차별화된 소프트웨어 시험을 위한 시험항목 우선순위 조정

정회원　이 재 기*, 이 재 정**

Test item prioritizing metrics for a selective Software Testing

Jae-ki Lee*, Jae-jeong Lee**　***Regular Members***

요　약

다양한 소프트웨어 기능들에 대해서 활발한 시장의 요구사항에 부응하기 위해서는 주어진 납기에 시스템시험을 마쳐야 한다. 특히, 사용자나 개발목표 시스템의 주요 핵심기능과 밀접한 주요 고장들은 시스템시험에서 반드시 검출, 제거되어야 한다. 대부분 제안되고 있는 시스템시험 방법은 효율적이고 가격 경쟁력을 갖춘 선택적인 시험 방법이 아닌 일상적인 방법이다. 이러한 방법들은 시스템 개발 초기나 단기간(short-term)의 효과적인 시험에 응용할 수 없으며, 새로운 기능의 추가에 따른 시험 준비에 많은 비용이 수반되므로 효과적인 시험이 되지 못한다. 이러한 문제를 극복하기 위한 새로운 시험방법으로 시험수행의 핵심적인 역할을 수행하는 선택적인 소프트웨어 시험이 필요하다. 선택적인 시험방법은 3가지 정보를 조합하여 시험항목 순위를 결정하는데 즉, 사용빈도, 시나리오 복잡도, 고장강도 등이다. 이 정보를 사용하여 시험을 수행함으로써 시스템의 치명적인 고장을 찾는데 활용함으로써 보다 효율적인 시스템 시험을 수행할 수 있다. 본 논문에서는 주요기능과 관련된 심각한 오류와 시스템의 치명적인 결함을 찾는 선택적인 소프트웨어 시험 방법에 대해 제안된 방법을 적용, 그 결과를 검증한다.

Key Words : 소프트웨어 시험, Selective testing, Test item priority, Fault, Failure, Traffic Manager

ABSTRACT

The system test was accomplished in delivery time for a suitable of various requirements at the software market. Especially, critical faults must be detected and removed for a close main functions and users against target system. In generally, proposed test methods are executed with a calendar time, not a competitive and effectiveness method as selective software testing. These methods are inapplicable to short term test or early system development stage. Moreover, it's accompanied by heavy cost. Overcoming of these problems, must attempted to new software test method role of core function in the system test. Selective software testing method is decided to mixing with the three-information such as a frequency, complexity of use scenario and fault impact. Using this information, searching a fatal error and usefully system test for an executed test scenario. In this paper, we have proposed new test method and verified testing results for the detection of critical faults or search a fatal errors with a system main function.

Ⅰ. 서　론

최근 소프트웨어 개발에 있어서 구현된 소프트웨어 기능에 따라 2가지 경향을 띤다. 첫째는 소프트웨어 구조가 복잡해지고 규모가 증가하게 된다[14]. 둘째는 소프트웨어 개발기간과 자원이 제한된다는 것이다. 특히, 소프트웨어 시스템 인도시기에 대한 압박으로 개발기간이 단축된다는 것이다. 반면에 소프트웨어 품질을 보장하기 위한 소프트웨어 시험은 효율적으로 행해져야 한다는 점이다.

* 한국전자통신연구원 솔루션개발프로세스연구팀 (jkilee@etri.re.kr),　** 한국전자통신연구원 품질보증팀 (jjlee@etri.re.kr)
논문번호 : KICS2007-06-244,　접수일자 : 2006년 5월 8일,　최종논문접수일자 : 2006년 12월 07일

시스템 시험은 크게 3가지로 구분되는데 단위시험(unit test), 종합시험(build test), 시스템시험(system test)으로 그중에서도 시스템 시험은 사용자의 요구사항과 직접적인 관계를 갖는 최종 목표시스템의 기능을 검증하는 것이다[4].

소프트웨어 구조의 복잡도 증가와 사용자의 소프트웨어에 대한 높은 품질의 요구는 더욱더 시스템에 대한 소프트웨어 시험의 효율성과 창의성을 요구한다. 이런 요구를 수용하기 위해서는 선택된 시험 항목과 축소된 시험항목에 대한 시험이 철저히 수행되어야 한다[15]. 이러한 접근방법은 시험기간의 단축에 매우 효과적이며, 만약 사용자 관점의 시험항목이 검토되지 않으면 시스템의 주요 기능들에 대해 소프트웨어에 남아있는 고장들이 시험되지 않아 소프트웨어의 품질이 저하된다. 그러므로 선택적 시험에서 시험항목의 선정은 매우 중요하다[16].

본 연구의 대상은 FTTH(Fiber to the Home) 실현을 위한 핵심장비의 하나인 sOLT(standard Optical Line Termination) 시스템으로 성공적인 임베디드(embedded) 소프트웨어 개발의 사례이다[18].

이 시스템의 소프트웨어는 다양한 기능과 제한된 개발기간을 가지고 추진되었기 때문에 시스템 시험의 역할은 제한된 기간 내에 소프트웨어의 품질을 향상시키는데 매우 중요했다. 비록 소프트웨어가 다양한 기능을 포함한다고 하더라도 사용자가 동일한 내용으로 소프트웨어 내에 있는 모든 기능들을 사용하지 않기 때문이다. 특수한 경우에는 여러 기능들은 전혀 사용되지 않는 경우가 발생하며, 경우에 따라서는 이러한 기능들은 시스템 동작 관점에서 매우 중요한 역할을 할 수 가 있다. 이러한 사실로 볼 때 소프트웨어 기능들은 사용빈도와 중요성이라는 2가지 관점에서 분류할 수 있다. 결과적으로 사용빈도와 중요성에 대한 기능들에 대해 신중한 시험항목의 선택이 필요하다.

우선 앞에서 언급한 시험 항목 축소 기술을 검토해보면 아래와 같이 2가지 관점으로 요약할 수 있다.

◎ 새로운 시험기술의 요약
(1) 사용자 관점의 테스트 항목 선택
(2) 회귀시험(regression test)에서 시험항목의 축소

사용자 관점에서는 사용자의 습성과 사용(운용) 환경에 대한 고려가 있어야 한다. 시스템 성능 평가(performance evaluation)를 위한 시험항목 선택과 수행은 Musa et al. 이 제안한 운용 프로파일(operational profile)을 사용한다[12,13]. 운용 프로파일은 사용자의 다양한 운용환경 가정을 기반으로 한 기술로 단순한 마코프 모델(unified Markov Model)을 적용하여 사용자의 운용 시나리오와 절차에 대한 만족여부를 분석한 시험방법이다.

운용 프로파일이나 마코프 모델은 제어 흐름(control flow)이나 동작중에 발생할 확률을 계산하기 위한 실행이나 시험규격 정보로 사용된다. 개발 시스템의 운용에 대한 모델링은 실제 시스템 개발에 많은 영향을 주며 이런 이유로 시험 활동에 요구되는 많은 시간은 제한된 시스템 개발기간에 적용이 어렵다.

회귀시험(regression test)을 위한 시험 항목의 축소에 대한 연구는 많이 수행되어 왔으며, Rothmel et al. 등에 의해 소프트웨어 초기 시험의 시험 커버리지(test coverage)나 여러 소프트웨어 버전의 정보에 대한 차이를 이용한 소프트웨어 안정성 요소(safety factor)를 고려한 시험의 효율성 분석 방법이 많이 이용되고 있다[1]. 이와 유사한 연구결과로는 Harrold et al.의 회귀시험(regression test)에서 비용의 유효성(cost-effectiveness)을 평가한 방법이 있다[11].

그밖에 Gupta의 회귀시험을 위한 소스코드의 분리를 적용한 시험기술 등 다양한 방법들과 Wong et al. 이 제안한 시험 실행시스템에서 시험경로에 대한 coverage를 기반으로 한 시험항목 축소 기술들이 있다[10,17]. 이러한 기법들은 다양한 소프트웨어 버전과 고장 정보들, test coverage에 대한 아이디어를 바탕으로 그 차이를 이용하였다.

앞에 서술한 기법들은 시험활동에 대한 많은 데이터와 시험 준비에 많은 노력을 투입하는 관계로 새롭게 개발되는 소프트웨어에 대한 시스템 시험에 적용하기는 곤란하다. 그러므로 본 논문에서는 시스템 개발 초기에 쉽게 적용이 가능한 새로운 시험방법을 제안한다.

제안된 방법은 선택적인 시험 방법으로 주요 고장을 안전하게 검출할 수 있고 제한된 시스템 개발기간 내에 결함들을 제거할 수 있는 “중요결함 제거용 시스템시험” 방법이다. 이 방법을 요약하면 아래와 같다.

(1) 사용자나 시스템에서 빈번히 사용되는 기능과 관련된 고장(결함)들
(2) 시스템 신뢰도나 안정도에 심각한 영향을 주는 고장(결함)들

첫 번째 고장은 시스템 사용자에 의해 자주 사용되는 기능들과 비정상 동작 상태에서 결함을 유발하는 원인과 밀접한 관계가 있는 고장들이며, 두 번째 고장들은 시스템의 오동작을 유발하는 고장들로 심각성 관점에서 평가되어야 할 고장이다.

기능의 우선순위(Function priority) 결정과 고장의 심각성(fault severity)은 위에 언급한 2개의 요소(factor) 간에 상관관계를 제공하지 못하기 때문에 높은 우선순위를 갖는 기능이 심각성이 높은 치명적인 고장의 원인에 대한 필요성을 이용하지 못한다.

본 논문에서는 각 기능에 대한 목록과 시험항목을 검토함으로써 시스템 설계 단계에서 개발될 소프트웨어 기능의 목록과 기능들로부터 추출될 시험항목을 쉽게 도출할 수 있었다. 제안된 방법에서는 고장의 강도와 기능의 활용도에 의한 정보를 이용하여 각 시험항목의 우선순위에 따라 효과적인 시험이 되도록 기능의 우선순위를 정하였다.

또한 시험의 효과성을 검증하기 위해서 응용 프로그램에 대한 소프트웨어 실 시험(real testing)을 수행하고 그 결과를 분석하였다.

결론적으로 말하면 선택적인 시험은 높은 우선순위를 갖는 기능들과 밀접한 고장들을 검출하는 신뢰성을 보였으며, 시스템의 안정성과 신뢰도에 관련되는 중대한 고장들을 대부분 발견하는 결과를 보였다.

선택적인 시험방법의 기본 개념(basic concept)과 아이디어는 수행된 시험항목과 함께 시험항목에 대해 효과적인 시험수행 결과의 예를 그림 1, 2에 간략하게 나타내었다.

본 논문의 구성은 2장에 제안된 선택적 시험방법의 개요를 간략히 언급하고 3장에서 제안된 시험방법의 상세 절차를 4장에서 실제 소프트웨어 시험에 적용된 결과를 보이고 마지막 5장에서는 결론 및 향후 연구방향에 대해 제안한다.

Ⅱ. 제안된 시험 방법

제안된 선택적 시험방법은 제한된 시험기간 내에 사용자에게 심각한 영향을 주는 많은 고장(에러)들을 검출하도록 시도되었다. 즉 기능들에 대한 우선순위를 결정하기 위한 여러 metrics 정보를 대상 소프트웨어의 특성을 고려하고 사용자 관점에 초점을 맞추어 도입하였다. 우선 선택적인 시험방법을 논하기에 앞서서 기존에 수행해 왔던 시험방법에 대해 요약해 보면 아래 1절의 내용과 같다.

2.1 기존 시험 기법

과거에 수행해온 전통적인 시험기법은 그림 1과 같이 시험자(test engineer)는 시험절차서(test procedure)에 명시된 절차에 따라 기능의 요구사항을 확인하는 과정을 거쳤다[15]. 첫 번째 단계에서 규격서(Specification Document)에 언급된 기능의 요구사항에 따라 순차적으로 시험 항목(또는 절차)을 추출하고 다음단계에서 시험절차서(Test Procedure)에 작성된 절차에 따라 시험을 수행한다. 그러므로 시험이 수행되는 동안에 시험항목들은 검증되거나 선택되지 않는다. 그림 2와 표 1에 표현된 시험절차서와 기능 규격(function specification)의 예를 참조해보면 알 수 있다.

그림 2에 기술된 소프트웨어 기능 규격서는 sOLT 시스템의 스위치 패브릭(switch fabric)에서 패킷처리 기능에 대한 절차의 예이다[18].

시험자는 시험절차서 작성 수립시 소프트웨어 기능규격서(Function Spec.)를 참조하여 기능의 요구사항을 정리하여 확인할 시험 항목(예, test-item or checklist 작성)을 정한다.

2.2 제안된 방법의 개요

제안된 선택적 소프트웨어 시험방법은 3개의 측면을 고려하는데 기능의 우선순위 조정, 시험절차의 순서 배정, 시험수행 등으로 나눌 수 있다. 그 절차에 대해 그림 3에 나타내었다.

우선 시험 규격은 2장 1절에서 언급한 것처럼 기

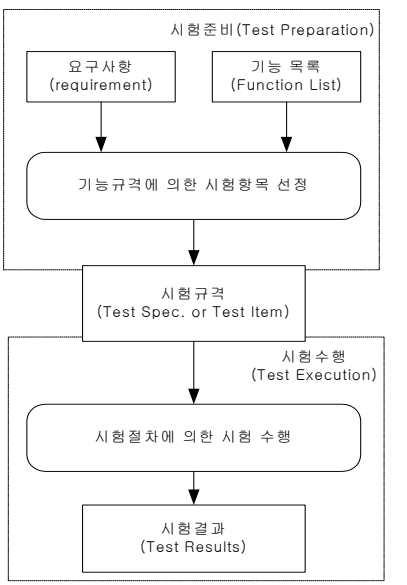


그림 1. 기존 소프트웨어 시험 방법

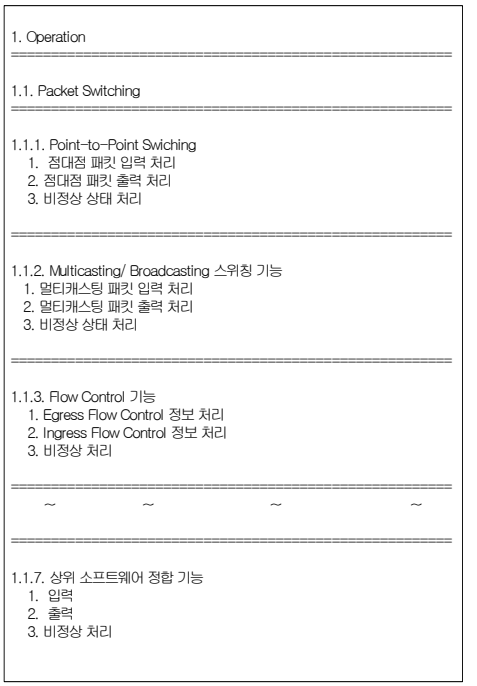


그림 2. 소프트웨어 동작 규격서 예

존 시험방법에 따라 구성되며, 시스템의 여러 기능들은 다양한 측면의 검토를 거쳐 우선순위를 정한다.

시험수행(test execution) 절차는 최우선 순위 기능들에 대해서 상세 시험절차들을 적용하며, 시험항목의 우선순위는 기능의 순서에 따라 상속된다. 이때 상세 시험절차들에 대한 변경에 따라 시험의 질과 양이 결정된다. 그러므로 효과적인 시험을 이용하기 위해서는 제한된 기간 내에 최대의 신뢰도를 얻을 수 있는 방법이 적용된다. 마지막 시험단계에서는 시험절차를 수행하게 된다. 제안된 선택적 소프트웨어 시험은 우선순위가 높은 기능들이 낮은 순위(low priority)의 기능들에 앞서 시험이 수행된다.

표 1의 기능_ID(Function_ID) 1, 2는 트래픽관리기(Traffic Manager : TM)로부터 수신되는 패킷들을 즉, 시스템 정합부(system interface part)로부터 입력된 최대 길이가 제한된 패킷을 받아서 점대점 [point-to-point(ptp)] 스위칭 기능을 수행해서, 시스템의 정합부로 송신하는 동작이다.

기능_ID 3, 4는 시스템의 정합부로부터 입력된 최대 길이가 제한된 패킷을 받아서 Multicasting/ Broadcasting 스위칭 기능을 수행하여 시스템 정합부로 송신하는 동작으로서 최대 길이가 제한된 패킷을 TM으로부터 수신하기 전에 멀티캐스팅(Multicasting) ID를 Index로 하는 Look-up Table에 해당 Multicasting/Broadcasting 패킷이 출력되는 목적지 표시 정보를 설정한다.

입력단(Ingress) 스위치 패브릭은 패킷을 수신하여 Multicast/Broadcasting Queue에 Class별로 저장하고 Multicast/Broadcast Queue를 Scheduling하여 Queue에 저장된 패킷에 QoS를 제공한다.

Interconnection 스위치와의 중재 기능 및 교환을 통하여 교환된 패킷을 출력단(Egress) 스위치의 Queue에 저장한다. 이 때 Egress Queue는 우선순위에 따른 Queueing을 처리하며, 출력단 스위치 패브릭은 Queue를 Scheduling하여 해당 패킷을 시스템 정합부로 송신한다.

기능_ID 10 ~ 12는 수집된 스위치 패브릭의 상태 정보를 상위 소프트웨어로 전달하여 시스템 차원에서 분석, 관리된다. 이를 위하여 스위치 패브릭은 상위 소프트웨어와 정합한다. 상위 소프트웨어와의 정합은 시스템 장애관리, 성능 관리, 그리고 운용자 정합 소프트웨어와 정합된다. 또한 상위 소프트웨어와의 정합은 2단계 이상의 단계적으로 정합하며, 단계별로 송수신된 스위치 패브릭의 정보를 상세화 한다.

표 1. 기능에 대한 시험절차서 예

Func. ID	Func. Name	Testing Item	Expected Results
1	TM으로부터 패킷을 수신	최대 길이가 제한된 점대점 패킷 처리	TM으로 점대점 패킷 송신
2	스위치 패브릭 제어 정보	점대점 Queue 및 버퍼의 Threshold 값 처리	통계 정보, 비정상 상태 발생 정보
3	TM으로부터Multicasting/ Broadcasting 패킷을 수신	가변길이의 Multicasting/Broadcasting 패킷 처리	가변길이의 Multicasting/ Broadcasting 패킷 송신
4	스위치 패브릭 제어 정보	패킷 통계 정보 및 비정상 상태 발생 정보 수집	통계 정보
~	~	~	~
10	스위치 패브릭 장애 상태 요청,	장애 Event 발생마다 스위치 패브릭 장애 상태 보고	스위치 패브릭 장애 상태 보고
11	주기 적인 신호	주기적인 통계 데이터 보고	스위치 패브릭 통계 데이터
12	운용자명령어 처리	운용자 명령 수행 결과 보고	운용자 명령 결과

Ⅲ. 상세 시험 절차

3장에서는 시험 수행시 수행할 순서(우선순위 조정)를 정하는 방법과 상세 절차에 대해 논한다. 서비스 이용 빈도의 관점에서 주요기능에 대한 시험의 우선 수행과 검증방법, 절차에 대해 기술한다.

3.1 기능의 우선순위 조정

3.1.1 정보의 조합(metrics)

선택적 시험방법은 사용자나 시스템에 대해 주요 기능들에 관련된 결함(고장)들을 안전하게 검출해 내는 것이다. 먼저 시스템과 사용자에 대한 주요 기능들에 대한 조사를 단행하고 추출된 주요 기능들을 사용자의 관점에서 분석하고 여기서 추출된 정보인 3가지 파라미터들을 조합, 적용한다. 다시 말해서 사용자에 의한 이용빈도(frequency of use), 시나리오 복잡도(complexity of use scenario), 고장의 영향도(impact of fault) 등이다.

이러한 관점에서 그 정보들이 개발자의 경험과 질의응답(interview)을 통해서 선택적 시험방법 적용에 고려되었으며, 사용자에 의한 이용 빈도가 높은 주요 기능들에 고장이 발생할 경우 많은 사용자에게 영향을 줌으로 이 데이터는 매우 중요한 정보이다[12]. 그래서 우리는 고장 발생을 가정한 "고장 영향도"와 "사용빈도" 등 2개의 metrics에 대한 평가를 하였다.

사용빈도는 Musa et al.의 운용 프로파일 관점에서 본 시험항목을 적용하였다. 사용자에 대한 고장 영향도 정보는 Musa 방법의 시험항목 선택에 대한 metrics 정보로 이용되지 못하는 정보로써 배포된 소프트웨어 버전에 대한 신뢰도 분석 metrics로 활용하였다.

반면에 사용자 이용 경향에 따른 기능들은 사용자들의 이용 습관에 따라 사용자 운용 에러가 발생하게 된다. 이러한 형태의 소프트웨어는 다수의 복합 모듈을 갖는다. 그러므로 우리가 적용한 방법은 시나리오를 이용한 복잡도를 검증하는 것이다. 이러한 검증방법은 전 세계 여러 곳에서 테스트 엔지니어(test technician)의 경험(empirical methodology)에 의해 소개되고 있다.

3.1.2 검증방법(evaluation)

기능에 대한 우선순위 조정은 1절에서 언급한 metrics 정보를 이용하여 정량적으로 분석하였다. 특히 시험항목의 선택은 이 방법의 가장 중요한 요소 중의 하나이며, 새로운 시스템 개발에도 이 방법을 적용할 수 있다. 즉, 별도의 문서가 필요 없이 기존의 문서(document)만 가지고도 기능들에 대한 시험을 수행할 수 있다. 이것들에 대한 내용은 표 2에 가이드라인을 정리하였다.

시험의 우선순위 metrics를 정하는데 있어서 가능한 한 위의 3개 Metrics 가이드라인 정보를 참조하여 해당 정보들을 조합하여 적용하였다.

3.2 상세 시험절차 배정

기존에 시행되어 오던 시험방법은 시험항목 자체가 시스템 개발 산출물의 설명서로부터 포괄적으로 추출되어 사용자 관점의 시험 우선순위가 반영되지 못한 채 커다란 그룹 형태의 시험항목으로 도출되어 기능 검증시험에 제대로 반영되지 못했다.

제안된 방법은 각 기능 규격서의 덩어리로부터 추출된 정보가 배제된 기능에 대한 시험의 우선순위가 정해진 시험절차를 토대로 시험항목이 도출되었다. 즉, 시험절차는 우선순위에 따라 명쾌하게 작성된 항목들에 대해 실시된다. 반면에 순위가 낮은 시험항목들은 기능규격서로부터 상세 기술된 것을 토대로 환경에 맞게 직접 시행된다. 각종 규격서(specification)를 참조하여 시험의 양과 질을 조절하며, 이때 시험자의 운용기술(test operation skill)은 배제한다.

표 2. 각 metrics에 대한 검증 가이드라인

Fault impact to users(time to recovery)									
1 min/fault			1 hour/fault			1 day/fault			1week/fault
1	2	3	4	5	6	7	8	9	10
use frequence									
1 time/month		1 time/week		1 time/day			1 time/hour		1 time/min
1	2	3	4	5	6	7	8	9	10
complexity of scenario									
1 op/func.				10 op/func.					20 op/func.
1	2	3	4	5	6	7	8	9	10

* op/func. : operation/function

3.3 사례연구(case study)

3.3.1 시험 우선순위(test priority)

각 기능에 대한 순위를 정하기 위해 시나리오 복잡도(m_1), 사용빈도(m_3), 고장영향(m_2)에 대해 분석한 후 그 값을 계산한다. 이 값을 계산하기 위해서는 앞에서 언급한 3개 metrics 정보와 함께 각 metrics에 대해 3단계의 가중치(weighting factor)를 두고 이를 서로 조합하여 계산한다.

이에 대한 간략한 요약 정보는 표 3과 같다.

표 3. 각종 metrics 정보

	strategy		
	n_1	n_2	n_3
Metrics	0.2	0.5	0.3
m_1	0.1	0.3	0.6
m_2	0.3	0.4	0.3
m_3			
score	0.2	0.4	0.4

n1 : non-weighted
n2 : 사용자에 영향을 주는 요소(weighted summation)
n3 : 내부 소프트웨어 품질에 영향을 주는 요소 (weighted summation)

표 4의 각 기능별 metrics 정보를 종합, 적용한 결과 트래픽 관리 모듈로부터 길이 제한을 갖는 패킷의 수신 처리는 우선순위가 중간정도인 반면에 멀티캐스팅 및 브로드캐스팅 패킷을 수신 처리하는 기능은 순위가 높은 것으로 분석되었다. 즉, 패킷처리라는 커다란 기능을 가지고 세부 기능 item별로 분리하고 이를 3개의 metrics 정보로 계산하여 조합하면 평균적으로 스위치 패브릭(switch fabric)이 처리하는 기능은 우선순위가 중간정도로 계산된다. 이와 같은 방법으로 기능의 우선순위를 정해 시험항목 선정 및 수행의 순위를 정하는 입력 데이터로 활용한다.

3.3.2 시험 수행(test instruction)

우선 예상되는 시험환경(test-environments)과 기능들에 대한 예상 결과를 시험규격서(또는 절차서)에 기술하고 환경 및 결과에 따른 동작 요구를 기존의 시험방식에서 사용한 것과 같이 기술한다. 그 다음 시험순위에 따라 높은 순위의 기능에 대해 시험항목을 적용, 선택적 소프트웨어 시험을 수행한다. 이러한 절차는 시험자(test engineer)가 각 항목에 대한 순서(test execution order)를 알기 때문에 경험에 의한 상세 시험 절차를 분류함으로써 효율적인 시험은 점점 더 개선될 것이다.

예를 들면 시험자(test engineer)는 소프트웨어의 비정상적인 환경(illegal behavior)에 대한 검증이나 예기치 않은 입력에 대한 처리 여부를 확인하는 것이다. 낮은 순위의 기능들에 대한 시험은 단지 "시험환경에 대한 기본 기능만 확인" 하는 간단한 절차로 수행함으로써 시험의 효율성과 시간, 비용을 절약하는 방법이다.

Ⅳ. 소프트웨어 시험 적용 결과 분석

본장에서는 실제 시험에 적용한 결과에 대한 분석 및 소프트웨어 시스템에 구축한 내역에 대해 기술한다.

4.1 적용 대상 소프트웨어의 특징

적용대상 소프트웨어 시스템은 광가입자망 구축을 위한 핵심장비로 음성, 데이터, 영상서비스 제공 및 품질(QoS) 보장 등 다양한 가입자의 욕구를 단일 플랫폼으로 차별화하여 제공하는 E-PON(Ethernet Passive Optic Network) 시스템으로 고품질의 VoD 서비스, CATV 방송서비스, TDM 모듈을 통한 음성서비스 제공이 가능한 소프트웨어 시스템이다. 이 시스템에 대한 주요 정보를 요약하면 아래와 같다.

표 4. 각 기능별 우선순위

Func. ID	function name	complexity	impact fault	freq. to use	priority	
1	TM으로부터 패킷을 수신	3	3	2	2.7	medium
2	스위치 패브릭 제어 정보	3	3	3	3.0	medium
3	TM으로부터Multicasting/ Broadcasting 패킷을 수신	6	7	2	5.0	high
4	스위치 패브릭 제어 정보	2	4	1	2.3	low
5	flow control	6	5	2	4.3	medium
~	~	~	~	~	~	~
10	스위치 패브릭 장애 상태 요청,	4	3	2	3.0	medium
11	주기 적인 신호	2	2	1	1.7	low
12	운용자명령어 처리	2	2	5	3.0	medium

* Priority range : 0 ~ 2.5(low), 2.5 ~ 5.0(medium), 5.0 ~ 10.0(high)

● 시스템 소프트웨어 정보
① Language : C
② 소프트웨어 규모 : 1,350K
③ 소프트웨어 재활용 정도 : 30 %
④ 개발환경 : Linux Redhat 7.3 이상

4.2 시험 목적

이절에서는 시험 절차서(test procedure or test specification)를 이용하여 실제 시험절차(test procedure)를 수행해서 얻은 결과에 대한 분석이다. 2장 및 3장에 기술한 것과 같은 시험방법을 이용하여 시험순위에 따른 선택적 시험을 B-team에 적용하고 A-team은 기존에 수행해온 시험방법으로 수행한 결과다. 이렇게 적용한 시험결과는 다음과 같은 질문에 초점을 맞추어 검증하였다.

(1) 순위 결정시, 우선순위를 갖는 기능에 대해서 많은 고장(에러)을 검출하였는가?
(2) 신뢰도와 안정도에 관련하여 발견된 치명적인 고장에 차이가 있는가?
(3) 시스템 인도(배포 : release)전 핵심기능이나 시스템 신뢰도, 우선순위 기능에 관련된 고장들이 제거될 수 있는가?

하는 것으로 이에 대한 분석 자료는 시스템 개발 및 향후 새로운 시스템 개발에 매우 유용한 데이터로 활용된다.

4.3 적용절차(application procedure)

4.3.1 시험팀(test team)

검출된 고장 데이터의 동향을 비교하기 위해서 시험자의 기술적인 경험에 의한 시험기술(test skill)을 배제하고 유사한 경험과 기술을 사용하여 실험하였다. 즉, 각 팀에서는 가능한 한 동일한 조건을 위해 실험환경을 만들어 수행했다. 이에 대한 개요는 그림 3과 같다.

4.3.2 시험항목 추출(test item generation)

실험기간동안 각 팀은 독립적으로 시험을 수행하였으며, 검출된 고장들은 정리(summary)되어 소프트웨어 개발자에게 고장 제거 및 디버깅을 위해 feedback 되었다.

4.4 시험시간과 고장검출

4.4.1 시험시간(test time)

시험시간의 소요는 전체 기능을 기존의 방법대로 수행하는 경우(conventional testing)와 시험의 우선순위(test priority)를 갖는 선택적 시험방법(selective testing method)을 비교할 때 선택적 시험방법이 기존방법에 비해서 약 1/4 정도 소요되는 것으로 판명되었다.

기존방식의 시험은 4개월에 걸쳐 총 153개 고장이 검출되었으며, 반면에 선택적 시험방법은 2주에 걸쳐 52개 고장을 검출하여 조치하였다. 이 결과는 표 5-1, 2에 정리하여 표시하였으며, 시험의 효율성

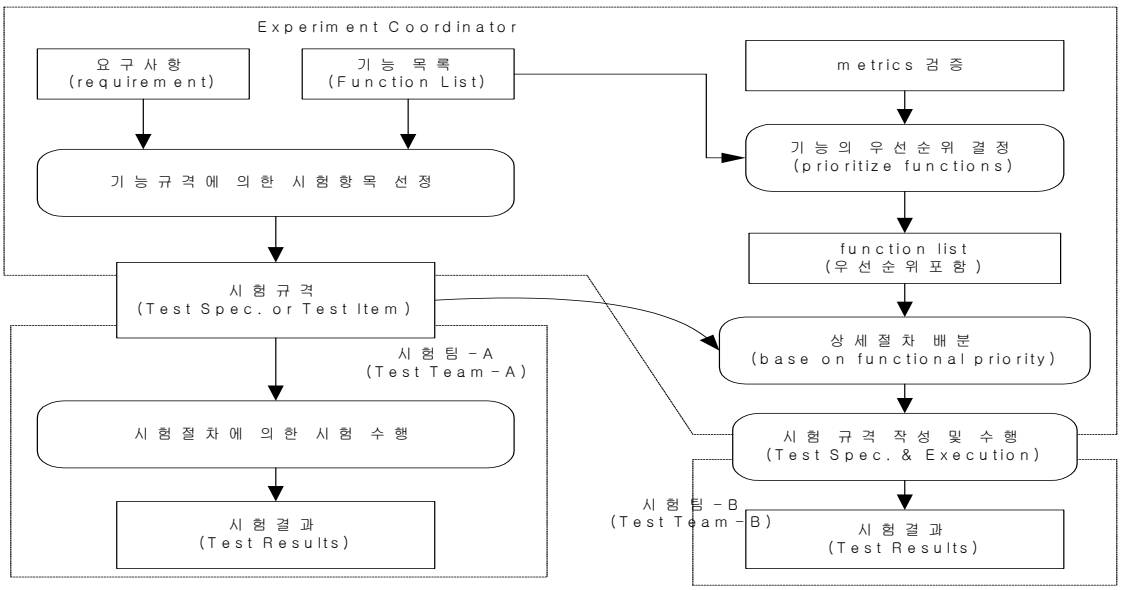


그림 3. 선택적 시험을 위한 실험 환경 개요

과 경제성 분석에 매우 중요한 정보로 활용되었다.

그밖에 시스템 시험에서 검출된 총 고장 데이터에 대한 심각도(severity) 분석결과는 그림 4와 같다. 대체로 검출된 고장 데이터는 영향도가 보통(severity 3)인 것이 가장 많고(약 44%) 고장해결에 많은 시간이 소요되는 심각한 고장(severity 4 이상)은 전체 2%, 비교적 사소한 고장(severity 1, 2)이 절반(52%) 정도 차지하는 것으로 분석되었다.

4.4.2 고장 검출(fault detection)

검출된 고장데이터는 시험의 우선순위 metrics 정보와 사용자 관점의 분석 결과를 토대로 "주요고장"으로 분리되고 이 고장들은 다시 수준별로 critical(심각), major(중요), minor(사소) 등 3개의 레벨로 분류된다[14].

● 고장의 수준(failure level)

- critical faults :
  시스템 신뢰도나 안정성에 영향을 주는 심각한 고장으로 시스템의 오동작을 유발하는 결함들
- major faults :
  시스템 동작에 제한적인 영향을 주는 고장으로 critical 보다 약함
- minor faults :
  다수의 사용자가 연속 이용시 별도의 고장발생 없이 순간적으로 복구되는 미약한 고장들

시험에서 검출된 고장들은 시험의 우선순위와 주요고장들(serious of faults)과의 metrics 정보로 정리, 분석된다. 이에 대한 현황은 표 5-1, 2와 같다.

표 5-1, 2의 고장검출 데이터에 대한 분석 종합결과를 놓고 볼 때 기존 수행방식은 고장검출에 많은 시간이 소요되었으나 Test priority가 낮은 기능(시스템 및 사용자 측면의 사소한 고장들)들에 대해 많은 고장을 발견하고 있음을 알 수 있다. 반면에

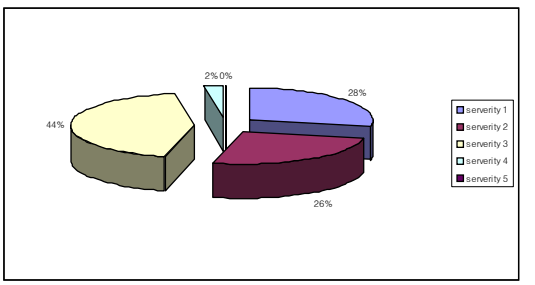


그림 4. 심각도(深刻度)별 故障 데이터 분포 현황

표 5-1. 기존방식의 시험 종합 결과

		serious of faults			
		critical	major	minor	total
test priority	high	3	0	1	4
	medium	15	10	23	48
	low	20	34	47	101
total		38	44	71	153

표 5-2. 선택적 시험의 종합 결과

		serious of faults			
		critical	major	minor	total
test priority	high	5	4	3	12
	medium	4	7	7	18
	low	2	5	5	12
total		11	16	15	52

선택적 시험방법은 짧은 기간 안에 주요 고장을 검출하여 시스템 개발에 feedback 시킴으로써 개발기간을 줄일 수 있다는 장점을 가진다.

4.5 주요고장 검출과 고장 제거

4.5.1 주요고장 검출(critical fault detection)

3장 1절에 언급한 것과 같이 시험항목의 우선순위는 기능의 우선순위에 의해 상속된다. 즉, 고장은 높은 순위의 기능과 관련되는 시험항목에 의해 검출된다. 가능한 한 우선순위를 갖는 기능들에 관련된 고장들에 많은 관심이 집중되며 이러한 고장들은 심각한 고장 원인에 의한 고장이거나 사용빈도가 높은 기능에 관계된 고장으로 고장 발생시 쉽게 상호관계(interrelation)를 밝힐 수 있다.

사용자에 의한 높은 순위를 갖는 기능에 관련된 고장들은 그 문제가 사소한 것일지라도 예외 없이 반드시 검출되고 제거되어야 한다. 이와 같은 실험데이터는 우선순위를 갖는 기능검증에 대해 시험절차로써 제공되어야 하며, 이러한 절차는 고장검출을 위해서 그 효과 분석이 필요하다.

4.5.2 고장제거(fault remove)

소프트웨어 인도(引渡)전에 검출된 고장의 시험방법에 대한 검증이 필요하고 검출된 고장은 반드시 제거되어야 한다. 제거되어야 할 고장의 의미는 아래와 같이 정의 할 수 있다.

● 고장의 의미(meaning of fault)

(1) 사용자 관점 또는 시스템의 핵심 기능에 관련된 고장들

(2) 시스템의 신뢰도 또는 안정도 관점에서 다른 시스템에 심각한 영향을 주는 고장들

이며, 이 고장들은 분류되어 관리되어져야 한다. 이에 대한 내용을 유형(type)별로 요약을 해보면 아래와 같다.

● 고장의 유형(fault type)
- Type-1 : 반드시 제거되어야 할 고장(심각한 고장으로 즉시 조치 필요)
- Type-2 : 수정이 요구되는 고장(시스템 고장으로 간주하고 빨리 제거되어야 함)
- Type-3 : 사소한 고장(운용상 지장 없음)

위와 같이 sOLT 시스템 개발과정에서 시스템시험을 통해서 고장이 검출, 해결된 월별 추이도(推移度)는 그림 5와 같다. 표 6의 2개 팀에서 실시한 결과를 놓고 볼 때 선택적 시험방법에 의해 검출된 고장들은 시스템을 사용자에게 인도하기 전에 조치하여야 하고 또 시험방법의 효율성에 대한 고려가 수행되어야 한다. 각 타입별 검출결과를 표 6에 종합, 정리해 놓았다.

이 결과를 보면 사용자에게 시스템을 인도하기 전에 조치하여야 할 주요 고장 즉, 시스템의 신뢰도나 안정도에 관련된 고장들을 선택적 시험방법을 이용해 실시한 B-팀이 더 많은 고장을 발견했음을 알 수 있다.

사소한 고장(Type-3)인 경우에는 개발자가 아무런 조치를 취하지 않아도 해결(복구)되는 고장(latent fault)도 일부 포함되어 있어 시험방법의 효율성을

표 6. 주요 고장에 대한 비교 결과

고장유형	Team-A	Team-B
Type-1	4	12
Type-2	35	6
Type-3	114	24

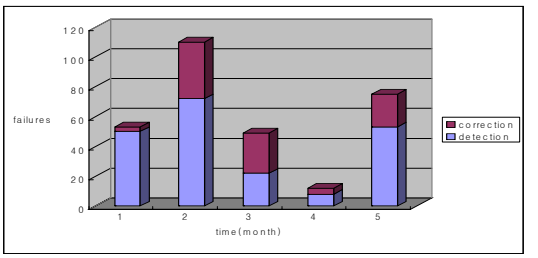


그림 5. 월별 고장 검출 및 제거 현황

따질 경우 기능의 우선순위(Function priority) 및 시험우선순위(test priority)를 고려한 선택적 시험방법이 더 효율적임을 알 수 있었다. 이러한 시험방법은 향후 계속 보완, 적용하고 그 유효성을 검증하여 새로운 시스템 개발에 적극 활용해야 한다.

V. 결 론

본 논문은 시험의 효율성을 높이기 위한 선택적 시험방법에 대해 논하였다. 과거 시스템 개발과정에서 결함 검출율(fault exposure ratio)을 높이기 위해 핵심 기능범주(function category)로 분류하여 시험의 우선순위를 부여, 소프트웨어 버전별 시스템시험을 수행한 결과가 발표[9]된 적이 있으나 각 기능별로 priority를 조정하고 test item에 대한 순위와 함께 종합적인 metrics 분석 정보를 적용한 예는 극히 드물다.

선택적 시험방법의 목적은 고장 검출의 효율을 높이고 발견된 고장을 제거하는 것이다. 특히, 대형 시스템 개발 프로젝트에 적용한 이와 같은 시도는 많은 노력과 시간이 필요하다.

향후 시스템시험의 효율성을 검증하는데 좀 더 다른 시각 즉, 사용자나 시스템 관점에서 제안된 방법의 정량적인 분석이 필요하다. 그밖에 선택적인 시험방법을 여러 시스템에 광범위하게 적용하여 그 효율성을 검증 받아야 한다.

참 고 문 헌

[1] Gregg Rothermel, Mary Jean Harrold, "Empirical Studies of a Safe Regression Test Selection Technique", IEEE Trans. on Software Engineering, Vol. 24, No. 6, pp.401-419, JUNE 1998.

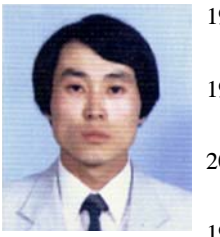
[2] Gregg Rothermel, Roland H. Untch, Chengyun Chu, Mary Jean Harrold, "Prioritizing Test Cases for Regression Testing", IEEE Trans. on Software Engineering, Vol. 27, No. 10, pp.929-948, Oct. 2001.

[3] SHNJI INOUE and SHIGERU YAMADA, "Testing-Coverage Dependant Software Reliability Growth Model", International Journal of Reliability, Quality and Safety Engineering, Vol. 11, No. 4, pp.303-312, 2004.

[4] B. Beizer, Block-Box Testing, John Wiley

& Sons, New York, 1995.
[5] D. Binkley, “Semantics guided regression test cost reduction”, IEEE Trans. on Software Engineering, Vol. 23, No. 8, pp.498-516, Aug. 1997.
[6] Sebastian Elbaum, Gregg Rothermel, “Incorporating varying test costs and fault severities into test case prioritization”, Proc. 23rd International Conference on Software Engineering, pp.329-338, 2001.
[7] N.E. Fenton and S.L. Pleeger, Software Metrics : A Rigorous & Practical Approach, PWS Publishing, 1997.
[8] Sebastian Elbaum, Alexey G. Malishevsky, Gregg Rothermel, “Test Case Prioritization : A Family of Empirical Studies”, IEEE Trans. on Software Engineering, Vol. 28, No. 2, pp.159-182, Feb. 2002.
[9] 이재기, 유재연, “기능블럭을 갖는 교환 소프트웨어의 정량적인 신뢰도 평가”, 대한전자공학회논문지, SE No. 29, pp. 1096-1104, 1998.
[10] R. Gupta, M.J. Harrold and M.L. Soffa, “An approach to regression test using slicing”, Proc. Conference on Software maintenance, pp.299-308, 1992.
[11] M.J. Harrold, D. Rosenblum, Gregg Rothermel, and Weyuker, “Empirical Studies of a prediction model for regression test selection”, IEEE Trans. on Software Engineering, Vol. 27, No. 3, pp.248-263, 2001.
[12] J.D. Musa, “Software Reliability Engineering testing”, IEEE Software, Vol.29, No. 11, pp.61-68, Nov. 1996.
[13] J.D. Musa, Software Reliability Engineering : Faster Development and Testing, McGraw-Hill. 1998.
[14] D.M. Marks, Testing very big Systems, McGraw-Hill. 1992.
[15] W. Perry, Effective methods for Software testing, Wiley Publications, 1995.
[16] J.A. Whittaker, “What is Software testing? and Why it is so hard?, IEEE Trans. on Software Engineering, Vol. 21, No. 1, pp.70-79, 2000.
[17] Wong, J. Hogan, S. London and H. Agawal, “A Study of effective regression testing in practice:, Proc. 8th International Symposium on Software Reliability Engineering, pp.230-238, 1997.
[18] ETRI, S-OLT/ONT 시스템 기술, ETRI 광대역통합망연구단, 2004. 6.

이 재 기 (Jae-ki Lee) 정회원



1985년 2월 서울산업대학교 전자공학과 졸업
1988년 2월 청주대학교 전자 공학과 석사
2004년 2월 공주대학교 정보 통신공학과 박사
1983년 3월~현재 한국전자통신연구원 책임연구원

<관심분야> 소프트웨어 신뢰도, 시험검증, 요구공학

이 재 정 (Jae-jeong Lee) 정회원

2002년 2월 한밭대학교 전자 공학과 졸업
2007년 3월~현재 공주대학교 정보통신공학과 석사과정
1999년 3월~현재 한국전자통신연구원 연구원

<관심분야> BcN, 시스템 시험/검증(네트워크장비)