블랙 박스 테스팅 방법들간의 결함 검출 효율성에 관한 실험적 비교

문중희[O] 전성희 김성훈 권용래
삼성전자㈜
{joonghee.moon[O], seonghee, kimsh7}@samsung.com, {kwon}@cs.kaist.ac.kr

The Experimental Comparison of Fault Detection Efficiency of Black Box Testing Methods

Joong Hee Moon[O], Seong hee Jeon, Sung Hoon Kim, Yong Rae Kwon†
Samsung Electronics, KAIST †

요 약

소프트웨어 테스팅을 위해서 테스트 케이스를 작성하는 작업은 어렵고 많은 비용을 요구한다. 예로 약 100,000라인의 코드를 테스트하기 위해서는 천문학적인 테스트 시나리오들이 필요할 수도 있다. 따라서 경험 있는 테스터들은 필요한 테스트 케이스들만을 선별적으로 사용하고자 한다. 그리고 이를 위한 많은 테스트 기법들이 연구되고 있다. 그러나 다수의 연구 자료들은 기법의 효과를 이론적으로만 제시한다. 일부 사례를 통해서 그 효과를 제시하는 자료들도 있으나 그 적용 과정이 구체적이지 않아 신뢰를 얻기가 어려운 경우가 많다. 본 논문에서는 업계에서 많이 사용되는 9가지 테스팅 방법들을 소개하고 이들을 실제 개발 과제에 적용하였다. 그리고 각각의 결과를 비교하고 분석하였다. 본 논문의 결과를 일반화하기는 어려울 것이다. 하지만 하나의 사례 연구로서 참고되고 활용될 수 있을 것이다.

핵심어: 소프트웨어 테스팅, 테스트 케이스, 테스트 기법, 블랙 박스 테스팅

1. 서 론

블랙 박스 테스팅을 위해서 테스터가 테스트 케이스를 만들 때에는 크게 다음과 같이 두 가지를 고려한다. 첫째, 테스트할 함수의 입력 파라미터들에 어떠한 값들을 테스트 데이터로 사용할 것인가? 둘째, 각 파라미터 별 입력 값들을 어떠한 기준에 의해서 조합할 것인가? 전자의 경우 크게는 입력 가능한 도메인에서 무작위로 값을 선정하는 Random Testing 방법과 Domain을 Sub-domain으로 나누고 각 Sub-domain 내에서 어느 대표 값을 입력 값으로 정하는 Partitioning Testing방법이 있다[7]. 입력 값들을 조합하는 방법에는 모든 가능한 조합을 적용하는 Strong analysis방법, Default 테스팅 방법 그리고 Pair-wise Combination방법이 있다[5]. 그러나 이러한 방법들의 효율성이 아직 완전하게 입증되지는 않았다. 과거 몇몇 연구들이 진행이 되었으나 그 실험 과정 및 내용이 상세하지 못하였으며 비교를 위해 적용된 방법들의 수 또한 제한적이었다. 본 논문에서는 잘 알려진 블랙 박스 테스팅 기법들을 실제 응용 시스템을 대상으로 실험적으로 평가하였다. 평가를 위해서 3가지 블랙 박스 테스팅 기법에 각 3가지 조합 방법들을 적용하였다. 적용할 응용 프로그램은 모두 디지털 텔레비전에 탑재되는 2 가지 프로그램이다. 하나는 음악 파일들의 메타 데이터 정보들을 관리하여 사용자에게 검색의 다양성을 제시하는 프로그램이다. 이에 대한 실험에서는 음악 파일 내부에 들어가는 메타 데이터들의 종류와 그 값들을 다양하게 구성하고 이들을 테스트 입력 값으로 사용하여 시스템에 어떠한 고장이 발생하는가를 관찰하였다. 다른 하나는 사진 파일이 나타내는 상의 구도를 자동적으로 분석하여 구도 별 분류를 하는 프로그램이다. 이에 대한 실험에서는 사진 파일이 보이는 상의 구도를 다양하게 설정하고 이들의 색에 대한 투명도를 변화시켜 테스트 입력 값으로 설정하였다. 이와 같이 성격이 다른 2가지 응용 프로그램을 사용하여 다양한 종류의 결함들에 대해서 여러 블랙 박스 테스팅 방법들의 결함 검출 능력을 실험적으로 비교하고자 하였다. 그리고 이 실험에서는 시스템이 비정상 종료되거나 기대한 것과 테스트 결과가 다르게 나타날 경우에는 결함으로 간주하고 발견되는 결함 종류의 수를 비교하였다.

2. 기존의 연구

본 장에서는 현존하는 테스팅 기법들을 실제 개발 과제들에 적용하고 그 결과를 평가한 자료들을 소개한다.

2.1. On Random and Partition Testing[7]

Partition testing과 Random Testing을 적용하여 결함 검출 능력을 실험적으로 비교하였다. 실험 결과에서 주목할 만한 차이가 없었으나 비용을 고려한다면 오히려 Random Testing이 효과적이라고 설명한다. 결과의 신뢰성을 높이기 위해서 동일한 실험을 1,000번 나아가 2,000번 수행하였다.

2.2. The Combinatorial Design Approach to Automatic Test Generation [9]

Combinatorial Test 기법을 사용하여 기존의 방법으로는 발견할 수 없었던 오류를 발견하였다고 설명한다. 또한 비교 결과를 Coverage측정 값으로 제시한다.

2.3. Experimental Comparison of Three System Test Strategies[10]

인위적으로 defect를 삽입한 시스템을 대상으로 세가지 테스트 기법을 적용하였다. 첫 번째 경우에는 입력 데이터 값을 무작위로 설정하여 적용하였다. 두 번째 경우에서는 입력 값을 무작

위로 선정하여 적용하고 만일 Fail로 되었을 경우에는 오류를 발생시키는 부분을 바로 잡고 다음 입력 값을 적용하였다. 마지막 세 번째 방법에서는 bottom-up 방식으로 가장 하위 모듈에 대해서 테스트를 진행하고 모듈 간의 integration에 대해서 테스트를 진행하면서 최종적으로 전체 시스템을 대상으로 테스트를 진행하였다. 이 논문은 테스트 진행 방법상의 효율성을 비교하였으며 본 논문에서 설명하는 블랙 박스 테스팅 기법들간의 효율성 비교와는 차이가 있다.

3. 사용된 블랙 박스 테스팅 기법들

3.1. 입력 값들을 설정하기 위해 사용한 테스팅 기법들

테스터들은 아래 소개되는 테스팅 기법들을 사용하여 입력 변수를 선정하고 그 값을 설정할 수 있다.

- A Special Value Testing Technique [5]

  테스터는 시스템에 관한 지식, 경험을 바탕으로 입력 변수의 값을 설정한다.

- An Equivalence Partitioning Testing Technique [1]

  분할된 영역의 모든 부분에 대해서는 시스템이 같은 동작을 수행한다고 기대하고 테스트 범위를 나눈다. 테스트 입력 값은 각 분할 영역을 대표할 수 있는 값으로 선정한다.

- A Boundary Value Analysis Testing Technique [2]

  테스트 입력 값들은 입력 영역의 경계부분에서 선택된다.

3.2. 입력 값들간의 조합 방법들

2.1절에서 소개된 테스팅 기법으로 생성된 테스트 케이스들 가운데 우선적으로 적용할 테스트 케이스들을 선별하는 방법들을 소개한다.

- Strong Combination [5]

인자들이 가지는 값들의 모든 가능한 조합을 포함하는 Test Suite를 생성한다. 이 방법을 적용하면 생성해야 하는 테스트 케이스들의 수가 엄청나게 많아질 수 있기에 실제 상황에서는 가장 비효율적인 방법으로 분류된다. 본 논문에서는 실험적 비교를 위해서 Cartesian product[3]로 생성된 테스트 케이스들 가운데 pair-wise combination으로 생성된 경우와 같은 수의 테스트 케이스를 무작위로 선정한다.

- Default Combination [5]

Single Fault 가정에 기반을 둔 것으로 입력 가능한 값이 최소한 하나의 테스트 케이스에는 포함되도록 조합한다.

- Pair-wise Combination [4][8]

대부분의 오류가 하나 혹은 두 개의 입력 인자들이 가지는 값들의 상호 작용에 의해서 발생한다는 관찰에 기반한다. 이에 의해서 한 쌍의 입력 파라미터에 대하여 이 두 파라미터가 취할 수 있는 유효환 값들의 모든 조합이 최소 하나의 테스트 케이스에는 포함되도록 한다. 본 논문에서는 all-pairs라는 도구를 사용하여 Test Suite를 생성한다.

4. 평가 방법

4.1. Subject Application

실험에 적용할 프로그램은 사용자가 방대한 양의 컨텐츠를 저장하고 이들을 손쉽게 찾을 수 있는 CE(Consumer Electronics, 소비자 전자 제품) 기기를 위한 DCM(Digital Contents Management) 시스템[6]이며 두 개의 프로그램으로 구성되어 있다. 각각의 Application은 다량의 사진 및 음악 파일들을 저장하고 이들이 가지는 메타 데이터 정보들을 별도의 데이터 베이스로 관리한다. 그리고 메타 데이터 정보를 기반으로 사용자에게 다양한 기준에 의한 검색 기능을 제공한다. 또한 컨텐츠의 내용을 분석하여 음악 파일들을 무드(Mood) 별 정렬하고 사진 파일들을 구도 별 정렬하여 보여준다.

4.2. 음악 파일 등록 Application을 사용한 실험 방법 및 절차

사용자는 DCM을 사용하여 mp3 음악 파일들을 등록할 수 있다. 사용자는 mp3 음악 파일을 저장할 때 파일에 관한 메타 정보를 파일 내부에 넣어서 제공해야 한다. 그러면 Application은 저장된 음악 파일들이 가지는 메타 데이터 정보들을 분석하여 사용자가 가수, 생성 날짜, 제목 등의 기준으로 원하는 파일들을 검색할 수 있게 한다. 그러나 등록되는 파일의 메타 데이터 정보가 없거나 깨진 상태일 경우에는 시스템의 결함으로 이어질 가능성이 있다. 또한 메타 정보 외에 mp3 음악 파일 이름의 길이가 비정상적으로 길거나 파일의 내용이 비정상적일 경우에도 결함으로 이어질 가능성을 가지게 된다. 본 실험에서는 사용자가 등록하게 되는 입력 파일들의 메타 데이터, 파일 이름의 길이 및 파일의 상태를 입력 데이터로 사용하였다. 그리고 이들 입력 데이터의 내용을 일반 정상적인 값과 자체 결함이 포함된 값으로 구성하였다. 그리고 이 두 가지 종류의 입력 데이터에 대해서 시스템이 기대와는 다르게 비정상 동작하게 될 경우 시스템 코드 내부에 결함이 있다고 판단하였다. 이를 테스트 하기 위해서 테스트 입력 파라미터들을 메타 데이터, 파일 이름의 길이 및 파일의 상태를 나타내는 항목들로 설정하였다. 그리고 이들에 대한 값은 메타 데이터의 상태 혹은 수치적인 값으로 설정하였다. 그림 1은 결함에 의한 Application의 오류 메시지 및 테스트 환경을 보여준다.

[그림1] 음악 등록 테스트 환경

입력 값들을 설정하기 위해서 Special Value Analysis를 적용하는 과정은 다음과 같다. 프로그램의 요구사항을 분석하고 입력할 mp3음악 파일의 특성 가운데 프로그램에 영향을 줄 수 있는 다섯 개의 입력 파라미터들을 선택하였다. 표1에서 각각의 열들 가운데 ‘ FileNameLeng’ 는 입력 파일 이름의 길이를 나타낸다. ’ Normal’ 값인 경우에는 파일이 약 10byte정도의 일반적인 길이를 가지게 되며 MAX일 경우에는 요구사항을 기반으로 128byte의 길이를 가지도록 설정하였다. ID3Version은 입력파일의 ‘ ID3Tag’ [6] 버전을 의미한다. 4가지의 version이 존재하며 복합적으로 사용되는 경우의 수를 포함하면 모두 15가지가 되었다. ‘ File Size’ 는 입력 파일의 크기를 나타낸다. ‘ AVG’ 는 일반적인 4Mbytes의 크기를 나타내며 ‘ LARGE’ 는 요구사항을 기반으로 20Mbytes의 최대 크기를 나타낸다. ‘ ZERO’ 는 0Mbyte의 크기를 나타낸다. ‘ TAGContent’ 는 TAG정보의 상태를 의미한다. ‘ NORMAL’ 은 정상적인 상태를 의미하며 ‘ FRAGILE’ 은 깨져있음을 의미한다. ‘ 중간에서 끊긴경우’ 는 TAG정보정보가 중간에서 끊긴 경우를 의미한다. ‘ Music Contents’ 는 mp3음악 파일을 구성하는 바이너리 코드의 상태를 나타낸다. ‘ NORMAL’ 은 정상적인 상태를 의미

하며 ‘ FRAGILE’ 은 바이너리 코드가 비정상적으로 깨져있음을 의미한다. 이러한 설계를 반영하여 본 논문에서는 세 개의 조합 방법들인 Pair-wise Combination[4][8], String Analysis Combination[5] 그리고 Weak Analysis Combination[5] 을 적용할 수 있었다. 표2는 special value analysis와 pair-wise combination[4][8]에 의해서 생성된 Test Case들 및 테스트 결과를 보인다. 세 개의 조합방법들은 Equivalence Partitioning[5]과 Boundary Value Analysis[2]를 사용한 테스트 설계에도 모두 적용된다.

[표1] special value analysis를 적용한 테스트 설계 예

FileNameLeng(2)	ID3Version(15)	File Size(3)	TAGContent(3)	Music Contents(2)
NORMAL	v1.0	AVG(4M)	NORMAL	NORMAL
MAX	v1.1	LARGE(20M)	중간에서끊긴경우	FRAGILE
	v2.2	ZERO(0M)	FRAGILE	
	v2.3			
	v1.0+v1.1			

[표2] special value analysis design와 pair-wise combination가 적용된 테스트 케이스 예

case	FileNameLeng (2)	ID3Version(15)	File Size(3)	TAGContent(3)	Music Contents (2)	PASS/ FAIL	수행결과
1	AVERAGE	v1.0	AVG(4M)	NORMAL	NORMAL	PASS	
2	MAX	v1.0	20MEGA(20M)	중간에서끊긴경우	FRAGILE	FAIL	1. "-92%"라고 표시되고 멈춤 2. 그러나 계속 기다리면 업로드는 완료됨
3	AVERAGE	v1.1	AVG(4M)	중간에서끊긴경우	FRAGILE	FAIL	1. "100%"라고 표시되고 멈춤 2. command창이 사라짐

4.3. 사진 파일 등록 Application을 사용한 실험 방법 및 절차

그림2는 사진 파일들을 다섯 가지의 구도로 분류하는 기능을 나타낸다. 즉, 상이 ‘ 왼쪽’ , ‘ 오른쪽’ , ‘ 가운데’ 그리고 ‘ 분산상태’ 로 있을 때 각기 다른 구도로 분류되며 어느 경우에도 해당하지 않을 경우에는 ‘ 미분류’ 로 구분된다.

[그림2] 사진 등록 테스트 환경

테스터는 테스트 입력 사진 파일들을 USB기기를 사용하여 등록한다. 이 때 application은 사진 파일의 내용을 분석하여 이들을 각각의 구도에 맞게 분류한다. 테스터는 구도 별로 적절하게 사진들이 배치되었는지를 확인한다. 만일 적절하게 배치되었으면 결과를 ‘ PASS’ 로 처리한다. 그렇지 못 하였을 경우에는 ‘ Fail’ 로 처리한다. 예컨대 사진의 상이 ‘ 가운데’ 로 구분되기를 기대하였으나 ‘ 왼쪽’ 혹은 ‘ 오른쪽’ 과 같이 잘못 구분되었을 경우에는 Fail로 간주한다. 또한 사용자의 시각에서 사진의 상에 대한 투명도가 매우 높아서 거의 보이지 않을 경우임에도 ‘ 왼쪽’ 혹은 ‘ 가운데’ 등으로 구분을 하였을 경우에는 Fail로 간주한다. 표3는 테스트 입력 파일들의 구도에 영향을 미치는 네 가지 변수들을 보여준다. 각 열은 사진의 상이 ‘ 왼쪽’ , ‘ 가운데’ 그리고 ‘ 오른쪽’ 에 나타남을 설명한다. 마지막 ‘ 평균분산’ 은 상들이 흩어져 있다는 것을 나타낸다.

[표3] special value analysis를 적용한 테스트 설계

왼쪽(5)	가운데(3)	오른쪽(5)	평균분산(3)
없음	없음	없음	없음
투명도95%A	투명도95	투명도95%A	투명도95
투명도0%A	투명도0	투명도0%A	투명도0
투명도95%B		투명도95%B	
투명도0%B		투명도0%B	

첫 번째 행을 제외한 각 행들의 내용은 각 변수의 값들을 나타낸다. 각각의 값들은 상들이 가지는 투명도를 나타낸다. 즉 투명도가 0%일 경우에는 원래의 색을 보이게 되며 투명도가 높거나 없을 경우에는 색을 희미하거나 나타내지 않게 된다. 표3에서 나타난 입력 변수와 값들을 기반으로 strong analysis combination을 하였으며 그 결과는 표4에 반영하였다. 그리고 3.2절의 실험과 같이 세 개의 테스팅 기법에 대한 세 개의 조합 방법을 적용하여 그 결과를 역시 표4에 반영하였다.

5. 실험결과 및 분석

5.1. 테스팅 결과

표4에서 각각의 열은 9가지의 테스팅 방법들을 나타낸다. 첫 번째 행을 제외한 행들은 9가지의 결함들과 결함 종류의 수 그리고 하나의 테스트 케이스 당 결함 종류의 수를 나타낸다.

5.2. 발견된 결함들

결함 A~E는 음악 파일을 등록하는 Application에 관한 결함들이다.

- 결함 A

음악 파일이 업로드 되면서 Progress bar가 0%~100%로 증가하게 된다. 그리고 업로드가 완료되면 완료 메시지가 나타나야 한다. 그러나 Progress bar가 100%가 된 이후 완료 메시지가 나타나지 않고 Application과 함께 동작 중인 Console창이 비정상적으로 종료되는 경우는 결함 A에 해당한다.

- 결함 B

음악 파일이 업로드 되면서 Progress bar가 “ -92%” 라는 비정상적인 값을 나타내는 경우이다. 이 경우에는 파일 업로드는 성공적으로 수행되었으나 결함으로 간주하였다.

- 결함 C

음악 파일이 업로드 되는 도중 Windows Application Library에 의한 오류 메시지를 나타내는 경우이다.

- 결함 D

음악 파일이 업로드 되면서 Progress bar가 “ 100%” 를 나타낸다. 그러나 이후 완료 메시지가 나타나는 시간이 약 35초 이상 소요되는 경우에는 결함 D로 하였다. 완료 메시지는 1~2초 이내에 나타나야 하기에 결함으로 간주하였다.

- 결함 E

음악 파일이 업로드 되는 도중 Windows Operating System에 의한 오류 메시지를 나타내는 경우이다.

결함 F~I는 사진 파일을 등록하는 Application에 관한 결함들이다. 사진의 구도가 예상과는 다르게 분류된 경우이며 이는 테스터의 주관적 판단에 의해 결함으로 구분되었다.

- 결함 F

등록된 사진의 구도가 ‘ 수직’ 으로 잘못 분류된 경우이다.

- 결함 G

등록된 사진의 구도가 ‘ 미분류’ 로 잘못 분류된 경우이다.

- 결함 H

등록된 사진의 구도가 ‘ 가운데’ 로 잘못 분류된 경우이다.

- 결함 I

등록된 사진의 구도가 ‘ 평균분산’ 으로 잘못 분류된 경우이다.

5.3. 테스팅 결과 분석

사진과 음악에 대한 두 Application의 결과를 종합하여 계산하였을 경우 그림6, 7와 같았다. 전체적으로 boundary value analysis와 pair-wise combination을 적용하였을 경우에 defect type의 검출 비율이 가장 크게 나타났다. 그러나 항상 그러한 것은 아니었다. 그림4, 5에서는 pair-wise combination보다 strong analysis를 적용하였을 경우 defect type에 대한 발견 비율이 더 높았다. 또한 그림3에서는 Special value analysis와 Equivalence value partitioning을 적용하였을 경우에 pair-wise combination이 다른 조합 방법들에 비해서 특별히 defect type의 발견 비율을 높이지 못했음을 확인할 수 있다. 그러나 그림6, 7에서 보여지듯이 전체적으로는 boundary value analysis와 pair-wise combination을 함께 적용하였을 경우에 defect type의 발견 비율이 가장 높았다.

6. 결론

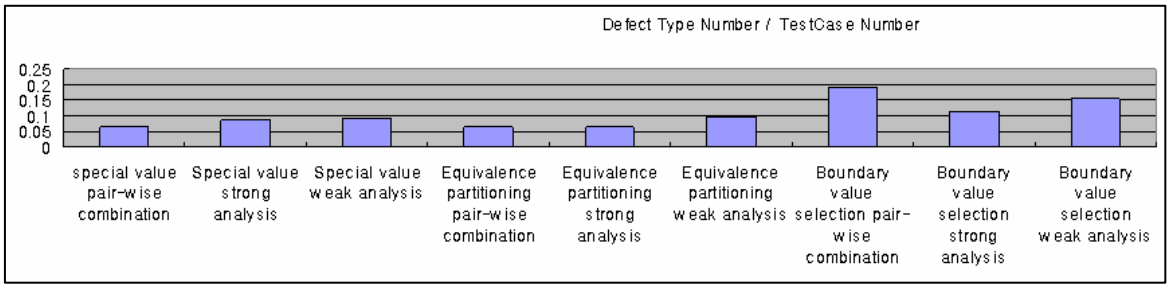
본 논문은 하나의 테스트 사례만을 소개한 것으로 테스트 결과를 일반화할 수는 없을 것이다. 논문 내용의 제약으로 테스트 기법을 올바르게 적용 하였는가에 대한 구체적인 설명이 부족하였고 발견된 결함의 종류도 테스터가 임의적으로 판단하여 분류하였다. 또한 테스트를 위한 입력 변수의 종류를 특별한 기준이 없이 테스터가 주관적인 판단으로 선택하였다. 이러한 제약을 감안하고 boundary value analysis와 pair-wise combination이 상대적으로 효과적이었음을 보여주었다. 향후 이러한 실험들이 계속적으로 수행되고 연구된다면 테스팅 방법들간의 결함 검출 효율성을 객관적으로 비교할 수 있을 것이다.

참고문헌

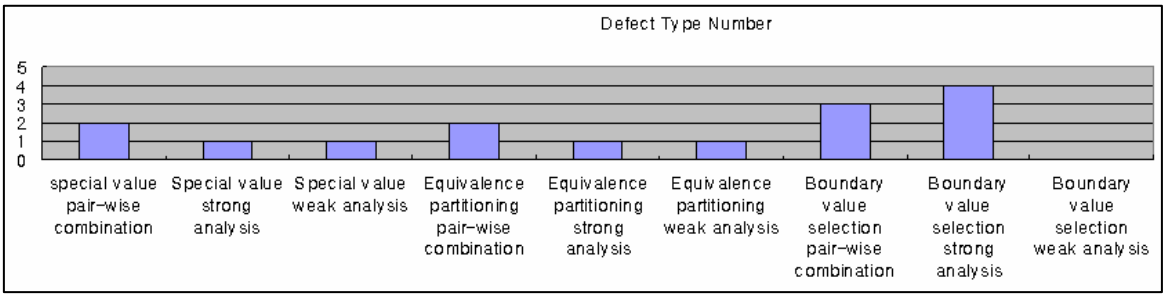
[1] http://en.wikipedia.org/wiki/Equivalence_partioning
[2] http://en.wikipedia.org/wiki/Equivalence_partioning#Boundary_value_analysis
[3] http://en.wikipedia.org/wiki/Cartesian_product
[4] Changhai Nie, Dept. of Computer Sci. & Eng. Southeast University, Automatic Test Generation for N-way Combinatorial Testing
[5] Yong Rae Kwon, Department of Computer Science Korea Advanced Institute of Science and Technology, Test Case Evaluation
[6] http://en.wikipedia.org/wiki/ID3
[7] Simeon Ntafos, University of Texas at Dallas, On Random and Partition Testing, Proceedings of the 1998 ACM SIGSOFT international symposium on Software testing and analysis, pp. 42-48, 1998
[8] http://www.pairwise.org
[9] David M. Cohen Siddhartha R.Dalal, Jesse Parelius and Gardner C.Patton, The Combinatorial Design Approach to Automatic Test Generation, IEEE Software, Vol. 13, No. 5, pp. 83-87, September 1996
[10] J. Rowland and Y. Zuyuan, Experimental comparison of three system test strategies, Proceedings of the ACM SIGSOFT '89 third symposium on Software testing, analysis, and verification, pp. 141-149, 1989

[표4] 음악 및 사진 등록 테스트 결과 예

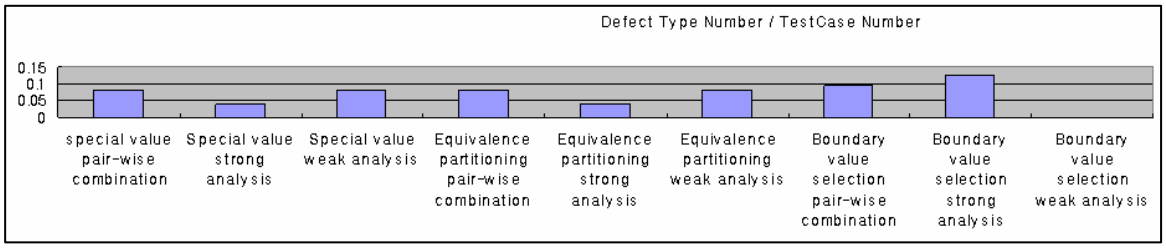
	special value analysis and pair-wise combination	Special value analysis and strong analysis	Special value analysis and weak analysis	Equivalence partitioning and pair-wise combination	Equivalence partitioning and strong analysis	Equivalence partitioning and weak analysis	Boundary value selection and pair-wise combination	Boundary value selection and strong analysis	Boundary value selection and weak analysis
결함 A	15/46	13/46	2/22	15/46	10/46	1/21	4/26	8/26	2/13
결함 B	7/46	3/46	0/22	7/46	4/46	0/21	3/26	1/26	0/13
결함 C	16/46	22/46	1/22	15/46	22/46	2/21	5/26	2/26	1/13
결함 D	0/46	0/46	0/22	0/46	0/46	0/21	2/26	0/26	0/13
결함 E	0/46	1/46	0/22	0/46	0/46	0/21	1/26	0/26	0/13
Defect Type Number	3	4	2	3	3	2	5	3	2
Defect Type Number / Test Case Number	0.065(3/46)	0.087(4/46)	0.09(2/22)	0.065(3/46)	0.065(3/46)	0.095(2/21)	0.192(5/26)	0.115(3/26)	0.154(2/13)
결함 F	0	0	0	0	0	0	0	(1/32)	0(0/16)
결함 G	0	0	0	0	0	0	(2/32)	(2/32)	0(0/16)
결함 H	(6/25)	(7/25)	(2/12)	(6/25)	(7/25)	(2/12)	(2/32)	(1/32)	0(0/16)
결함 I	(1/25)	0	0	(1/25)	0	0	(1/32)	(1/32)	0(0/16)
Defect Type Number	2	1	1	2	1	1	3	4	0
Defect Type Number / Test Case Number	0.08(2/25)	0.04(1/25)	0.083(1/12)	0.08(2/25)	0.04(1/25)	0.083(1/12)	0.094(3/32)	0.125(4/32)	0(0/16)
TOTAL, AVERAGE									
Defect Type Number	5	5	3	5	4	3	8	7	2
Defect Type Number / TestCase Number	0.0725	0.0635	0.0865	0.0725	0.0525	0.089	0.143	0.12	0.077



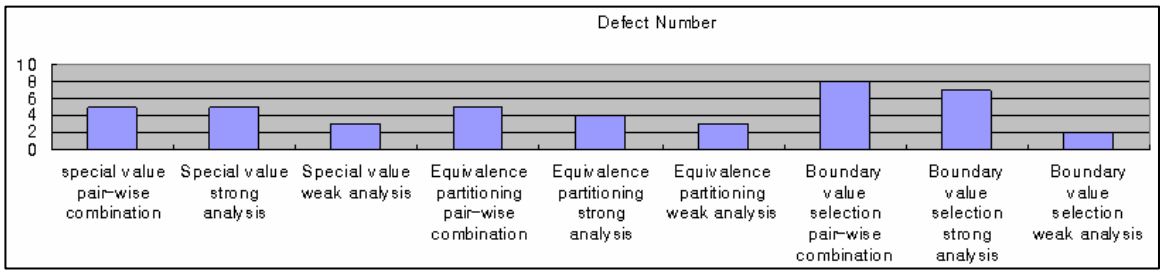
[그림3] 음악 프로그램에 대한 테스트 케이스당 발견된 결함 수



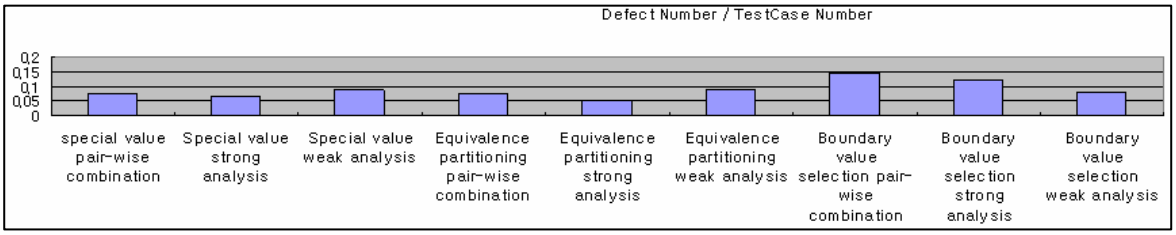
[그림4] 사진 프로그램에 대한 발견된 결함 수



[그림5] 사진 프로그램에 대한 테스트 케이스 당 발견된 결함 수



[그림6] 음악, 사진 프로그램에 대한 발견된 결함 종류의 수



[그림7] 음악, 사진 프로그림에 대한 테스트 케이스당 발견된 결함 종류의수