소프트웨어 블랙박스 테스팅 기법 비교 및 커버리지 분석을 통한 성능 측정

맹상우*, 박홍성**
강원대*, 강원대**

Comparing Black-box Testing Methodology and Performance Measurement by Test Coverage Analysis

Sang-Woo Maeng*, Hong-Seong Park**
Kangwon National University*, Kangwon National University**

Abstract - 정밀한 소프트웨어 테스팅에서 테스트 케이스의 생성과 테스트 수행 작업은 많은 시간과 노력을 필요로 한다. 따라서 경험 있는 테스터 들은 적은 수의 테스트 케이스를 선택적으로 사용하여 보다 정확한 테스트를 수행할 수 있기를 요구한다. 테스트 케이스의 수를 줄이기 위해 수많은 연구가 진행되었다. 소프트웨어 테스팅에서 가장 기본이 되는 테스트는 단위 테스트이다. 본 논문에서는 블랙박스 테스팅에 사용되는 잘 알려진 테스트 후보 값 생성 및 조합 기법에 관한 기존 연구를 살펴보고 성능을 비교해 본다. 성능 비교를 위해 몇몇 예제 코드를 실험적으로 이용하였다.

1. 서 론

신뢰성 있는 소프트웨어 개발을 위해서 거쳐야 하는 가장 중요한 단계가 소프트웨어 테스트이다. 소프트웨어 테스트는 범위에 따라 여러 단계로 나뉘는데 그중 가장 기본이 되는 것이 함수 테스트 혹은 단위 테스트이다[1]. 단위테스트는 크게 블랙박스 테스팅과 화이트 박스 테스팅으로 분류될 수 있다. 블랙박스 테스트는 기능 테스트나 행동 테스트라 부르기도 하는데 입출력 관계만을 고려하고 내부 상황에 대해서는 고려하지 않는 기법이다. 따라서 명세 기반의 테스팅 기법이 사용된다.

본 논문에서는 이 중 블랙박스 테스팅 기법에 초점을 둔다. 블랙박스 테스트 케이스 생성 관점을 크게 2가지로 분류하면 다음과 같다. 첫째, 테스트할 함수의 입력 파라미터들에 대한 후보값 선출 방법에 대한 관점과 둘째, 각 파라미터의 후보 값 조합기준의 관점으로 볼 수 있다.

전자의 경우 크게는 입력 가능한 도메인에서 무작위로 값을 선정하는 방법과 도메인을 서브 도메인으로 나누고 각 서브 도메인 내에서 어느 대표 값을 입력 값으로 정하는 동치분할 방법, 그리고 도메인의 경계부분을 집중적으로 테스트 하는 경계값 분석 테스팅 방법이 있다.
입력 값들을 조합하는 방법에는 모든 가능한 조합을 적용하는 All-pairs 조합 방법, Default 테스팅 방법 그리고 직교배열을 이용한 조합 방법 등이 있다.

본론에서는 이와 같이 잘 알려진 블랙 박스 테스팅 기법들을 조합하는 방법을 간략히 소개하고 이에 대해 몇몇 테스트 용 함수를 대상으로 커버리지를 측정하였다. 평가를 위해서 2가지 입력 패러미터 선정 방법에 대해 All-pairs조합과 직교배열 조합법을 적용하였을 경우를 측정하였다.

2. 기존 연구

2.1 블랙 박스 테스팅 기법

2.1.1 각 패러미터 후보 값 선출

입력 값 설정 방법에는 먼저 입력 가능한 도메인을 서브 도메인으로 나누고 각 서브 도메인 내에서 어느 대표 값을 입력 값으로 정하는 동등분할 방법이 있다. 분할된 영역의 모든 부분에 대해서는 시스템이 같은 동작을 수행한다고 기대하고 테스트 범위를 나눈다. 테스트 입력 값은 각 분할 영역을 대표할 수 있는 값으로 선정한다. 이렇게 설계된 테스트 케이스는 같은 특성을 가지면서 같은 방식으로 처리된다고 판단하는 모든 등가 집합에서 대표하는 입력 값들을 적어도 한 개씩은 사용하여 작성되었다는 것까지를 보장한다. 즉, 동등 분할은 특정 입력과 출력 커버리지를 달성하게 한다. 여기서 경험과 필요에 따라 하나 이상의 값을 선정하여 테스트 케이스를 작성하는 경우, 하나만 선정하여 테스팅하는 것보다 더 많은 결함을 발견할 수는 있지만 결과적으로 보장성은 동일하다.

두 번째로 경계 값 분석 방법이 있다. 동등분할의 경계부분에 해당되는 입력 값에서 결함이 발견될 확률이 경험적으로 높기 때문에 결함을 방지하기 위해 경계 값까지 포함하여 테스트하는 기법이다. 해당 분할영역의 최대 값과 최소 값은 그 영역의 경계 값이 된다. 경계 값 분석은 결함 발견율이 높고, 적용하기 쉬운 장점이 있어 가장 많이 사용되는 테스트 기법 중 하나이다. 이렇게 당연하고 간단한 방법을 굳이 테스트 기법으로 분류하는 이유는 경계 값을 입력 값으로 사용하여 테스트하였는데 거기에서 결함이 없었다는 것을 "보장"하기 때문이다.

위의 두 방법은 출력이 입력조건이나 변수들 사이의 관계에 따라 달라지는 경우, 입력 조건을 동등 분할하는 것이 매우 어려울 수 있다는 단점이 있다.

2.1.2 패러미터 간 후보 값 조합

앞 절에서 소개된 방법으로 선출된 패러미터 후보를 모든 경우의 수를 따져 조합하는 방법을 경우를 All-pairs라고 일컫는다. 이 방법을 적용하면 패러미터의 개수와 후보 수가 늘어남에 따라 생성해야 하는 테스트 케이스들의 수는 기하급수적으로 늘어날 수 있기 때문에 실제 상황에서는 가장 비효율적인 방법으로 분류된다.

테스트 케이스의 개수는 시간과 비용에 직결되는 중요한 요소이기 때문에 이를 줄이기 위한 많은 방법 들이 제시 되었다. 본 논문에서는 소프트웨어는 상대적으로 몇 안 되는 조건들의 조합들로 오류가 발생할 가능성이 높다는 특징에 착안하여 기존에 제안된 직교배열법을 이용해 패러미터 간의 후보 값을 조합하는 방법을 살펴본다.

Kuhn, Wallace, Gallo, Nair , Wallace[2]와 kuhn, Kuhn 등은 기 개발된 소프트웨어 오류 데이터를 분석한 후, 많은 오류가 2개 이상의 파라미터간 교호작용에 의해 파생되는 것을 밝혀내면서, 교호작용 테스팅의 당위성을 입증하였다. 교호작용 테스트 방법은 테스트 데이터의 갯수를 효과적으로 줄일 수 있는 장점이 있다. 파라미터 간 교호작용 강도 t를 갖는 t-way 테스팅을 통해 효과적으로 테스트 스위트를 줄이면서도 많은 오류를 검출할 수 있다. 그 상호관계를 갖는 인자의 개수에 따라 2개일경우 pair-wise, 3개 이상일 경우 t-way 조합이라 일컫는다. pair-wise의 경우 한 쌍의 입력 파라미터에 대하여 이 두 파라미터가 취할 수 있는 유효한 값들의 모든 조합이 최소 하나의 테스트 케이스에는 포함되도록 한다.

Myra B. Cohen 등은 시뮬레이티드 어닐링 알고리즘을 이용한 유연하고 확장 가능한 n-way 교호작용 테스팅을 소개하였다[3]. 이 테스팅 방법은 파라미터들이서로 다른 개수의 값을 가지고 있을 경우와 모든 파라미터에 대해서 n-way조합을 실시하는 동시에 몇 가지 파라미터에 대해서는 더욱 강한 강도의 조합을 설정할 수 있도록 지원한다.

2.2 커버리지 측정 기준

시스템 또는 소프트웨어의 구조가 테스트 케이스에 의해 테스트된 정도를 커버리지(Coverage)라 하며, 특정 구조의 종류에 대해 커버된 백분율로 표시한다. 예를 들어, 100% 결정 커버리지를 달성했다는 것은 코드의 구조 중 모든 결정 포인트(Decision points) 내의 전체조건식이 참 값과 거짓 값 모두를 갖도록 테스트가 되었다는 것을 의미한다.

본 논문에서는 일련의 과정에 의해 생성된 각각의 테스트 스위트에 대한 평가방법으로 분기 커버리지와 경로 커버리지 방법을 사용한다.

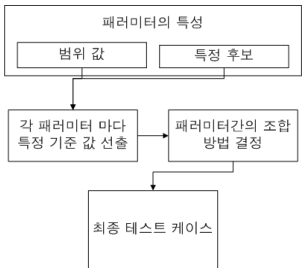
2.3 패러미터 후보 선정 및 조합

블랙박스 테스팅 환경에서 테스터는 주어진 변수에 대한 스펙, 즉 범위 값만을 알고 테스트에 임하게 된다. 실제 프로그램에서는 테스트 대상 함수의 패러미터 특성에 따라 크게 숫자의 범위 값으로 주어지는 타입과 몇 가지 후보 값으로 제한된 입력을 갖는 타입이 있다.

만일 범위 값에 대한 테스트에서 입력 값을 설정하기 위해 사용한 테스팅 기법을 사용하지 않고 모든 범위에 걸쳐 모든 조합을 생성해 테스트 하는 것은 천문학적 비용이 따르게 된다. 따라서 테스트 케이스의 수를 줄이기 위해 2절에서 설명한 범위의 기준에 따른 선정 방법과 직교배열 조합 방법을 사용하게 된다.

<표 1> 각 실험 함수에 대한 생성 및 평가방법에 따른 평가표

함수명 \ 분류		All combination			Pair-wise		
		테스트케이스 개수	분기커버리지	경로커버리지	테스트케이스 개수	분기커버리지	경로커버리지
mid	동치분할	125	10 / 10	6 / 6	26	10 / 10	**5 / 6**
	경계값테스트	216	10 / 10	6 / 6	38	10 / 10	6 / 6
test	동치분할	625	8 / 8	12 / 12	32	8 / 8	**10 / 12**
	경계값테스트	1296	8 / 8	12 / 12	45	8 / 8	**9 / 12**



<그림 1> 테스트 케이스 선출 과정

다음 절에서는 위의 프로세스에 따라 생성된 테스트 케이스의 절감 효율을 확인하고 커버리지 측정을 통하여 에러 검출 성능을 측정, 비교하여 본다.

3. 테스트 및 평가

본 논문에서는 비교를 위해 작성된 2가지의 함수를 대상으로 하여 테스트를 수행하였다. 각 함수의 코드를 본 논문에서는 Pair-wise 조합을 생성하기 위해 PICT라는 툴을 사용하였다[9].

mid함수는 입력된 세 숫자중 가운데 값을 리턴하는 함수로 총 3개의 패러미터를 갖는다. 각각의 패러미터의 입력범위는 -10 ~ 10으로 제한되어 있다. 각 패러미터에 대해 동치분할과 경계값 분석방법을 적용하여 입력 후보 값을 선출한 후 각 패러미터 간 후보 값에 대한 조합에 있어 All-pair 조합과 Pair-wise 조합으로 테스트 케이스를 생성한 결과의 개수를 표 1에 표시 하였다. 생성된 테스트 케이스의 개수는 All-pair 조합에 비해 Pair-wise 조합의 경우가 훨씬 적게 생성된 것을 확인할 수 있다. 이와 동시에 분기 커버리지 면에서도 거의 동등한 커버리지 수치를 보여주고 있다. 단 모든 경로에 대해 프로그램의 수행여부를 확인하는 경로 커버리지의 경우는 약 80%에 달하는 것을 확인할 수 있다.

<표 2> test 함수 코드

```
// a: <-10..10>,
// b: <-10..10>,
// c: <-10..10>,
// d: <-10..10>
void test(int a,int b,int c,int d)
{
  NODE(0)
  int x=0,y=0;

  NODE(1)
  if ( a > 0 )
  {
    x = 2;          NODE(2)
  }
  else
  {
    x = 5;          NODE(3)
  }

  NODE(4)
  if ( b > 0)
  {
    y = 1 + x       NODE(5)
  }

  NODE(6)
  if ( c > 0)
  {
    NODE(7)
    if ( d > 0)
    {
      output(x); NODE(8)
    }
    else
    {
      output(10);NODE(9)
    }
  }
  else
  {
    output(1/(y-6));
    NODE(10)
  }

  NODE(11)
}
```

두 번째 test 함수의 경우도 마찬가지 방법으로 각 패러미터 마다 두 가지 방식에 의해 후보 패러미터 값을 선출한 뒤 조합을 이용해 테스트 케이스를 생성하여 결과를 표 1에 표시하였다. 마찬가지로 경로 커버리지의 경우에 있어서 약 80% 밖에 미치지 못하였다.

여기서 주목해야 할 점은 test 함수를 흐름제어그래프(CFG)로 표현하였을 때 10번 노드에서는 0으로 나누는 에러를 발생할 소지를 갖고 있다는 사실이다. 노드 번호 0-1-3-4-5-6-10-11을 거쳐 가는 테스트 케이스 { 0, 10, -10, 10 }의 경우 10번 노드에서 0으로 나누는 에러를 발생시키는 위험에 노출시켜 에러를 검출해 낼 수 있다. 하지만 Pair-wise 조합 생성시 다른 테스트케이스와 중복성에의해 이 테스트 케이스가 제거된다면 에러 검출확률이 떨어지게 되는 단점을 갖고 있다. 실제로 실험에서 PICT를 이용한 테스트 케이스 생성 결과 이 테스트 케이스가 누락되어 있어 에러를 검출할 수 없었다.

4. 결 론

본 논문에서는 블랙박스 테스트에서 테스트 케이스 생성 과정에 2가지 방법을 분류하고 테스트 대상 코드에 대표적인 기법 2가지씩을 직접 적용하여 효율성을 실험적으로 확인해 보았다. 이 두 가지 종류의 기법을 통해 테스트 케이스의 개수는 크게 감소 시킨 반면 커버리지는 거의 감소하지 않는 효과를 확인할 수 있었다.

그러나 Pair-wise에 의해 제외된 테스트 케이스 내에 위험요소가 포함되어 있을 경우 에러 검출확률이 떨어진다는 단점을 드러내기도 하였다. 이러한 위험요소는 Pair-wise를 단위테스트에 적용했을 때의 단점뿐만 아니라 블랙박스 테스트의 한계점을 동시에 드러내고 있다. 이러한 에러는 함수 내부의 데이터 흐름을 모르는 상황에서는 알 수 없기 때문이다.

화이트 박스 테스팅의 경우 대규모화 되어가는 소프트웨어 테스트에 적용하기 힘든 점이 있다. 그리고 기존에 나온 컴포넌트나 라이브러리 등의 재활용이 높아지고 있는 시점에서 블랙박스 테스팅은 그 중요성이 점점 강조되어 갈 것이다.

본 논문에서 살펴본 테스트 케이스 생성 방법은 테스터가 입력 패러미터의 조건만 명시해준다면 자동화시키기 매우 용이하다. 앞으로 이러한 일련의 과정을 자동화 시켜 테스트 케이스를 생성시키는 툴을 제작하는 것이 향후과제로 남겨져 있다.

[참 고 문 헌]

[1] 이유정, 최승훈, "XML 을 이용한 명세 기반 단위 테스트 자동화 도구", 한국인터넷정보학회 2002 추계학술발표대회 논문집 제3권 제2호, 2002. 11

[2] D. Richard Kuhn, Dolores R. Wallace, Albert M. Gallo Jr. “Software Fault Interactions and Implications for Software Testing,”IEEE transactions on Software Engineering, vol. 30, no. 6, June 2004.

[3] M. B. Cohen, C. J. Colbourn, J.S. Collofello, P. B. Gibbons and W. B. Mugridge, “Variable Strength Interaction Testing of Components,” In Proc. of the Intl. Computer Software and Applications Conference, (COMPSAC 2003), Dallas TX, 2003.

[4] 고병각, 이상용, 장중순,"소프트웨어의 선택적 교호작용 테스팅", 32회 추계학술발표대회 논문집 2005. vol.32, no.2

[5] 이경환, 김정종 “데이터 플로우 커버리지를 이용한 테스트 데이터 생성의 자동화에 관한 연구”, 한국정보과학회 논문지 2. 1998

[7] 심관보, 송영지, “분기기법을 이용한 테스트 경로 선택 알고리즘에 관한 연구”, 전자계산반도체 재료 및 부품합동학술회의 발표회 학회지. 1991.

[8] 박영석, 박용진, “컴퓨터 프로그램 테스팅을 위한 테스트 케이스 선택기준과 커버리지 척도”, 한국정보과학회 논문지.

[9] PICT (Pair-wise combination generation tool) http://blogs.msdn.com/nagasatish/archive/2006/11/30/pairwise-testing-pict-tool.aspx