뮤테이션 기법을 이용한 컴포넌트의 테스팅

마유승○ 장윤규 권용래
한국과학기술원 전산학과

Testing of Adapted Component Using Mutation Techniques

Yu Seung Ma, Yoon Kyu Jang and Yong Rae Kwon
Department of Computer Science, KAIST

요 약

컴포넌트 기반 시스템 개발 과정은 크게 컴포넌트 선택,변용,통합의 세 단계로 이루어진다. 이 중 컴포넌트 변용은 범용목적으로 개발된 컴포넌트를 자신의 시스템의 구조에 맞도록 변경하는 과정으로 변용이 끝난 컴포넌트는 사용자 입장에서 제대로 동작하는지 테스트 되어야 한다. 이 논문에서는 변용이 끝난 컴포넌트의 뮤테이션 테스트 기법을 제시한다. 이를 위해 먼저 컴포넌트의 변용 유형을 살펴보고 여기서 나타날 수 있는 오류의 형태를 살펴본 뒤 이를 발견할 수 있는 뮤테이션 변환 연산자를 정의한다.

1. 서론

컴포넌트 소프트웨어 기술이 발전함에 따라 프로그래머가 모든 구현을 완전히 새로 할 필요없이 컴포넌트의 조립방식을 통해 프로그램을 작성할 수 있게 되었다. 재사용과 생산성 향상을 가져올 것으로 기대되는 이러한 패러다임은 소프트웨어의 규모와 복잡도가 증가하면서 많은 관심을 끌고 있다.

컴포넌트는 보통 실행가능한 OTS(off-the-self)형태이고 이를 이용한 시스템 개발 과정은 크게 컴포넌트 선택 변용,통합의 세 단계로 이루어진다 [2] 이 중 컴포넌트 변용은 범용목적으로 개발된 컴포넌트를 자신의 시스템 구조에 맞도록 변경하는 과정으로 변용이 끝난 컴포넌트는 제대로 동작하는지 사용자 입장에서 테스트 되어야 한다

이 논문에서는 변용이 끝난 컴포넌트의 테스트에 뮤테이션 기법[1]을 적용시키려 한다. 뮤테이션 기법이란 테스트 하려는 프로그램 P가 있을 때 이를 약간 변형한 뮤턴트 프로그램 P'을 만들어 P와 P'을 차별화 할 수 있는 테스트 데이타를 선택하는 기법이다

변용이 끝난 컴포넌트는 사용자가 변경할 수 있는 사용자 코드와 사용자의 접근이 허용되지 않는 블랙박스 형태의 컴포넌트로 구성된다. 이 중 사용자 코드는 접근이 가능하므로 뮤테이션 연산자를 적용하여 뮤턴트 프로그램을 만들어 낼 수 있지만 블랙박스 형태의 컴포넌트는 사용자가 컴포넌트의 원시코드에 접근할 수 없기 때문에 컴포넌트 내부에 뮤테이션 변환 연산자를 적용시킬 수 없다. 원시코드에 접근할 수 없는 상황에서 컴포넌트로의 접근은 인터페이스를 통해서만 가능하다. 따라서 이 경우 인터페이스를 이용한 뮤테이션 기법[4]을 적용시킬 수 있다.

이 논문의 구성은 다음과 같다. 먼저 2장에서 컴포넌트의 변용 유형에 대해 살펴본 뒤 3장에서는 변용이 끝난 컴포넌트에서 발생할 수 있는 오류와 이러한 오류를 검출할 수 있는 오류삽입 연산자를 정의한다. 4장에서는 이 논문에서 제시한 내용을 예제에 적용시켜본 뒤 5장에서 결론 및 향후 연구 방향을 제시한다.

2. 컴포넌트의 변용

컴포넌트 기반 소프트웨어 개발과정은 크게 컴포넌트 선택, 변용, 통합의 세 단계로 이루어진다[2]

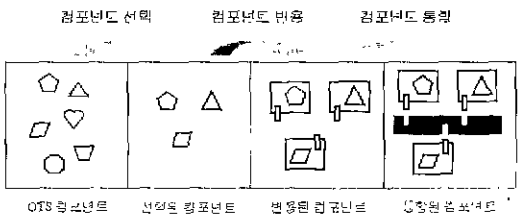

그림 1. 컴포넌트 기반 소프트웨어 개발과정

컴포넌트 선택 과정에서는 시중에 나와있는 컴포넌트 중 자신의 시스템에 맞는 컴포넌트를 찾는다. 하지만 이러한 컴포넌트들은 대부분 다수의 사용자와 다양한 환경을 위해 만들어졌으므로 자신의 시스템에 완벽하게 맞지 않는 경우가 많다. 따라서 사용자는 자신이 만드려는 시스템의 구조에 맞게 컴포넌트를 변형시켜야 하는데 이러한 작업을 컴포넌트의 변용[3]이라한다. 이 경우 사용자는 컴포넌트의 내부에 접근할 수 없으므로 사용자 코드로 컴포넌트의 외부를 덧싸는 형식으로 변용이 이루어진다. 마지막으로 통합과정에서는 변용이 끝난 컴포넌트들을 연결하여 시스템을 완성한다.

2.1 컴포넌트 변용의 유형

앞에서 언급한 것처럼 컴포넌트의 변용은 컴포넌트의 외부를 덧싸는 형식으로 이루어진다. 아래에서 컴포넌트 변용의 유형에

대해 살펴본다.

- 컴포넌트 속성의 변경

  주어진 컴포넌트에서 변용이 허용되는 속성의 값을 바꾼다 컴포넌트에서 접근이 허용된 경우 직접 속성을 바꾸거나 컴포넌트에서 제공된 서비스를 이용하여 바꾼다

  UI컴포넌트의 색이나 가로길이 속성을 바꾸는 SetColor(red)나 SetWidth(100) 같은 서비스 호출을 예로 들 수 있다.

- 컴포넌트 서비스 환경의 변경

  컴포넌트가 제공하는 서비스의 기본적 기능은 변경시키지 않으면서 약간의 부수적인 코드를 추가하여 서비스 환경을 바꾼다 서비스 이름 변경, 서비스 영역 변경, 호출 상태 설정등이 유형에 속하며 표 1에 간단한 예가 나타나있다.

서비스 이름 변경	IntPush → Push
서비스 영역 변경	Push(float x) → Push(int a)
호출 상태 설정	Push(a) → if(!ISFull()) Push(a)

표 1. 컴포넌트 서비스 환경 변경 예제

- 컴포넌트 서비스의 조합

  컴포넌트가 제공하는 여러가지 서비스를 조합하여 하나의 복잡한 서비스를 만든다.

  예를 들어 사칙연산의 기능을 제공하는 컴포넌트가 있을 때 입력값의 세배값을 돌려주는 triple(x)는 y=mul(x,x), result=mul(y,x)처럼 곱셈 서비스를 두번 호출하여 계산할 수 있다 그림 3의 Reverse()서비스는 스택 컴포넌트의 IntPop(),IntPush()서비스들과 큐 컴포넌트의 EnQueue()DeQueue()서비스들의 조합으로 이루어졌다.

- 컴포넌트의 연결

  컴포넌트들간의 연관관계를 설정해 준다 연관관계가 설정된 후 이들사이의 통신은 사용자의 간섭없이 컴포넌트 내부에서 자동적으로 행해진다.

  JavaBeans의 경우 이벤트 e를 발생시키는 컴포넌트 A가 있을 때 이 이벤트의 리스너를 컴포넌트 B로 하는 연관관계를 설정하여 주면 이벤트 e가 일어날 때마다 이벤트를 처리하는 과정이 컴포넌트 A와 B사이에서 자동적으로 수행된다[8]

- 사용자 코드의 추가

  컴포넌트에서 제공된 서비스 이외에 사용자가 필요로 하는 속성이나 서비스를 추가로 정의한다

3. 변용이 끝난 컴포넌트를 위한 뮤테이션 테스트 기법

뮤테이션 테스팅을 하기 위해서는 검출하고자 하는 오류의 형태를 파악하고 이에 마추어 뮤테이션 변환 연산자를 고안해야 한다 아래에서는 변용이 끝난 컴포넌트에서 발생할 수 있는 오류의 형태와 이를 위한 뮤테이션 변환 연산자를 정의한다

3.1 변용된 컴포넌트의 오류

변용이 끝난 컴포넌트에서의 오류는 크게 두가지로 나뉘어 진다 첫째는 사용자가 변용을 잘못한 경우이며 두번째는 사용하는 컴포넌트 자체에 오류가 있는 경우이다 앞에서 살펴본 것처럼 컴포넌트 변용은 컴포넌트의 인터페이스를 이용하여 이루어진다. 따라서 사용자가 컴포넌트의 인터페이스를 제대로 이해하지 않은채 서비스를 이용한다면 잘못된 값을 컴포넌트에게 전달하거나 컴포넌트 서비스들을 바른 순서로 호출하지 않는 등의 오류를 범할 수 있다

컴포넌트 자체에 오류가 있는 경우 이 오류의 결과가 변용된 컴포넌트에 영향을 미친다 따라서 사용자가 변용을 올바르게 하였다 하더라도 컴포넌트 내부의 오류에 의해 그릇된 행위를 수행할 수 있다.

3.2 뮤테이션 변환 연산자

변용이 끝난 컴포넌트는 수정이 가능한 사용자 코드와 사용자가 접근할 수 없는 블랙박스 형태의 컴포넌트로 구성된다 이 중 사용자 코드는 원시코드에 접근할 수 있으므로 뮤테이션 연산자를 적용시킬 수 있지만 블랙박스의 컴포넌트는 사용자가 원시코드에 접근할 수 없기 때문에 컴포넌트 내부를 변경할 수 없다. 원시코드에 접근할 수 없는 상황에서 컴포넌트로의 접근은 인터페이스를 통해서만 가능하다 따라서 이 경우 인터페이스에 뮤테이션 기법을 적용시킬 수 있다

표 2에서는 컴포넌트 인터페이스를 고려하여 변용이 끝난 컴포넌트에 적용시킬 수 있는 뮤테이션 변환 연산자를 정의한다

CPC	컴포넌트 속성 변경 예) SetColor(red) → SetColor(yellow)
APC	사용자 속성 변경 예) width=10 → width=15
SDC	서비스 영역 변경 예) Fun(int) → Fun(float)
SCC	서비스 호출 상황 변경 예) if(a>5) Fun() → if(a>6) Fun()
SSC	서비스 호출 순서 변경 예) A(),B(),C() → A(),C(),B()
IVC	서비스 입력값의 변경 예) Fun(a) → Fun(a+1)
ISC	서비스 입력값의 순서 변경 예) Fun(a,b) → Fun(b,a)
OVC	출력값의 변경 예) result=Fun() → result=Fun(); result++;
OVD	출력값의 삭제 예) result = Fun() → Fun()

표 2 뮤테이션 변환 연산자

4. 적용 예제

여기서 만드려는 컴포넌트는 스택의 기본적인 서비스인 Push, Pop이외에 스택 원소의 순서를 바꾼다거나 원하는 원소를 삭제하는 등의 기능이 추가된 정수형 스택이다 변용에 이용되는 컴

포넌트는 문자열 타입의 원소를 갖는 기본적인 스택 컴포넌트와 타입 변환 컴포넌트, 그리고 큐 컴포넌트이고 이들의 인터페이스는 그림 2에 나타나 있다. 그림 3에서는 이들을 이용한 변용된 컴포넌트의 사용자 코드를 보여준다.

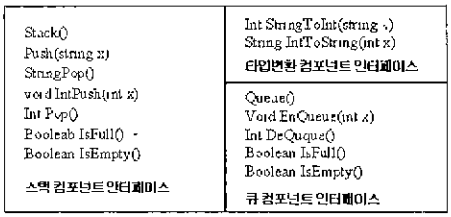

그림 2· 스택,타입변환,큐 컴포넌트의 인터페이스

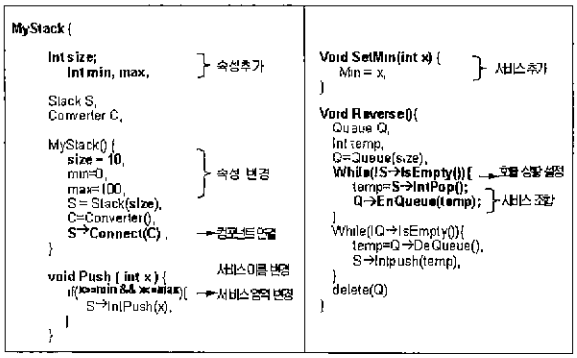

그림 3· 변용된 컴포넌트 MyStack

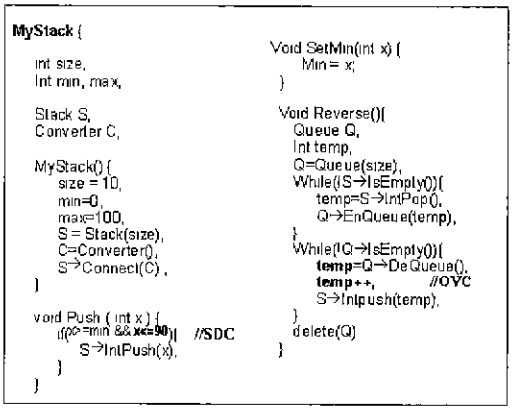

그림 4: 뮤테이션 변환 연산자 SDS의 OVC를 적용한 예

그림 4은 그림 3에 SDC, OVC 뮤테이션 변환 연산자를 적용시킨 것이다. 이 경우 Stack(),Push(1);Push(93),Pop() 을 수행시켜 서로 다른 결과를 내는 테스트 케이스를 찾을 수 있다.

5. 결론 및 향후 연구

이 논문에서는 변용이 끝난 컴포넌트를 위한 뮤테이션 테스트 기법을 제시하였다 이를 위해 먼저 컴포넌트 변용의 형태에 대해 조사하였고 이러한 변용으로 발생할 수 있는 오류와 이 오류들을 발견할 수 있는 뮤테이션 변환 연산자를 정의하였다

뮤테이션 테스트 기법은 오류 검색 효과가 뛰어나지만 테스트 시 수행시켜야 할 뮤턴트 프로그램의 수효가 큰 단점을 갖고 있다. 하지만 이 논문에서 제시하는 뮤테이션 기법은 뮤테이션 변형 연산자를 변용이 끝난 컴포넌트의 사용자 코드에 국한시킴으로서 이 문제를 해결하고 있다

향후 과제로는 컴포넌트가 주어졌을때 자동적으로 테스트 케이스들을 뽑아내는 도구를 구현할 계획이다

참고 문헌

[1] Timothy A Budd, *Mutation Analysis·Ideas,Esamples,Problems and Prospects, submitted on IEEE Software* 1996

[2] Alan W. Brown and Kurt C Wallnau, *Engineering of Component-Based System, Computer Program Testing* pp 129-148,1991

[3] George T. Heineman, *Adaptation of Software Components, Computer Scienve Department*, WPI-CS-TR-99-04

[4] Marcio E Delamaro, Jose C. Maldonado and Aditya P. Mathur *Integration Testing Using Interface Mutations*,1997

[5] Gary McGraw and John Viega, *Why COTS Software Increase Security Risks, In ICSE Workshop on Testing Distributed Coponent-Based Systems* 17 May 1999

[6] Sundiotp Ghosh, Aditya P. Mathur, *Issues in Testing Distributed Component-Based System, In ICSE Workshop on Testing Distributed Coponent-Based Systems* 17 May 1999

[7] Weyuker. E. J , *Testing Component-Based Software : A Cautionary Tale, IEEE Softwar*,pp.54-59 Sep/Oct, 1998

[8] David Krieger, Richaid m. Aldler, *the Emergence of Distributed Component Platforms, IEEE Softwar*,pp.43-53.March, 1998

[9] Jan Bosch, *Adapting Object-Oriented Components*

[10] Jan Bosch, *Superimposition : A Component Adaptation Technique*,