論文98-35S-1-4

기능 블록으로 구성된 대형 교환 소프트웨어의 신뢰도 성장

(An Evolution of Reliability of a Large Switching Software Composed of Functional Blocks)

劉 宰 年 *, 李 在 起 *

(Jae-Nyun Yoo and Jae-Ki Lee)

요 약

이 논문은 소프트웨어 구성요소가 기능에 따라 분할된 소프트웨어 기능블록인 대형 교환 소프트웨어에 대한 소프트웨어 신뢰도 공학의 응용 연구이다. 소프트웨어 신뢰도 관점에서 개발 기간 동안 필요한 재고(review), 튜닝, 기능 추가, 시험 등의 여러 활동에 대한 주요 관리 시점의 결정을 위하여 소프트웨어 고장의 검출과 수정 과정을 재조명하고, 개발과 실행 기본 단위가 되는 블럭 별 고장에 대한 통계적 분석과 부분적 고장데이터에 대해 S-자형, 전체 데이터에 대해 지수형 소프트웨어 신뢰도 성장 모델을 응용하여 대형 교환 소프트웨어의 신뢰도 성장의 한 패턴을 제시한다. 본 논문의 고장 데이터에 대한 분석과 신뢰도 해석 방법은 다른 중형, 대형 소프트웨어 개발 과제에 있어 고 품질 소프트웨어 개발을 위한 개발 관리에 참조 될 수 있다.

Abstract

We summarize, in this paper, what we have learned from the software reliability analysis of a large switching software composed of functional blocks which form software units. To determine the time of management activity related to software reliability growth, we review the process of detection and correction of software failures. Also we apply the two software reliability growth model, Goel-Okumoto and S-shaped model, to estimate the global software reliability growth to a set of failure found during period of the system test. The analysis methods and results can be applied to other large software development projects.

Ⅰ. 서 론

소프트웨어 개발에 100 명 이상의 인력과 1년 이상의 개발기간이 소요되는 과제는 대형과제로 분류된다. 대형 소프트웨어 개발에 있어 고품질 소프트웨어 - 적절히 책정된 예산 한도 내에서 개발 기간 내에 사용자의 요구 조건을 만족시키는 소프트웨어 - 개발을 위한 소프트웨어 신뢰도 관점에서의 개발 관리와 소프트웨어 종합화 및 시험 기술에 관한 경험이 발표되고 있다. 특히 대형 과제의 경우 개발 관리의 측면의 중요성이 강조되고 있다[1]. 또한, 대형 소프트웨어 개발 과제의 경우, 전체 시스템의 기능 실현에 있어 소프트웨어가 차지하는 부분이 커지고 사용자의 만족도가 소프트웨어 품질을 결정하는 주 요소가 되기 때문에 소프트웨어 신뢰도는 최근 크게 강조되고 있다.

소프트웨어 신뢰도는 "주어진 환경 하에서 일정한 시간 동안 프로그램이 고장 없이 동작할 확률"로 정의된다[4].

소프트웨어 신뢰도 공학은 소프트웨어 개발 전체 라이프-사이클에 있어 고장 발견 및 고장 자료의 측정과 분석 방법의 결정과 개발 사이클에 신뢰도 정보의

* 正會員, 韓國電子通信研究院 交換技術研究團
(Switching Technology Division, ETRI)

接受日字: 1997年9月8日, 수정완료일: 1997年11月19日

피드-백을 통하여 개발과 관련된 주요 관리 행위의 시기 -- 기능 추가 시기, 재검토 시기, 종합화 및 시스템 시험 시기, 시험 방법의 변경 시기 등 --를 조정할 수 있는 근거를 제공함으로서 효율적인 개발 관리를 위해 이용된다.

소프트웨어 품질 향상을 위한 개발관리 기술의 발전은 완료된 소프트웨어 개발 과제에서 제기된 문제에 대한 해석을 통해 배운 사실을 비슷한 다른 과제에 적용을 통하여 이루어진다. 대형 소프트웨어 개발에 있어서 신뢰도를 높이기 위하여 개발 관리 기술에 포함되는 것은 소프트웨어의 특성을 고려한 시스템 시험 방법과 전략, 발견되는 고장의 빠른 수정을 위해 통일되고 일괄적인 고장 관리 및 소프트웨어 종합화 방법, 개발 기간 동안 검출한 고장 데이터를 이용한 개발 체계, 전략, 소프트웨어 구조에 대한 재검토 등이 있다.

최근 상용화가 본격화 된 TDX-10A의 모체가 되었던 TDX-10 ISDN 시스템 소프트웨어 개발은 전형적인 대형 소프트웨어 개발 과제이다. 이 대형 소프트웨어에 대한 소프트웨어 신뢰도 관점에서의 소프트웨어 시험 기술과 종합화 등의 개발 과정에 대한 특징과 개발 과정에서의 경험은 대형 소프트웨어 신뢰도 분야의 체계적 연구를 위해 중요한 것으로 인식되고 있다 [2], [3], [4], [10].

사용자 관점에서의 소프트웨어의 기능적 고장은 시스템이 사용되는 시나리오에 따라 라이프-사이클 시험에 의해 발견된다. 대형 소프트웨어의 경우 시스템 시험을 통해 발견할 수 있는 고장은 소프트웨어의 개념 모델과 구조에 따라 분할된 단위 구성요소의 수준에서 결정된다.

연구 대상이 된 소프트웨어의 단위 구성 요소는 완전 분산형 시스템 구조에 따라 기능의 실현을 위해 분할된 기능 블록이다. 이 기능 블록들은 동작 시에는 하나의 프로세서 혹은 다수의 프로세서로 동작하며 시스템 시험 시 소프트웨어 고장 여부를 확인하는 기본 단위가 된다. 이러한 기능 블록들은 소프트웨어의 구조와 기능과의 연관성을 쉽게 이해시키며 고장 발견을 위한 국부화 시험 할 수 있게 하는 등의 시스템 시험과 관련하여 많은 이점을 갖는다.

본 논문에서는 이러한 기능 블록이 시험에서 고장 검출과 확인의 기본단위가 되는 소프트웨어에 대하여 품질 향상을 위한 개발관리에 도움을 주는 시스템 시험 기간에 검출된 고장 데이터를 이용하여 큰 단위에서의 전체 소프트웨어에 대한

- 고장 (혹은 결함)의 발생 이해,
- 소프트웨어 구조적인 문제의 확인,
- 시스템 시험의 전략과 방법

에 관해 살펴봄으로써 기능블록으로 구성된 소프트웨어 특징을 살펴보고, 소프트웨어 고장에 대한 통계적 특성 분석과 소프트웨어 신뢰도 성장 모델을 선택하여 신뢰도 성장에 관해 논한다.

2장에서는 연구 대상이 된 소프트웨어의 고장 자료 분석을 위한 기능 범주와 기능 블록에 대하여, 3장에서는 시스템 시험과 고장 자료의 관리 방법에 대해 설명하고, 4장에서는 고장 데이터에 대한 여러 통계 분석을 수행하고, 5장에서는 소프트웨어 신뢰도 성장을 전체적으로 이해하기 위하여 여러 버전의 국부적 고장 데이터에 S-자형 모델을 이용하여 고장 밀도를 구하고, 이를 전체 신뢰도 성장을 나타내는 Goel-Okumoto 모델과 비교하여 대역적(global)으로 소프트웨어 신뢰도 성장에 대해 논한다.

Ⅱ. 소프트웨어 설명

본 논문의 고려 대상이 되는 TDX-10 ISDN 소프트웨어는 6년에 걸쳐 년간 80 - 120여 명의 인원이 투입되어 개발된 130만 소스라인 규모의 대형 소프트웨어이다. 구현된 기능의 특징에 따라 크게 3개의 버전, N3.3, N3.4, 그리고 N3.5로 구분된다.

TDX-10 ISDN 소프트웨어의 신뢰도 관련 특징을 살펴보기 위하여 기능을 아래의 5개 범주로 구별한다.

- Kernel (K),
- 호처리 (CP),
- 데이터 처리 (DH),
- 운용 (Ad),
- 유지 및 보수 (OM).

Kernel 범주(K)는 타스크 스케쥴링과 메모리 관리, 프로세서 간 통신 제어와 데이터 베이스 관리 등의 기능을 포함한다. 호처리 범주(CP)에 속하는 기능들은 PSTN, ISDN, 패킷 서비스 관련 기능들을 포함하며, 데이터 처리 범주(DH)는 호처리 관련 데이터와 시스템 데이터의 추가/변경/삭제 처리를 위한 기능들을 포함한다. 운용 범주(Ad)는 시스템 하드웨어 혹은 호처리에 관련된 가입자, 트렁크, 번호의 신설/삭제/변경이

나 과금, 통계, 휴먼-인터페이스와 같은 시스템의 운용과 관련된 기능들의 집합이며, 유지보수 범주(OM)는 각종 라인의 시험이나 시스템의 비정상 동작 감시, 감사에 관련된 기능들을 포함한다. 위의 범주들은 기능 분류의 최상위 단계로서, 각 범주는 다시 여러 단계의 하부 범주를 가질 수 있다. 하부 범주의 분할은 기능에 따라 제어 흐름에 바탕을 두고 타스크 분할 기법을 응용하여 구성된다. 분할의 최소 단위는 개발 과정에 사용되는 시스템 개념 모델의 바탕이 되는 소프트웨어 기능 블록이다.

대형 소프트웨어는 규모가 방대하기 때문에, 기능블록은 분리 컴파일 될 수 있어야 하며, 온-라인 변경이나 고장의 확인 등을 위하여 소프트웨어 개발 단위와 관리 단위로 적절히 유지되어야 하며 부분적인 로딩이나 패키징 단위로 제공되어야 한다. 본 논문에서 고려되는 소프트웨어는 소프트웨어 구성 요소인 기능 블록은 개발과 관리 및 패키징, 그리고 로딩의 기본 단위로 이용되며 사용자 요구 사항을 수용한 850개 시스템의 기능이 140개의 블록으로 구성된다.

소프트웨어 개발은 사용자 요구사항을 여러 단계에 걸쳐 기능을 추가함으로써 실현시키는 방법이 이용된다. 추가되는 기능은 새로운 블록의 개발과 동시에 기존 블록의 변경이나 기능 확장을 요구한다. 한 블록은 보통의 경우 한 개발 팀에 의해 개발되므로 블록은 또한 개발의 기본 단위로서 간주된다. 기능 범주 별 소프트웨어 블록 수와 규모, 사용자 요구 기능은 표 1과 같다.

부록(Appendix)에 140개 블록의 범주 별 분류를 보인다.

표 1. 소프트웨어 블록 수 및 규모
Table 1. No. of block and size.

Categories	Number of Block	Size(KLOC)	User Functions
CP	45	390.7	270
DH	14	171.9	33
Ad	44	315.3	290
OM	29	240.7	241
Kernel	8	217.5	-
Total	140	1336.1	834

Ⅲ. 시스템 시험과 고장 데이터 관리

1. 시스템 시험

시스템 개발 기간 동안 단계 별로 단위, 종합, 시스템 시험이 수행된다[11]. 여러 시험 중에서 시스템 시험은 소프트웨어 신뢰도 추정이나 예측을 위해 매우 중요하다. 그 이유는 시스템 시험에서는 소프트웨어가 실제 사용되는 환경과 유사한 상황에서 시험을 전체적으로 수행할 수 있기 때문이다. 시스템 시험은 기능의 정상 동작 여부를 확인하는 시험으로써 주로 요구 사항이나 시스템 동작 규격 만족 여부를 확인하는 데 이용되는 블랙 박스 시험이다. 시스템 시험 기술은 신뢰도 성장 분석에 이용되는 고장 데이터 수집과 소프트웨어 신뢰도 공학에서 요구되는 기본 사항을 고려하여 결정되어야 한다.

논의 대상이 되는 소프트웨어의 구조는 실행 단계의 최소 단위가 기능 블록으로 구성되어 사용자 기능들은 다수 블록의 연동으로 이루어지므로, 시스템 시험에서 확인할 수 있는 고장의 확인 단위는 기능블록이다. 시스템 시험은 개별 블록 단위로 기본적인 시험이 끝난 후, 모든 블록들이 시스템 내에서 연동되어 진 형태로 실행된다. 블록의 연동으로 기능이 이루어지기 때문에 시스템 시험은 검증과 종합을 위해 특히 중요하다. 시스템 시험에 이용되는 시나리오는 블럭 간 시그널에 의한 제어 흐름에 따라 빈도수가 높은 순서의 조합으로 구성된다. 여기에서 시나리오는 소프트웨어 신뢰도 공학에서 말하는 운용 프로파일이다.

시스템 시험은 잘 정의된 고장 개념에 의거하여 일반적인 시스템 시험 방법으로 진행된다[11]. 소프트웨어 신뢰도 평가를 위한 정확한 데이터를 얻기 위하여 수정된 패키지로 반복 시험이 수행되며, 그 기간에 따라 1주일 단위로 실시되는 단기 시험과 3개월 단위로 실시되는 장기 시험으로 구분된다. 단기 시험은 특정한 기능의 정상 동작 여부에 대한 시험을 위주로 종합 시험으로 수행되며, 장기 시험은 고정된 한 버전의 소프트웨어에 대하여 기능과 품질에 대한 인증을 위해 모든 기능이 연속적으로 시험된다.

또한 시스템 시험에서 고장은 사용자 요구 사항과 시스템의 기술 규격에 의해 만들어진 시험 절차서에 따른 시험 결과가 기대 결과와 일치하지 않을 경우 비정상 동작으로 분류되어 원인 분석 단계를 거치게 된다. 고장의 수정 후, 기능의 정상 동작 여부는 회귀 시험으로 검증되며 파생된 고장 존재 여부 또한 시험된다. 시스템 시험의 전략과 방법은 피이드-백된 소프트웨어 신뢰도 해석 결과에 따라 변경되거나 실행 시기 등이 결정된다.

2. 소프트웨어 고장 관리

시스템 시험에서 검출된 모든 이상 동작은 발생 상황과 시험 조건과 결과가 기록된다. 이 자료는 고장의 진위 여부와 원인 및 내용을 밝히기 위해 시스템 고장 검토 회의에 의해 검토되고, 검토된 고장 보고 중 원인이 명확한 것은 고장으로 분류된다. 시험 기간 동안 발견되는 소프트웨어 고장은 고장 데이터 관리시스템(FMS : Failure Management System)에 의해 종합 관리된다. FMS를 이용하여 시스템 시험을 통해 검출된 고장에 대한 버전 별, 위치별, 시험 대상 항목, 재현성 여부, 발생 일시 및 상황 등 전반적인 고장 현황이 일괄적으로 기록, 관리된다. 발생한 고장에 대한 명확한 진단 및 조치를 위한 고장 분석은 원인분석(Cause Analysis) 기법과 내용분석(Semantic Analysis) 기법이 적용된다. 고장이 발생하면 발생 위치, 원인 및 내용 별로 분류된다. 고장 발생 위치는 고장의 원인이 된 결함이 위치한

- 범주/기능/블록

으로 표현되고, 원인 및 내용에 따라 다시 다음과 같은 사항으로 세분된다.

- 소프트웨어 구조(Structure) -- 상황을 고려치 않은 경우, 데이터 구조 설계 상의 결함,
- 시그널(Signal) -- 입출력 시그널의 규격에 의한 결함,
- 제어/선언(Control/Assignment) -- 제어에 대한 코딩 결함,
- 데이터 생성(Data Generation) -- 데이터 생성, 초기화 결함,
- 패키지 -- 개발환경 혹은 패키징에 의한 결함.

이 분류는 소프트웨어 설계와 프로그래밍 두 부분에 대한 결함 발생 원인과 내용 분석을 위주로 생각하기 위한 것이다. 연구 시각에 따라 결함 내용 분석은 다양한 형태로 분류될 수 있다 [3], [5], [6].

소프트웨어 수정은 소프트웨어 변경이력관리시스템(CRMS : Change Request Manag -ement System)에 의해 관리되며 소프트웨어 수정 요구(CR : Change Request)는 소프트웨어 형상 관리 위원회(SCRC : Software Change Request Committee)에 소속된 각 분야 전문가에 의해 검토되어 고장 원인이나 고장 내용 분석 등을 거쳐 소프트웨어 변경이나 결함의 수정이 관리되고, 이 결과는 소프트웨어 버전 관리에 반영되어 종합적이고 체계적으로 관리가 이루어 진다.

고장을 수정하기 위해 변경된 소프트웨어는 기능블록 별로 관리되며 이것들은 CRMS에 의해 변경 이력으로 기재되어 개발 관련 사항들을 확인하고 검증하고 소프트웨어 신뢰도 평가를 위한 기본 데이터로 이용된다[5].

Ⅳ. 고장 데이터 통계

이 절에서는 개발 기간 동안 검출된 소프트웨어 고장에 대하여 시간과 버전 별로 분포를 살펴보고, 고장의 내용 분석을 통한 결함이 많이 발생한 블록의 소프트웨어 특성과 고장 해결에 소요된 시간 등에 관해 살펴본다. 이 내용은 시스템 시험의 시나리오 결정과 시험 방법의 선택에 주요한 자료로 이용된다. 약 5년 동안의 시스템 시험에서 고장으로 보고된 1989개의 리포트 중, 소프트웨어 고장으로 최종 확인된 수는 1780개이다.

1. 버전 별 소프트웨어 고장 분포

소프트웨어 N3.3은 시범 버전 N3.2에 기능 추가가 실현된 패키지로 실용 시험 및 현장 적용 시험이 시도된 첫 패키지이고 N3.4는 종합적인 기능이 추가된 패키지이며 N3.5는 N3.4에서 발견된 고장의 수정 버전으로 상용 서비스를 위한 현장 배포용이다. 기능의 중요도에 따라 각 버전에 대하여 수행된 시스템 시험의 기능 범주 별 우선 순위는 아래 표 2와 같다.

표 2. 시험 우선 순위
Table 2. Priority of test

Versions	Priority
N3.3a	CP > DH > Ad > OM
N3.3b	Ad > OM > CP > DH
N3.4a	CP > Ad > OM
N3.4b	Ad > CP > OM

전체 발생 고장과 시스템 시험 시간과의 관계는 그림 1과 같다. 여기에서는 월 단위의 고장 수를 사용한다. 소프트웨어 개발 기간은 1년 단위이지만 실제 시험에 순수하게 소요되는 시간은 9개월 정도이므로 앞으로 고장 자료를 생각할 때는 항상 이 단위를 사용한다.

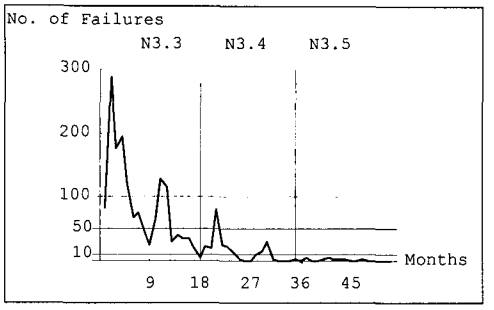


그림 1. 시험 기간 동안 고장 수의 분포
Fig. 1. Distribution of all failures.

버젼 별 고장 발생을 비교하기 위하여 시스템 시험 시작 시기를 동일하게하여 발생한 고장 수를 비교해 보면 그림 2와 같다. 그림 2로 알 수 있는 것은 시스템의 기능이 여러 블록의 연동으로 이루어지기 때문에 새로운 기능 추가에 따른 새로운 버젼에서 초기 고장 발생은 많으나 주요 기능이 블록 중심으로 구현되어 있어, 시스템 시험을 통해 고장 위치 탐지와 디버깅이 용이하며 단 기간 내에 많은 고장의 발견과 수정을 할 수 있는 점이 블록으로 구성된 소프트웨어의 장점임을 알 수 있다. 이 점을 이용하여 시스템 시험 시에 기능 범주에 따라 블럭 별로 시험의 강도에 대한 가중치를 부여할 수 있어 고장 발생이 높은 블록에 대하여 시험의 강도를 조정할 수 있다.

실제 N3.4 패키지의 경우, 시험 시작후 고장 발생이 3월(전체 21 번째 월)과 12월(30 번째 월)에 피크가 되는 것을 볼 수 있다(그림 1 참조). 이는 21번째 월에 추가된 기능에 대한 블록의 시험 강도를 높인 상용 시험에 의한 것이고, 이 후 30번째 월에는 온-라인 운용 프로파일을 변경한 현장 운용 시험에 의한 것이다.

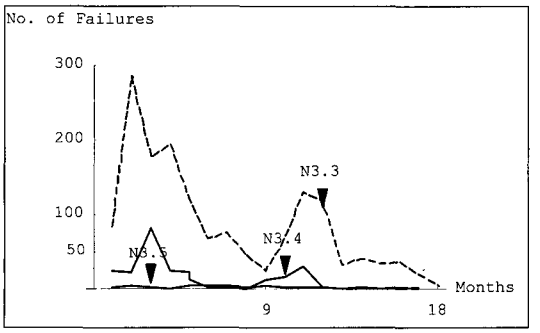


그림 2. 버젼 별 고장 수의 분포
Fig. 2. No. of failures per versions.

2. 소프트웨어 기능 범주 별 고장 분포

II 장에서 제시한 범주에 대한 버젼 별 고장의 분포는 소프트웨어 고장으로 인지된 1780개의 고장을 기준으로 하면 표 3과 같다

표 3. 범주 별 소프트웨어 고장 분포
Table 3. Distribution of software failure.

Versions Categories	CP	DH	Ad	OM	K	%
N3.3	25.3	3.7	42.0	25.7	3.3	100
N3.4	24.4	4.0	42.2	26.1	3.3	100
N3.5	33.0	0.5	42.8	21.1	2.6	100

모든 버젼에 걸쳐 Ad 범주가 많은 고장을 가지는 이유는 이 범주가 시스템 운용에 관련된 CP, DH, OM 기능들에 이용되는 기본 데이터 설정과 시스템의 모든 상태 변화 관리를 담당하는 전체 기능의 약 60%의 기능을 포함하기 때문이다. 표 3를 보면, N3.3, N3.4의 경우는 각 기능범주 별 비슷한 고장 분포를 보이는 반면, 소프트웨어의 현장 적용 후, 즉, N3.5에 대하여, 사용 빈도가 급격히 증가하는 CP 범주에서 N3.3, N3.4에 비해 상대적으로 많은 고장이 발생하는 것을 볼 수 있다. 이러한 현상은 실제 운용 상황에 의한 것으로 시험 환경에서 발견되지 않던 고장이 발생하는 것을 볼 수 있다.

3. 블록 대 결함

하나의 소프트웨어 고장 해결을 위해 수정되어야 하는 블록의 수에 대한 통계는 블럭 간 상호 결집도에 관계되는 지표를 제시한다. 표 4는 결함 수정을 위해 수정 되어진 블록 수의 통계 자료이다. 1-2 블럭의 수정으로 해결된 고장은 88% 정도로 기능의 실행 시블럭의 독립성이 최대한 유지되고 있음을 알 수 있다. 그러나 나머지 많은 수정이 요구된 블록은 소프트웨어 구조상, 연동의 중심에 위치하는 몇개의 블록으로 타 블록의 수정 시에도 영향을 받고 있는 것을 알 수 있다. 시스템 동작 시에 연동이 많은 블록에 고장의 발생이 높다는 직관적, 경험적 사실을 확인할 수 있다.

표 4. 고장 당 수정된 블록 수
Table 4. No. of modified Blocks.

Number of modified Blocks	CRs(%)
1	1295(73%)
2	265(15%)
3	111(6%)
> 4	109(6%)

시스템 기능이 여러 블록의 연동으로 이루어지므로, 하나의 고장이 여러 블록의 수정을 거쳐 해결되는 경우가 많으므로, 소프트웨어 고장 해결을 위해 유발되는 작업의 양과 잠재적 결함을 많이 포함하는 블록의 확인을 위해 여기에서 소프트웨어 수정 요구 데이터를 이용하여 CR과 블록 수와의 관계를 살펴보기로 한다. 1780개의 소프트웨어 고장의 해결을 위해 30번 이하의 수정을 필요로 한 블록이 약 81%임을 알 수 있다 (그림 3 참조).

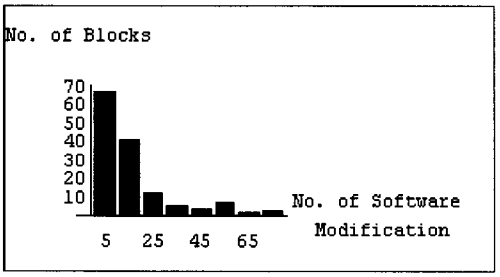


그림 3. 블록 수와 CR 수의 분포
Fig. 3. No. of blocks vs no. of software modifications.

블록의 규모와 고장수와의 관계는 그림 4와 같다. 그림 4에서 보는 것처럼, 블록의 크기(단위 : KLOC)와 결함 수는 선형적 관계를 보이지 않는다. (이 그림에서 실선은 전체 소프트웨어 규모에 대한 고장 수의 평균으로 약 1/1000 결함/라인을 나타낸다.) 이는 교환 소프트웨어가 매우 복잡한 상태 천이를 갖는 프로그램임을 나타내는 것으로 크기 보다 알고리즘의 복잡성과 데이터 변경에 민감한 연동을 갖는 블럭이 보다 많은 결함을 가지는 사실로부터 설명된다. 이 사실은 소프트웨어 디자인 단계에서 블록 분할에 필요한 자료로 활용된다.

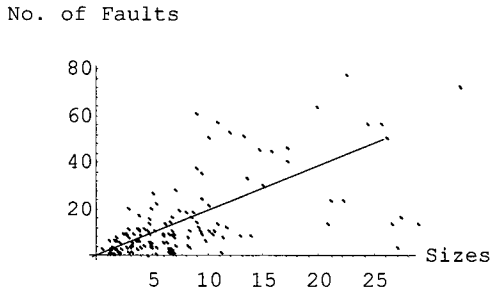


그림 4. 블록 규모와 소프트웨어 수정 수
Fig. 4. Block size vs no. of software modifications.

4. 고장 내용과 해결 시간 분포

IV장 1절에 보인 바와 같이 소프트웨어 고장은 새로운 기능 추가시 마다 많이 확인된다. 고장 발생이 많은 블록의 확인을 위해 결함의 내용을 분석하면 고장이 많은 블록은,

- 상태 변화에 따른 복잡한 제어를 포함하는 블록,
- 수정을 위해 자료 구조의 변경을 포함하는 블록,
- 예외 처리가 많은 블록,
- 시스템 상태 변수의 영향을 받는 블록,
- 한 입력 시그널에 포함되는 변수가 많은 시그널을 갖는 블록,

등의 순서이다. 이 사실은 연구되는 소프트웨어도 역시 잘 알려진 바와 같이 소프트웨어 신뢰도는 소프트웨어의 구조와 개발 단위 할당에 많은 영향을 받는 사실을 보여주고 있다[4]. 소프트웨어 신뢰도 개념의 디자인 단계에서의 도입의 중요성을 보여주는 것이다.

소프트웨어 고장 발견 시점부터 해결 완료 시점까지 소요된 시간에 대한 분포는 고장 발견 후, 소프트웨어 수정에 대한 효율의 분석과 현장에서 운용 중에 보고된 문제점을 해결하기 위한 유지 보수 노력을 예측하는 데 응용된다. 고장 해결에 소요되는 시간은 결함의 내용과 관련성이 높은 것으로 설계 단계에서 미 고려되거나(missing) 불완전한(incomplete)것은 해결에 많은 시간이 소요되나 빈번하지는 않다. 고장 해결에 소요된 시간 통계는 아래 그림 5와 같다.

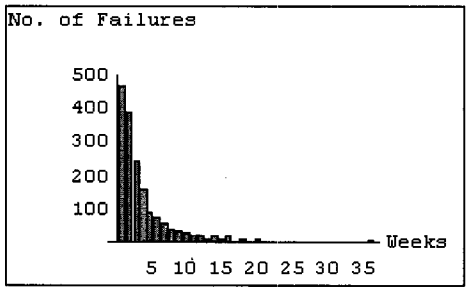


그림 5. 소요 시간 분포
Fig. 5. Distibution of correction times.

소프트웨어 고장 내용을 살펴보면 65% 정도가 블록 내의 제어/선언의 결함으로 인한 고장으로 분류되는 데, 이는 고장 해결에 한달 정도의 소요 시간이 필요한 고장이 70%라는 사실(그림 5 참조)과 일치한다.

그러나 구조와 시그널과 관련된 결함의 수정은 데이

터 구조 변경과 재시험 등을 거쳐야 하므로 수정과 수정의 올바름 여부를 확인하는데 많은 시간이 소요된다.

V. 소프트웨어 신뢰도 평가

이 절에서는 소프트웨어 신뢰도 성장 경향 분석과 운용 중의 소프트웨어 고장율을 추정하기 위하여 고장 데이터에 대하여 소프트웨어 신뢰도 성장이론을 응용하여 분석한다. 소프트웨어 신뢰도 성장에 대한 이론과 모델은 [3], [4]에 소개된 내용을 참조로 한다. 분석고자 하는 소프트웨어 고장 자료는 시스템 시험 과정에서 검출된 고장들로 시험 기간 동안 수정되기 때문에 NHPP (Non-Homogeneous Poisson Process) 모델에 기반을 둔 신뢰도 성장 모델이 선택된다.

1. 소프트웨어 신뢰도 성장 모델

소프트웨어 신뢰도 성장 모델은 지수형과 S-자형 성장 모델을 생각하기로 한다. 이유는 두 모델이 분석하고자 하는 소프트웨어 의 고장 검출을 위한 시험의 방법과 관련된 조건을 만족시키는 가정 - 전체 고장은 유한하며, 시험에서 발견되는 고장은 서로 독립적이며 그 발생은 NHPP를 따르며 고장은 수정된다 - 에 따라 만들어진 모델이기 때문이다. 지수형 모델은 단위 시간 당 결함의 검출 수, 즉, 고장 발견율이 시험 시작점에서 최대가 되고 이후 지수 함수적으로 감소한다는 가정으로 고장 발생을 설명하는 모델이고, S-자형 모델은 임의의 시험 기간이 경과한 시점에서 최대가 된 후에는 지수형과 동일하게 감소한다는 가정에 근거를 둔 모델이다. 소프트웨어 신뢰도 성장 모델은 시간에 대한 누적 고장수의 관계로 표현되며, 시험 시작 전 소프트웨어에 잠재하고 있는 고장수, 혹은 최종 검출 기대 고장 수를 유한한 값 a로 표시하고, 시험에서의 평균 고장 검출율을 b라 하면

지수형 성장 모델(G-O 모델) [3], [4]

G-O 모델은 시간 t 에서의 누적 고장수 N(t)는

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

로 주어지고,

o S-자형 신뢰도 성장 모델 [7], [8], [9]

이 모델에서의 시간 t에서의 누적 고장수 N(t)는

$$N(t) = a(1-(1+bt)e^{-bt}),\ a,b>0 \qquad (2)$$

로 주어진다.

시험에서 얻은 실제 고장 데이터를 이용하여 maximum likelihood estimation 방법으로 a, b 값을 추정하고, 이렇게 얻어진 a, b 값을 이용하여 실제 고장 데이터의 시간에 대한 누적 고장 수의 변화를 선택한 모델에 따라 식 (1) 또는 (2)로 fitting함으로써 소프트웨어 신뢰도 성장에 관한 해석을 수행한다.

2. 신뢰도 분석

신뢰도 평가를 위해 적용한 모델은 국부적인 각 버젼의 고장 자료에 대하여는 S-자형 모델이고, 전체 고장 자료에 대하여 대역적으로는 G-O 모델이다. 국부 고장 데이터에 S-자형 모델이 이용되는 이유는 새로운 기능이 추가되어 버젼이 바뀌면 시험 방법이 어려워지고 많은 경우의 조합으로 구성된 시험이 수행되어야 하므로 시험이 계속됨에 따라 고장 발견에 소요되는 시간이 무시할 수 없는 만큼 커지기 때문이다. 그러나 어느 정도 시험이 계속 되면 고장 발생에 대한 시험 방법에 익숙해짐에 따라 고장 검출 효과가 높아지게 되는 현상을 반영하기 위함이다. 먼저 S-자형 모델을 적용한 각 버젼 별 고장에 대한 식 (2)의 a, b 값은 표 5와 같이 계산된다.

표 5. S-자형 모델의 a, b 추정치
Table 5. Estimated values of a and b of S-shaped model.

버 젼 번 호	a(추정치)	b(추정치)
N3.3a	1118.14	0.552831
N3.3b	490.311	0.641673
N3.4a	188.586	0.697083
N3.4b	62.1272	0.937313

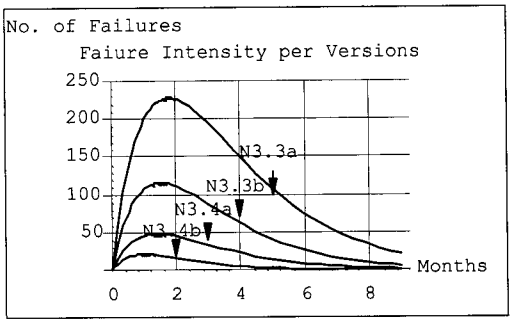


그림 6. 버젼 별 고장 밀도
Fig. 6. Failure Intensity per version.

표 5의 추정값 a, b를 이용한 식 (2)의 t에 관한 도함수로 주어지는 밀도함수의 그래프는 그림 6과 같다.

그림 6에서 보는 바와 같이 각 버젼의 고장 밀도를 비교하면 버젼이 높아질수록 소프트웨어는 지속적인 신뢰도 성장을 보이고 있는 것을 알 수 있다.

버젼 별 고장 수의 합은 표 6과 같다.

표 6. 버젼 별 전체 고장 수

Table 6. No. of total failures per versions.

Versions	N3.3a	N3.3b	N3.4a	N3.4b	N3.5	N3.5-field
Number of Failures	1072	482	186	62	21	4

표 6에서는 인증, 현장적용, 인수시험에서 발견된 고장을 모두 포함하여 전체 수는 1827개이다. 이 고장 데이터를 이용하여 전체 소프트웨어의 신뢰도를 G-O 모델을 적용하여 분석하면, 즉, 식 (1)에 따라 a, b의 값을 구하면 a = 1842.04, b = 0.101597를 얻는다. 위 a, b값을 이용한 식(1)의 전체 누적 고장 수 그래프는 아래 그림 7과 같다.

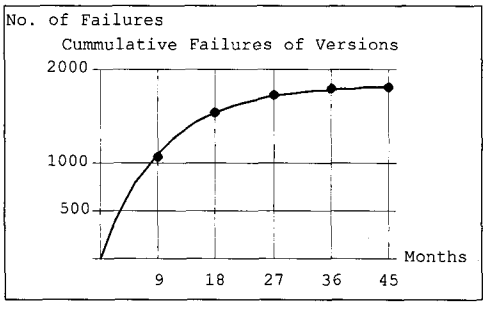


그림 7. 전체 누적 고장 (G-O 모델)

Fig. 7. Cumulative failures.

여기에서 점은 실측 데이터이고 실선은 G-O 모델의 추정치를 나타낸다. 대역적 고장 밀도 함수 h(t)는 (1)의 미분으로

$$h(t) = 1842.08*0.101597\,e^{-0.101597\,t} \qquad (3)$$

와 같고, 고장밀도 h(t)의 그래프는 그림 8과 같다.

이를 국부적인 데이터를 이용하여 얻은 고장 밀도 함수의 시간에 대한 진전과 비교해 보면 그림 9를 얻는다. 그림 9는 부분적인 버전 별 소프트 신뢰도 성장과 전체적인 성장의 모양을 동시에 보여준다(그림 1과 비교).

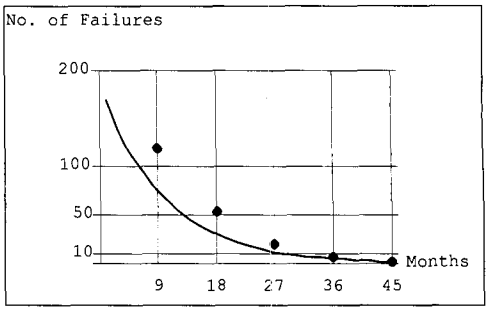


그림 8. 전체 고장 밀도(G-O 모델)

Fig. 8. Failure Intensity of G-O model.

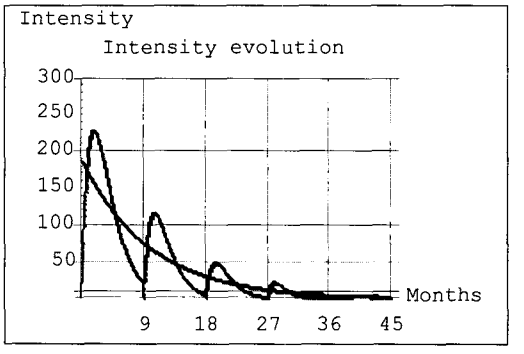


그림 9. 고장 밀도의 진전

Fig. 9. Intensity evolution.

식 (3)을 이용하여 다음 9개월 후, 즉, t = 54월 일 때 예측되는 고장율은 약 0.77452이다. 이 결과는 실제 현장에 적용하여 얻은 4개/9개월 = 0.444의 고장율과 충분히 근사함을 알 수 있다. 이를 시간으로 환산하면 운용중의 발생 고장율은 약 $1.007*10^{-4}$ 고장/시간 정도로 예측된다. 이 소프트웨어 신뢰도 예측 결과는 다른 교환 소프트웨어 신뢰도 연구의 결과와 비교될 수 있다 [2], [3], [4].

VI. 결 론

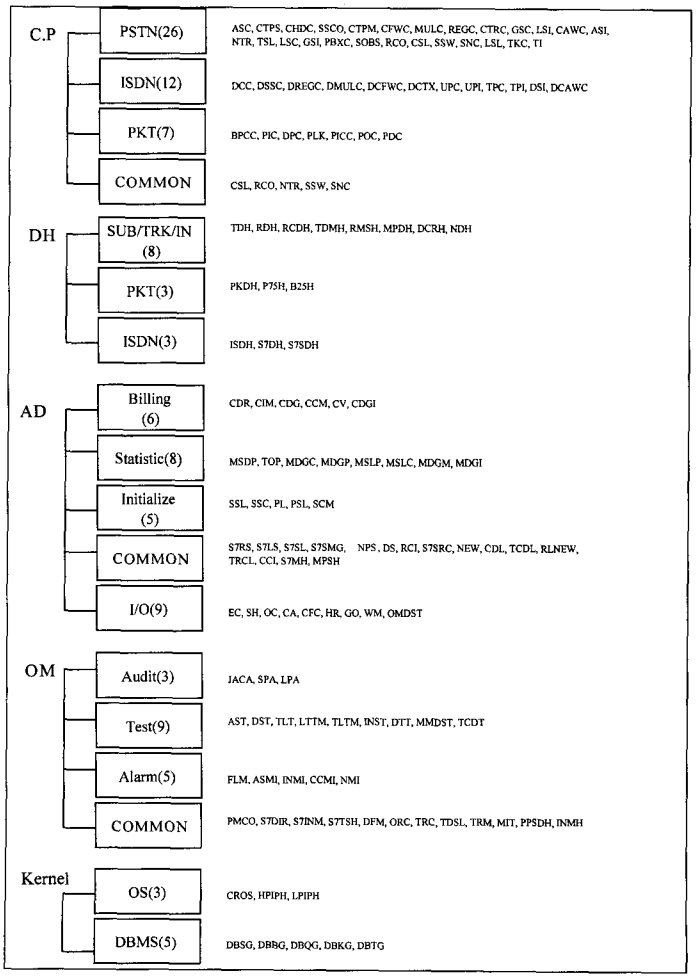
본 논문에서는 대형 소프트웨어 개발에 있어서 시스템 시험에서 얻어진 소프트웨어 고장 자료를 이용하여 소프트웨어 개발 및 실행의 단위가 되는 블록의 신뢰도 관점에서의 특징을 고찰하고, 소프트웨어 신뢰도 성장을 위한 시스템 시험 방법에 대해 논하고, 두 가지 전형적인 지수 형, S-자형 소프트웨어 신뢰도 성장 모델을 응용하여 소프트웨어 신뢰도를 예측하였다. 소프트웨어 구성 요소로서 기능 블록을 갖는 특수한 대형 교환 소프트웨어 개발에 있어서 시스템 시험 기

간 동안 얻어진 고장 자료의 분석을 통해 소프트웨어 신뢰도 성장의 한 패턴(그림 9참조)를 보였다. 이 결과는 버젼 별로 기능 추가가 실현되는 기능 블록으로 구성된 대형 소프트웨어의 신뢰도 성장의 한 사례로 제시될 수 있다.

소프트웨어 개념 모델의 전제가 되는 기능의 분할에 의한 블록은 개발과 시험을 위한 단위를 제공하고, 시스템 시험 시 고장의 발생 확율이 높은 블록에 대하여 시험의 강도를 조정하게 하여 단 기간 내에 많은 고장을 비교적 정확히 찾을 수 있는 장점을 가지고 있음을 다양한 통계 자료를 이용하여 보였다. 또한, 블록으로 구성된 소프트웨어에서 기능이 블록 간 연동으로 이루어지기 때문에 블록의 단위 시험으로 블록 자체 코드의 검증 이 어렵기 때문에 종합시험, 시스템 시험이 신뢰도 향상을 위해 중요한 수단이 됨을 확인하였다. 그러나 사용자 기능의 분할에 의한 블록으로의 변환에 있어서 피할 수 없이 제어가 집중화되는 블록이 존재하게 되고 또한 이러한 블록들은 개발 전 기간 동안 항상 수정을 감수해야 하며, 고장의 수정이나 기능 추가를 위해 하나의 블록이 수정된 경우 타 블록과의 연동 때문에 항상 회귀 시험이 수행되어야 하는 등의 문제도 많이 제기된다. 앞으로 이런 문제의 해결을 위한 소프트웨어 신뢰도 할당과 관련된 소프트웨어 구조 연구가 필요하다.

소프트웨어 고장 발생과 수정에 대한 개발 단위 별 통계와 신뢰도 성장 모델의 응용 사례에 대한 연구 결과는 다른 대형 소프트웨어의 개발에 있어 소프트웨어의 구성 요소 분할과 신뢰도 할당, 신뢰도 평가 연구에 대한 참고 자료로 활용될 수 있다.

Appendix



()안의 숫자는 소프트웨어 블록수를 의미함

참 고 문 헌

[1] L. Bernstein, "Software in the Large," *AT&T Technical Journal,* pp. 5-14, Jan/Feb. 1996.

[2] Kaaniche M., Kanoun K, "Reliability of a Commercial Telecommunications System," *ISSRE'96,* pp. 207-211, 1996.

[3] Michael. R. Lyu, "Handbook of Software Reliability Engineering", McGraw-Hill, 1995.

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

[5] 이재기외, "대형 교환기의 시스템 신뢰도 평가," *'95 통신학회 추계학술대회 논문집,* pp. 537-540, 1995

[6] 유재년, 이재기, "교환 시스템 고장 유형 분석과 신뢰도 평가," *'94 전자공학회 추계학술대회 논문집,* Vol. I, pp. 495-497, 1994

[7] S. Yamada and S. Osaki, "Nonhomoeneous error detection rate models for software reliability growth," *Stochastic Models in Reliability Theory,* pp 120-143, Springer-Verlag, Berlin, 1984.

[8] Yamada. S, et al, "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.

[9] Shigeru Yamada and Hiroshi Ohtera, "Software Reliability : Theory and Practical Application," *SE series,* pp. 135-196, Soft Research Center, Feb. 1990.

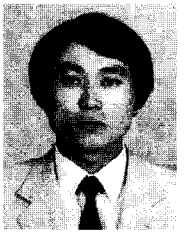
[10] Jelinski Z. and Moranda P., "Software Reliability Research," *Statistical Computer Performance Evaluation,* W. Freiberger Ed., New York, Academic Press, pp. 465-484, 1972.

[11] B. Beizer, "Software Testing Techniques," Van Nostrand Reinhold, New York, 1990.

저 자 소 개

劉 宰 年(正會員)

1981년 2월 연세대학교 수학과 졸업 (이학사). 1983년 2월 서울대학교 대학원 수학과(이학석사). 1985년 2월 한국전자통신연구소 입소. 1996년 8월 포항공과대학교 대학원 수학과(이학박사). 1985년 - 현재 한국전자통신연구원, 교환기술연구단, ISDN호처리연구실, 선임연구원 재직중. 주관심 분야 : 응용수학 관련 분야



李 在 起(正會員)

1985년 2월 서울산업대학교 전자공학과 졸업(공학사). 1989년 5월 청주대학교 대학원 전자공학과 졸업(공학석사). 1983년 3월 - 현재 한국전자통신연구원 교환기술연구단, S/W종합검증실 선임기술원 재직중. 주관심분야 : Software Testing, Software Quality