아키텍처 모델 기반의 유닛 테스트 자동 생성 방법

윤석진[O] 이승연 정양재 신규상
한국전자통신연구원
{sjyoon[O], coral, cornor, gsshin }@etri.re.kr

Method of Unit Test Driver Automatic Generation Based on Architecture Model

Seokjin Yoon[O] Seungyeon Lee, Yangjae Jeong, Gyusang Shin
Electronics Telecommunications Research Institute

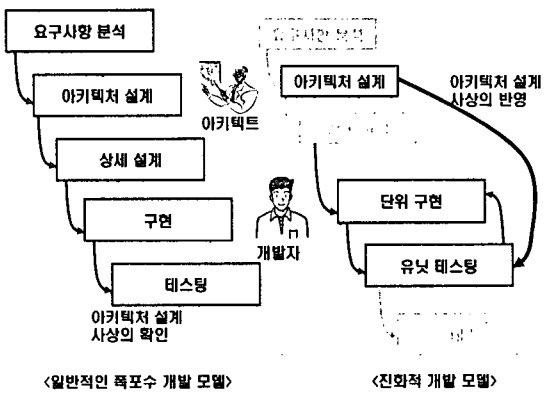
요 약

본 연구는 컴포넌트로 구성된 일반 아키텍처 모델에서 개별 컴포넌트의 기능성을 컴포넌트 개발 시에 확인할 수 있도록 유닛 테스트 기반의 테스트 드라이버 코드를 자동으로 생성하는 장치 및 방법에 관한 것이다. 즉, 본 연구는 아키텍처 모델에서 컴포넌트의 인터페이스에 대해서 아키텍처 설계자가 인터페이스에 대한 예상 기대값, 컴포넌트의 상태 정보, 특정 시점에서의 인터페이스의 입력정보들을 입력하면 이 정보를 이용하여 테스트를 수행하기 전에 컴포넌트의 상태를 설정하는 테스트 준비 코드와 테스트를 수행한 후에 발생하는 결과값과 예상 기대값을 비교하여 확인하게 하는 테스트 확인 코드를 포함하는 테스트 드라이버 코드를 자동으로 생성한다. 본 연구에 의하면; 아키텍처 설계 단계에서 아키텍처 설계자가 컴포넌트 개발자에게 테스트 드라이버 코드를 제공하게 함으로써 아키텍처 설계에서 요구하는 컴포넌트의 기능이 개별 컴포넌트 별로 제대로 개발되는지 개발시에 자동으로 검증하게 할 수 있다.

1. 서 론

소프트웨어 시스템을 설계하기 위해서는 일반적으로 블록 다이어그램 혹은 컴포넌트 다이어그램 형태의 아키텍처 모델을 작성한다. 소프트웨어 시스템은 이러한 아키텍처 모델에 따라서 여러 개의 컴포넌트로 나뉘어지며 컴포넌트들간의 상호작용을 통해서 소프트웨어 전체 시스템이 동작한다. 아키텍처 모델은 적용 프로젝트의 편의에 따라 다양한 아키텍처 스타일이 사용되지만 일반적으로 공통적인 것은 소프트웨어 모듈로 나뉘어질 수 있는 컴포넌트와 컴포넌트들간의 메시지를 주고 받을 수 있는 인터페이스간의 연결로 표현된다.

개발 방식에는 일반적으로 폭포수(Waterfall) 방식의 테스트와 진화적(Evolutionary) 방식이 있다. 폭포수 형은 요구분석, 설계, 코딩, 최종 테스트, 배포의 단계를 거치는 형태로서 일반적인 개발 절차에 따라서 테스트가 진행된다. 이 방식의 단점은 테스트 초기에 문제점을 식별하기 어렵다는 점이다. 진화적 방식은 시스템을 모듈 단위로 구현을 하고 테스트하고 수정하는 방식으로서 초기에 테스트 케이스를 작성함으로서 오류를 사전에 예방하는 효과가 있다.



<그림 1> 폭포수 개발 모델과 진화적 개발 모델

<그림 1>에서 표현하는 바와 같이 일반적 폭포수 개발 모델에서는 아키텍트의 설계가 제대로 구현되었는지의 확인을 구현 단계를 거쳐 테스팅 단계에 이르러 확인할 수 있다. 하지만 본 연구에서 제시하는 개발 방법에서는 아키

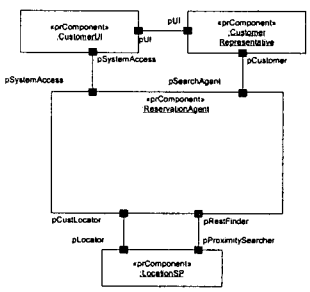
텍트의 설계의도가 반영된 테스트 케이스를 초기 구현단계에 제공함으로써 개발자가 설계 요구를 만족시키면서 개발할 수 있도록 강제하는 효과가 있다.

일반적으로 유닛 테스트 프레임워크와 같은 유틸리티를 사용하면 테스트를 진행하며 개발 과정 중에 올바르게 개발되었는가를 확인하면서 개발할 수 있는 장점이 있다. 하지만 초기에 테스트 케이스의 작성의 어려움이 있어서 유닛 테스트 작성에 익숙하지 않은 개발자는 접근하기 어려운 단점이 있다.

본 연구에서는 소프트웨어 시스템의 설계시 설계에 사용된 컴포넌트들이 만족해야 하는 설계자의 요구사항을 테스트 드라이버 형태로 생성하여 개발자에게 전달하여 설계자의 요구사항을 만족하는 시스템을 개발할 수 있도록 하는 테스트 드라이버 자동 생성 장치 및 방법을 제공하고자 한다.

2. 본 론

본 연구에서는 아키텍처 모델의 개별 컴포넌트의 인터페이스에 특정한 스냅샷 정보를 입력함으로써 컴포넌트의 상태정보를 저장하는 상태 변수를 추출하고 상태 변수를 설정 및 확인하는 인터페이스를 추가함으로써 컴포넌트의 유닛 테스트가 원활하게 이루어질 수 있도록 하며 상태 변수에 접근하는 인터페이스를 이용하여 테스트 환경을 설정하는 테스트 준비 코드와 인터페이스 수행 후에 적절한 상태에 이르렀는지 확인하는 테스트 확인 코드를 생성하여 테스트 드라이버 코드를 자동으로 생성하도록 한다.



<그림 2> 식당 예약 시스템의 아키텍처 모델

<그림 2>는 식당 예약 시스템에 대하여 컴포넌트 아키텍처 모델을 구성한 예를 보여준다. 시스템을 설계한 아키텍처 설계자의 입장에서는 아키텍처를 구성하는 개별 컴포넌트가 설계자의 설계 의도를 만족하는지 개발과정에 걸쳐서 통제할 필요가 있다. 이를 위하여 xUnit 프레임워크와 같은 테스트 프레임워크를 설계 모델과 함께 구현 개발자에게 제공함으로서 적절한 구현 통제를 실시할 수 있다. 테스트 드라이버 코드는 일반적인 유닛 테스트 프레임워크의 규칙에 맞게 생성되도록 하여 유닛 테스트 프레임워크를 수행시킴으로써 컴포넌트 개발 시에 아키텍처 설계 상의 요구사항이 만족하는지를 즉시 확인할 수 있도록 하여 컴포넌트 개발 시에 발생하는 오류를 조기에 줄여주는 효과가 있다. 하지만 이와 같은 테스트 드라이버 코드는 수동으로 개발할 경우 그 어려움으로 인해 특별히 숙련된 개발자의 경우 외에는 개발이 이루어지지 않는다.

<그림 3> 테스트 코드 생성의 흐름도

<그림 3>은 본 연구의 시스템 동작에 대한 흐름도를 나타내고 총 다섯 단계로 구성된다. 첫 과정은 아키텍트가 컴포넌트로 이루어진 아키텍처 모델을 입력하는 과정이다. 본 연구에서 대상으로 하는 아키텍처 모델의 예는 <그림 2>와 같다. 아키텍처 모델은 구성요소로서 컴포넌트에 대한 정의와 컴포넌트간의 관계가 표현되어야 한다. 컴포넌트는 컴포넌트의 상태를 표현하는 상태 변수가 정의되어 있어야 하며 상태변수들에 대한 정보는 컴포넌트의 속성 정보 형태로 입력이 되어 있어야 한다.

2 단계에서는 사용자로부터 개별 컴포넌트에 대해서 컴포넌트의 개별 인터페이스 별로 스냅샷 정보를 입력한다. 스냅샷 정보는 여러 개의 row로 구성되면 하나의 row는 상태변수 리스트, 상태값 리스트, 입력값 리스트, 기대값 리스트로 구성된다. 상태변수 리스트는 해당 스냅샷에 관련있는 상태변수들의 리스트이다.

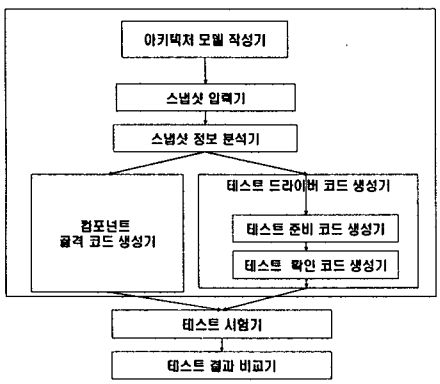
3 단계에서는 아키텍처 모델에서 입력한 정보를 이용

하여 컴포넌트들의 골격코드를 생성한다. 컴포넌트의 골격코드에는 컴포넌트의 정의 부분, 상태변수의 정의 부분, 일반 인터페이스의 정의 부분, 상태변수 인터페이스 정의 부분으로 구성된다. 상태변수 인터페이스 부분은 테스트 과정에서 위해서 상태변수를 설정하고 상태 변수의 값을 확인하기 위하여 사용된다. 이러한 상태변수 인터페이스 부분은 소프트웨어 시스템이 최종 완성되는 시점에서는 보안을 위하여 제거될 수 있다.

4 단계에서는 스냅샷 정보를 분석하여 테스트를 하기 위한 테스트 준비코드를 생성한다. 인터페이스 별로 테스트하기 위하여 각 스냅샷 별로 상태변수와 상태값을 설정하는 메소드를 호출하는 코드를 작성한다.

5 단계에서는 스냅샷에서 기대값에 해당하는 결과가 되었는지 확인하는 테스트 확인 코드를 생성한다. 확인 코드는 xUnit 테스트 프레임워크에서 제공하는 확인 메커니즘을 활용한다.

마지막으로 테스트 준비코드와 테스트 확인코드를 결합하여 시험하려고 하는 인터페이스를 테스트하기 위한 테스트 드라이버 코드를 생성한다.



<그림 4> 테스트 코드 생성의 흐름도

<그림 4>는 본 연구에서 개발되는 시스템의 구성도이다. 본 시스템은 아키텍처 모델 작성기를 통해서 아키텍처 모델을 입력한다. 아키텍처 모델은 그래픽 사용자 인터페이스를 사용하여 소프트웨어 시스템의 아키텍처 도면을 입력하도록 구성한다. 아키텍처는 <그림 2>와 같은 형태의 컴포넌트의 정의와 컴포넌트의 인터페이스 정보를 입력한다. 스냅샷 입력기는 각각의 컴포넌트의 인터페이스에 대하여 상태변수, 상태값, 입력값, 기대값을 입력할 수 있는 사용자 인터페이스를 제공한다. 이와 같이 작성된 아키텍처 모델을 이용하여 컴포넌트 골격코드 생성기에서는 컴포넌트에 대한 정의를 이용하여 컴포넌트의 초기 골격코드를 생성한다. 이 초기 골격코드에는 테스트를 위해서 상태 값들을 설정하기 위한 설정 메소드들이 추가된다. 테스트 드라이버 코드 생성기에서는 스냅샷 정보 분석기를 수행하여 개별 인터페이스에 대한 스냅샷 정보를 분석하고 스냅샷 정보에 따른 테스트 준비 코드를 생성한다. 테스트 준비 코드 생성기는 컴포넌트 골격코드 생성기를 이용하여 생성된 상태 설정 인터페이스를 이용하여 설정하는 부분을 자동으로 생성한다. 테스트 확인 코드 생성기는 메소드 수행후에 각 상태 변수 값들이 제대로 인터페이스 호출 후에 제대로 변경되었는지 확인한다.

3. 결 론

본 연구에서 개발 중인 아키텍처 모델 기반의 테스트 드라이버 자동 생성 기술은 아키텍처 설계자의 의도대로 소프트웨어 시스템이 개발될 수 있도록 유닛 테스트를 통해 확인할 수 있는 효과를 가진다. 본 연구는 일반적인 유닛 테스트를 수행하기 위하여 설계 단계에서 개별 컴포넌트를 시험할 수 있는 테스트 케이스를 생성해내며 이를 컴포넌트 개발자에게 테스트를 수행하면서 개발할 수 있도록 지원함으로써 소프트웨어 개발 중에 용이하게 사전 테스트를 수행할 수 있도록 지원한다. 이러한 기법은 현재 주목받고 있는 테스트 주도 개발 방법론(Test Driven Development)등에 응용되어 사용될 수 있다.

참고문헌

[1] Craig Murphy,, " Improving Application Quality Using Test-Driven Development (TDD)," Methods & Tools, Spring 2005, pp 2,-17

[2] Kent Beck, Erich Gamma, JUnit Cookbook, http://junit.sourceforge.net/doc/cookbook/cookbook.htm

[3] P. Selonen and J. Xu, " Validating UML Models Against Architectural Profiles" , Proc. ESE/FSE' 03, September 1-5, 2003, pp. 58-6