환 소프트웨어 고장 정의

교환시스템의 기능은 사용자 요구사항을 실현하는 기능과 이를 지원하는 시스템 내부 기능으로 나누어 진다.

사용자 요구사항을 실현하는 기능으로는 호처리, 운용관리, 유지보수기능으로 분류되고 시스템 내부기능으로는 응용 프로그램을 지원하는 운영체제(Operating System)와 DBMS(Data Base Management System)로 구분된다.

교환시스템의 고장은 각종 필요조건 및 시스템 개발 규격의 기준에 맞지 않는 경우를 포괄적으로 말하며, 소프트웨어는 다양한 기능을 수행하므로 이때 발생되는 고장은 여러 가지로 분류가 가능하나 대체로 서비스에 영향을 주는 정도에 따라 분류된다. 이것을 분류해보면 <표 1>과 같다.

<표 1> 교환시스템의 고장 분류

등급	서비스에 미치는 영향	비고
A	기본서비스 처리 불가	주요 기능
B	기본서비스 처리 품질 저하	
C	부가서비스 오동작	
D	서비스에 영향을 미치는 사소한 고장(예; 신호음 주기 오차)	운용상 지장없음

가. 운용 프로파일

운용 프로파일은 시스템 운용에 영향을 미치는 환경적 요소를 말하며, 이것들은 주로 시스템에 수용되는 가입자의 종류나 지역적 특성, 망 구성 상태, 가입자의 호 습성 및 가입자 대 중계선의 비율 등이 속하며 시스템이 처한 환경에 따라 다양하게 변화된다. 그렇기 때문에 운용 프로파일은 신뢰도 평가에 매우 중요한 역할을 한다.

나. 고장률 목표

교환시스템에 요구되는 서비스 중단시간(down

time)인 신뢰도 목표치는 20년당 1시간 정도로의 높은 조건을 갖는다. 이는 전체 시스템이 다운되는 정도를 나타내는 수치로 일반적인 기능의 신뢰도 목표치는 이보다 낮은 것으로 기대된다. 이처럼 고장률 목표치는 사용자와 시스템 엔지니어에 의해 프로젝트의 규모나 개발기간 등을 고려하여 구체적으로 정해져야 한다[7-8].

Ⅳ. 시스템 시험과 고장데이터 수집

시스템 시험의 목적은 개발된 기능의 정상 동작 여부를 검사하는 행위이며, 시스템의 신뢰도와 성능을 평가할 수 있는 자료수집을 포함한다.

1. 시스템 시험의 중요성

시스템 시험은 기능 시험에서 개별적으로 확인된 기능들을 종합하여 수행하는 시험으로써 기능 시험에서 찾지 못한 고장이나 결함들을 종합화한 환경에서 발견할 수 있고, 이 과정에서 새로운 문제를 찾아낼 수 있기 때문에 개발 과정에서 대단히 중요하다. 또 시험에서 수집되는 여러 가지 데이터를 이용하면 시스템의 성능 및 신뢰도 향상에 많은 기여를 할 수 있기 때문에 소프트웨어 개발 과정에서는 반드시 수행되어야 한다. 그러므로 시험이 개발에서 차지하는 비중이 매우 크다.

가. 종래의 시스템 시험 방법

시스템 개발 과정에 수행된 방법은 기능 개발의 일정에 따라 추가, 변경되는 기능 위주로 운용자에 의해서 주어지는 입력데이터에 의한 반응 결과를 확인, 기능의 정상 여부를 판단하였다. 이 방법은 개발 초기단계에 주로 적용되는 방법으로 Black Box Test라 불리며, 이것은 신뢰도 측정을 위한 시험으로는 다소 문제가 있다.

시스템이 field에 설치, 운용될 때 각 기능들이 서로 조합되어 multi-task processing되기 때문에 기능 위주의 검증시험은 미흡한 점이 많아 정량적인 신뢰도 평가를 위해서는 새로운 개념의 시험 방법의 도입이 필요하다.

나. 새로운 개념의 시험 방법

이 방법은 운용 프로파일을 고려한 기능의 분류, 중요도에 입각한 대상 시험항목의 선정, 반복시험 도입, 시험 수행절차(test scenario) 등을 고려한 소프트웨어 매트릭스 구조에 적합한 시험 방법이 적용되어야 한다. 즉, 개발 초기에는 시험 시나리오에 의한 정상적인 기능 수행 정도를 확인하고 점차 운용단계에 이르면 random하게 시험 시나리오를 배합하여 수행한다. 또 선정된 기능들에 대해 반복시험을 함으로써 일련의 기능들이 상호 조합된 형태로 교환시스템이 운용되는 것과 유사한 환경으로 simulation되어 여기서 얻어진 데이터를 수집하고 선별하여 신뢰도 평가를 위한 기초자료로 이용하여야 한다. 아울러 불필요한 자료에 의해 정확한 신뢰도 평가 및 예측이 이루어지지 않기 때문에 수집된 데이터는 전문가에 의해 철저히 분석되어야 한다.

소프트웨어 신뢰도 측정을 위한 시스템 시험에 고려되어야 할 사항은 다음과 같다.

다. 시험항목의 결정

시스템 시험은 시스템이 현장에 설치되어 겪게 될 상황이나 환경과 근사한 환경을 구축, 실시되어야 한다. 하나의 기능을 시험할 때 현장에서 발생하는 경우와 연관된 기능이 가능한 한 포함될 수 있도록 운용 프로파일을 참조하여 시험항목을 정하고 기능에 대한 반복확인이 가능토록 선정한다.

라. 반복시험(Regression Test)

이 과정은 소프트웨어 패키지 제작과 관계하여 대폭적인 기능의 추가없이 결함의 수정에 대한 내용만으로 선택된 모델의 고장률 함수가 목표치에 이를 때까지 시험을 반복 수행한다.

소프트웨어 신뢰도 정량화에 이용되는 고장데이터를 얻기 위해 시스템 시험에서 얻어지는 고장데이

터는 패키지의 버전, 운용 프로파일, 시험 시작시각 및 종료시각, 시험항목, 시험 준비기간 등을 모두 포함하며, 고장데이터는 필히 여과과정을 거쳐 가용한 데이터로 만들어지는데 이것은 일정한 형식을 가지고 기록되어야만 편리하게 처리할 수가 있다.

2. 고장데이터 수집과 시험 환경의 설정

소프트웨어 신뢰도 측정 노력은 보다 정확한 고장데이터 수집을 요구한다. 먼저 수집하고자 하는 데이터가 정해져야 하며, 데이터 수집 담당자를 결정하여야 하는데 주로 공정성을 유지하기 위해 시험자가 그 역할을 수행한다.

데이터 수집 행위는 일관된 데이터 수집을 위해 수집에 대한 동기와 과정 그리고 얻게 될 각종 결과들과 분석될 결과들이 어떤 효과를 가져다 주는지에 대한 사전 교육을 거쳐 실시한다.

고장데이터 수집은 신뢰도 평가 모델에 필요한 가정들을 만족시켜 줄 수 있도록 수집되어야 한다. 필요한 가정들은 고장이 즉시 수정되는 것과 큰 기능의 변동은 발생치 않는다는 것이다.

시험 환경의 설정은 실제 운용 환경과 유사한 환경으로 구축, 여러 경우의 수로 검증되어야 한다.

◈ 고장의 기록

고장 발생을 나타내는 시각 혹은 단위시간 동안 발생하는 고장 수를 정확히 기술해야 한다. 이렇게 얻어진 고장데이터는 신뢰도 평가단계에서 재분류되어 평가에 가용한 데이터로 만들어진다.

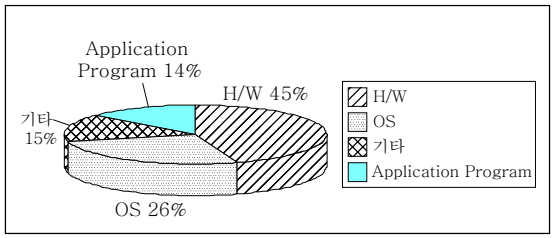
지수형 NHPP 모델과 대수형 비동차 실행시간 모델의 시간 t까지의 전체 고장 수와 시간 t에서의 고장률 감소는 식(7)과 식(8)로 표현된다.

$$N(t) = \nu_0 (1 - e^{-\frac{\lambda_0 t}{\mu_0}}) \quad (7)$$

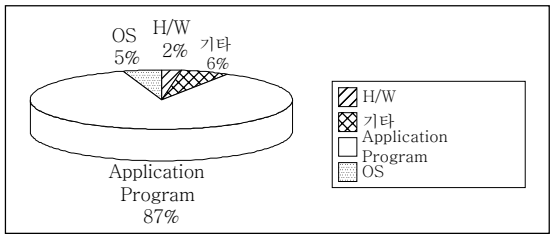
$$\lambda[\mu(t)] = \lambda_0 e^{-\theta\mu(t)} \quad (8)$$

3. 실제 적용 결과

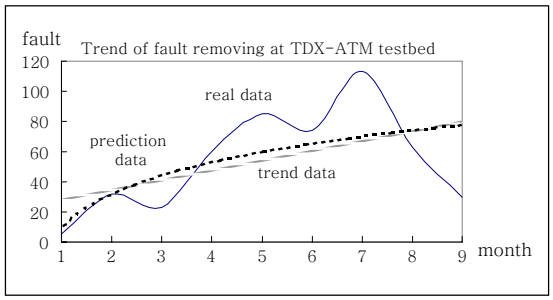
(그림 1)에서 보여준 고장데이터 수집의 예는 T



(그림 1) 개발단계의 고장 분포도



(그림 2) 운용단계의 고장 분포도



(그림 3) 고장 제거 활동의 추이도

DX-ATM 교환시스템의 개발 중에서 소프트웨어 버전 SV3.2와 SV3.4에 대한 고장 분포도이다. VC (Virtual Channel) 교환 개념이 도입된 시스템 개발 초기단계를 살펴보면 하드웨어 고장이 전체 60% 정도 차지하고 있음을 파이차트에서 알 수 있다. 이 기간에는 시스템을 구성하는 하드웨어의 변경 및 Firmware, 시스템의 리소스를 제어하는 OS가 대폭 변경되고 이를 지원하기 위한 개발 환경인 Compiler의 수정 작업이 수반되었다.

개발단계를 지나 시스템 개발 마무리단계인 운용단계에 점차 다가가면서 응용 프로그램의 분포가 많아지고 있음을 (그림 2)로 알 수 있다.

(그림 3)은 응용 프로그램의 변화에 따른 테스트

베드에서 실시된 고장 검출 및 제거에 대한 추이로 소프트웨어 내에 존재하고 있는 잔존결함의 제거 활동에 대한 데이터 수집 결과이다. 다시 말해서 시스템 개발 전략에 맞추어 신뢰도 목표치(95%)에 도달하기 위한 시험능력 향상의 결과가 4개월 이후 8개월까지 지속되어 많은 고장들이 발견, 조치되고 있음을 (그림 3)을 통해서 알 수 있다.

V. 소프트웨어의 종합과 관리

소프트웨어의 종합(integration)은 개발계획에 따른 시스템 시험에 필요한 형상이나 운용 프로파일에 따라 패키지를 제작하고, 시스템의 형상과 프로그램의 변경을 관리하여 검증에 필요한 환경이나 정보를 제공한다. 종합은 소프트웨어 관리뿐만 아니라 신뢰도 평가에 필요한 데이터를 고려하여야 하고, 기능들의 추가 및 내용 변경 등 개발일정에 따라 정기적으로 이루어지며, 시험에서 도출된 고장을 고치기 위한 프로그램의 변경이나 기능의 추가는 재시험을 수행할 때 새로운 패키지로 종합되는 것이 보다 정확한 신뢰도 평가데이터를 얻을 수 있다. 이런 이유 때문에 소프트웨어 신뢰도 측정을 위해서는 소프트웨어에 대한 종합 및 관리는 시스템 시험과 함께 검증의 중요한 결과가 된다.

ATM 교환시스템 같은 대규모 분산처리시스템은 시스템 내에 분산되어 있는 기능 블록간의 정보교환을 통해 기능이 분담 처리되고 있는 시스템으로서 연동에 필요한 공통정보(software component)들을 효율적으로 관리하여야만 시스템의 성능을 향상시키고 유지보수가 용이하게 된다. 이와 같은 소프트웨어의 공통정보를 구성형태에 맞게 체계적으로 관리되고 있는 버전별 ATM 교환 소프트웨어의 관리 현황은 <표 2>와 같다.

<표 2> 버전별 ATM 소프트웨어 관리 현황

Ver.	Date	기능 수	블록 수	규모	비고
3.2	'96.6.	101	68	313KL	VCC
3.4	'97.2.	162	115	568KL	VCC
3.5	'99.6.	324	120	977KL	VCC

* VCC : Virtual Channel Connection System
KL : Kilo-Line

VI. 결 론

본 고는 ATM 교환 소프트웨어를 개발하면서 경험한 결과를 토대로 작성된 결과이다.

소프트웨어 신뢰도 평가를 위해 소프트웨어 등록 변경 요구서 및 각 버전별 시험을 통해 얻어진 데이터를 이용, 고장시간에 따른 변화를 설명하고 시스템 시험 중 발생하는 고장데이터의 수집 이유와 방법 등에 대해 고찰하였다.

신뢰도 측정 모델은 고장을 일으킨 소프트웨어 결함이 발견 즉시 수정되는 것을 가정하고 있으므로 고장이나 결함은 수행된 시스템 시험기간에 따라 정리되어야 하며, 소프트웨어 종합화는 이런 점을 고려하여 각 단계별 시험이 끝나고 결함이 수정된 후에 종합 및 재시험이 되어야 한다.

신뢰도 기법을 이용한 소프트웨어 검증의 정량화는 모델에 의존하기 보다는 실측데이터에 의해 영향을 크게 받으므로 앞으로 소프트웨어 신뢰도 평가를 위해서는 시스템 시험 중에 발생된 고장데이터 수집에 많은 노력을 기울여야 한다.

향후 연구방향은 정확한 고장데이터 수집 후 소프트웨어의 개발 과정을 잘 설명할 수 있는 모델에 대한 판단과 모델 분석도구의 개발, 자동화된 고장데이터 수집 방법, 시험자동화 연구 등이 시스템 개발과 병행되어야 한다. 아울러 시스템 시험의 반복횟수, 기간, 신뢰도 목표치와의 간격, 개발비용이나 자원의 사용정도에 대한 종합분석도 진행되어야 한다.

끝으로 본 고에서 제안한 신뢰도 측정의 원리와 시스템 시험 방법이 차세대 시스템 개발에 있어서 신뢰도 측정이나 소프트웨어 품질 평가에 이용될 수 있기를 기대한다.

참고문헌

[1] Armit L. Goel, "Software Reliability Models: Assumptions, Limitations and Applicability," *IEEE Trans-*

action on Software Engineering, Dec. 1985.

[2] J. D. Musa and K. Okumoto, "A Logarithmic Poisson Execution Time Model for Software Reliability Measurement," *Proc. 7th Int. IEEE conference on Software Engineering*, May 1984.

[3] Gregory A. Kruger, "Validation and Further Application of Software Reliability Growth Model," *HP journal*, Apr. 1989.

[4] □□康□, "ソフトウェアの檢□と□□□□: 故□□□," □科技□ソフトウェア□□管理series□ 4卷, 1986, pp. 137 – 140.

[5] Samuel Keene and Christopher Lane, "Reliability Growth of Fielded Software," *Proc. Annual Reliability and Maintainability Symposium*, 1993.

[6] Shigeru Yamada, Hiroshi Ohtera, "Software Reliability Based on Theory and Practical Application," *Software Research Center SE Series*, Feb. 1992, pp. 179 – 195.

[7] AT&T, "Software Reliability Measurement Practical Application," *Kelly Education & Training Center*, 1985.

[8] N. Li, K. Malaiya, "Enhancing Accuracy of Software Reliability Prediction," *Proc. 4th Int. Symposium on SRE*, Nov. 1993.