1. 서론

컴포넌트 기반 시스템 개발 과정은 크게 컴포넌트 선택, 복용, 통합의 세 단계로 이루어진다. 이 중 컴포넌트 복용은 병용목적으로 가로зн된 컴포넌트를 자신의 시스템 구조에 맞도록 변경하는 과정으로 복용이 곧난 컴포넌트는 사용자 입장에서 채택이 될지 테스트 되어야 한다. 이 논문에서는 복용이 곧난 컴포넌트의 유효성 테스트 기법을 제시한다. 이를 통해 곧난 컴포넌트의 복용 유형을 살펴보고 여기서 나타날 수 있는 오류의 형태를 살펴본 뒤 이를 발견할 수 있는 유래성 발견을 정립한 다.

2. 컴포넌트의 복용

컴포넌트 기반 소프트웨어 개발과정은 크게 컴포넌트 선택, 복용, 통합의 세 단계로 이루어진다. 그림 1은 컴포넌트 선택과 시스템 구조에 맞도록 변경하는 과정으로 복용이 곧난 컴포넌트의 테스트 방법에서 사용자 입장에서 테스트 되어야 한다.

이 논문에서는 복용이 곧난 컴포넌트의 테스트에 유효성 기법을 적용시키기 한다. 유효성 기법이란 테스트 차원의 프로그램 P'가 있을 때 이를 약간 변경한 유래 체크 프로그램 P을 만들어 P와 P'의 차단을 확인할 수 있는 테스트 데이터를 선택하는 기법이다.

복용이 곧난 컴포넌트는 사용자가 변경할 수 있는 사용자 코 드와 사용자의 접근이 허용되지 않는 플랫폼 형태의 컴포넌트로 구성된다. 이 중 사용자 코드는 접근이 가능하므로 유래성 인자로 제공하여 컴포넌트의 보기가 없을 수 있지만 플랫폼 형태의 컴포넌트는 사용자가 컴퓨터의 원시코드에 접근할 수 없기 때문에 컴퓨터 내부에 유래성 변환 인자를 적용시킬 수도 없다. 원시코드에 접근할 수 없는 상황에서 컴포넌트의 접근은 인터페이스를 통해서만 가능하다. 따라서 이 경우 인터페이스를 이용한 유래성 기법을 적용시킬 수 있다.

이 논문의 구성은 다음과 같다. 먼저 2장에서 컴포넌트의 복 용 유형에 대해 살펴본 뒤 3장에서는 복용이 곧난 컴포넌트에서 발생할 수 있는 오류와 이러한 오류를 검출할 수 있는 유래성 발견을 정립한다.
3.1 변용된 컴퓨터드의 오류

변용이 끝난 컴퓨터드에서의 오류는 크게 두가지로 나뉘어 진다.
첫째는 사용자가 변용을 잘못한 경우이며 두번째는 사용하는 컴퓨터드 자체의 오류가 있는 경우이다. 앞에서 설명한 것처럼 컴퓨터드 변용은 컴퓨터드의 인터페이스를 이용하여 이루어진다.
따라서 사용자가 컴퓨터드의 인터페이스에 대해 이해하지 않은 것은 컴퓨터드를 사용한다는 장점만 있는 컴퓨터드에게 짧은거야 컴퓨터드에서 전달하게
나 컴퓨터드 서비스들을 빠른 순서로 호출하지 않는 등의 오류
을 발생시킨다.

 컴퓨터드 자체에 오타가 있는 경우 이 오타의 결과가 변용된 컴퓨터드에 영향을 미친다. 따라서 사용자가 변용을 올바르게 하였으면 하러라도 컴퓨터드 내부의 오류에 의해 잘못된 결과를 보게 될 수 있다.

3.2 유래선 변환 연산자

변용이 끝난 컴퓨터드는 수행된 사용자 코드와 사용자가 사전에 입력한 컴퓨터드의 인터페이스를 이용하여 컴퓨터드에 적용할 수 있다. 또한 컴퓨터드는 인터페이스에 대해 컴퓨터드의 특성을 지는 컴퓨터드의 인터페이스를 통해서만 가능하리라. 따라서 이를 컴퓨터드의 인터페이스를 적용시킬 수 있다.

표 2에서는 컴퓨터드의 인터페이스를 고려하여 변용이 끝난 컴퓨터드에 적용시킬 수 있는 컴퓨터드의 변환 연산자를 정의한다.

<table>
<thead>
<tr>
<th>연산자</th>
<th>세부 기능</th>
</tr>
</thead>
<tbody>
<tr>
<td>Int</td>
<td>수치 연산</td>
</tr>
<tr>
<td>Float</td>
<td>수치 연산</td>
</tr>
<tr>
<td>String</td>
<td>문자 연산</td>
</tr>
</tbody>
</table>

3. 변용이 끝난 컴퓨터드를 위한 모듈레이션 테스트 기법

모듈레이선 테스트라는 것은 컴퓨터드의 호환성 및 확인을 위한 테스트로서 컴퓨터드의 드라이버를 사용하여 컴퓨터드의 품질을 향상시킬 수 있는 방법이다.

4. 적용 예제

여기서 만드는 컴퓨터드는 다른 컴퓨터드의 서비스를 사용하여 싱글 태스팅을 수행한다. 여기서는 컴퓨터드의 이벤트를 단순화하거나 단순화하기 위한 컴퓨터드를 가지고 컴퓨터드의 이벤트를 단순화하기 위한 컴퓨터드를 완벽하게 수행한다.
포몬트는 문자열 탐색의 원소를 갖는 기본적인 스택 컴퓨터 프로그램의 타입 반환 포몬트, 그리고 스택 포몬트리인 이름을 인터페이스는 그림 2에 나타나 있다. 그림 3에서는 이들을 이용한 변형된 포몬트의 사용자 코드를 보여준다.

![포몬트 코드 예시](image)

그림 2. 스택, 탐색, 반환, 큐 포몬트의 인터페이스

<table>
<thead>
<tr>
<th>스택</th>
<th>Queue</th>
<th>Int Stack(Typ) (Int typ) (Int typ) (Int typ) (Int typ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Int</td>
<td>void</td>
<td>EStack (Int typ) (Int typ) (Int typ) (Int typ)</td>
</tr>
<tr>
<td>Boolean</td>
<td>Int</td>
<td>Empty (Int typ) (Int typ) (Int typ) (Int typ)</td>
</tr>
<tr>
<td>스택 검색 테이블 인터페이스</td>
<td>Queue 테이블 인터페이스</td>
<td></td>
</tr>
</tbody>
</table>

![포몬트 코드 예시](image)

그림 3. 변형된 포몬트 MyStack

조사하였고 이러한 변형으로 발생할 수 있는 오류와 이 오류들을 발견할 수 있는 뮤테이션 변환 연산자를 정의하였다. 뮤테이션 테스트 기법은 오류 검출 효과가 높아져야만 테스트시 수행시키어야 할 뮤테이션 프로그램의 수행이 큰 단점을 갖고 있다. 하지만 이 논문에서 제시하는 뮤테이션 기법은 뮤테이션 변형 연산자를 병합한 포몬트의 사용자 코드에 구현하므로 이 문제를 해결하고 있다.

향후 과제로는 포몬트가 주어졌을 때 자동적으로 테스트 케이스들을 뽑아내는 도구를 구현할 계획이다.

참고 문헌


[9] Jan Bosch, Adapting Object-Oriented Components