테스트 주도 개발을 위한 유연한 유닛 테스트 도구 설계 기법

전석환 [O1] 김정동[2] 백두권[2*]
[1]고려대학교 컴퓨터공학과
[2]고려대학교 컴퓨터학과[1]
{click92[O], kjd4u, baikdk}@korea.ac.kr

A Design Technique of a flexible unit testing tool for test driven development

Seokhwan Jeon[O1] Jeongdong Kim[2] Doo-Kwon Baik[2*]
[1]Dept, of Software Engineering, Korea University
[2]Dept, of Computer Science & Radio Communications Engineering, Korea University

1. 서론

테스트 주도 개발에서의 테스트는 소프트웨어에서 중복되는 로직을 제거하고 프로그램의 이해도를 향상시키며 새로운 코드의 추가로 인한 오류를 직관적으로 알 수 있도록 하는 수단이 된다[1]. 테스트 주도 개발은 소프트웨어의 효율적인 디자인을 가능하게 하고 테스트에 들어가는 시간적 오버헤드를 상쇄하고 최종적으로 소프트웨어 개발의 생산성을 향상시킨다[2, 3]. 테스트 주도 개발에서의 테스트는 주로 유닛 테스트에 기반하는데 기존의 대표적인 유닛 테스트 프레임워크는 XUnit기반 도구들이다[4, 5, 6]. 기존의 유닛 테스트 도구는 테스트 케이스 변경이 쉽지 않고 코드 추가 작업이나 컴파일 과정에서 시간적 낭비를 초래할 수 있다[4]. 테스트 주도 개발에서는 반복되는 테스트를 효율적으로 수행하기 위해 테스트 케이스를 쉽고 빠르게 수정할 수 있어야 한다. 본 논문에서는 테스트 도구 설계 시 표준 자바 스크립트를 지원하는 스크립트 엔진과 XML 파서를 포함시키고 Layers 패턴을 이용하여 테스트 도구가 유연한 구조로 설계될 수 있는 방법을 제안한다. 또한 제안한 기법에 따라 테스트 도구를 구현하여 그 가용성을 검증하고 개발자가 자바 스크립트로 테스트 케이스를 쉽게 생성하고 변경할 수 있음을 보인다.

2. 본론

테스트 주도 개발의 장점은 프로그램 개발 과정이 실용적이며 높은 품질의 프로그램을 개발 할 수 있다는 점이다. 이러한 장점을 유지하기 위해서는 요구 사항의 분석 주기를 짧게 하여 소프트웨어를 개발하는 동안 요구 사항 변경을 즉각적으로 적용한다. 요구사항 변경을 만족하려면 변경되는 프로그램의 모든 코드에 대해서 반복적이고 다양한 테스트가 가능해야만 한다.

테스트 주도 개발은 유닛 테스트에 기반하는데 기존의 유닛 테스트 기법과의 가장 큰 차이점은 테스트 케이스의 개발을 실제 코드의 개발 이전에 한다는 것이다[7]. 따라서 100%의 테스트 커버리지를 목표로 하게 되고 코드의 품질을 향상시킬 수 있다. 테스트 주도 개발에서 사용하는 기존의 유닛 테스트 도구들은 XUnit 프레임워크를 기반으로 만들어진 것이 많은데 테스트 대상이 되는 클래스를 테스트 프레임워크에 정의된 클래스로부터 상속받게 하는 방법을 주로 사용한다[5, 6]. XUnit 프레임 워크 기반의 테스트 도구는 테스트 케이스가 프로그램의 원시코드에 같이 기술된다. 따라서 테스트 케이스를 변경하여 테스트할 필요가 있을 때 테스트 코드를 포함하고 있는 원시코드를 수정 후 재 컴파일 과정을 거쳐야만 한다. 이 부분은 테스트가 반복되는 테스트 주도 개발에서 효율성이 떨어진다. 유닛 테스트 도구는 유연성을 갖추어 자주 변경되는 테스트 케이스를 효율적으로 적용할 수 있도록 설계되어야만 한다.

본 논문에서 제안하는 테스트 도구 설계 방법은 자바 스크립트 엔진과 XML 파서를 내장하여 테스트 대상 모듈과 자바 스크립트의 오브젝트를 맵핑 시키는 방법을 사용한다. 또한, 기능에 따라 레이어(Layers) 패턴을 적용하여 유연성을 가지고 필요 시 대체 모듈이나 확장 모듈로 쉽게 변경할 수 있는 구조이다. 테스트 도구는 총 4개의 레이어로 구성되는데 레이어 1은 테스트 케이스와 모듈에 대한 메타 정보를 입력 데이터로 사용하고 플러그인(Plug-In) 방식을 사용하여 테스트 대상 모듈을 로드하는 모듈

*:교신저자

인터페이스 레이어이다. 레이어 2는 자바 스크립트 런타임에 생성되는 오브젝트와 맵핑 되는 클래스를 관리하는 레이어이다[8]. native 코드를 사용한 테스트 대상 모듈은 자바 스크립트에서 사용하는 데이터와 그 타입이 다르기 때문에 레이어 2에서는 함수 호출 시 파라미터의 형 변환을 위한 데이터 타입 변환과정이 필요하다. Layer 3은 자바 스크립트 엔진을 내장하고 스크립트를 인터프리트(Interpret)하는 레이어이다. Layer 4는 테스트 수행 정보를 사용자가 볼 수 있도록 UI를 가지는 레이어이다.

본 논문에서는 제안한 테스트 도구 설계 기법의 가용성 검증을 위하여 테스트 도구를 구현하고 웹 서비스 로그인 처리 모듈을 대상으로 테스트를 수행하였다. 테스트를 위해 먼저 테스트 대상이 되는 모듈은 미리 정의된 헤더를 추가하여 플러그인 형태로 변경된다. 다음으로 XML 문서에는 테스트 대상이 되는 로그인 처리 모듈의 위치와 테스트 함수 및 파라미터 정보가 기술되고 테스트 케이스는 자바 스크립트로 기술된다. 구현된 테스트 도구에서는 XML 문서에 기술 된 테스트 대상 모듈의 정보에 따라 테스트 대상모듈을 로드 한다. 테스트 도구의 테스트 실행명령에 따라 자바 스크립트로 기술된 테스트 케이스가 인터프리트 되고 자바 스크립트 엔진 내에 동적으로 생성 된 객체 및 메소드와 맵핑 된 테스트 대상 모듈의 native 함수가 호출된다. 함수 호출 시 내부적으로 자바 스크립트 타입의 파라미터들은 native형의 파라미터로 타입이 변환되고 각각의 파라미터는 직렬화 되어 테스트 대상 모듈에 전달된다. 제안한 설계 기법에 따라 구현된 테스트 도구를 이용한 테스트에서 테스트 대상 모듈에 대해 성공하는 경우와 실패하는 경우를 각각 볼 수 있었다.

기존 XUnit기반 유닛 테스트 도구와의 비교 평가를 위해 구현 된 테스트 도구의 테스트 대상모듈과 XUnit 기반의 테스트 대상 모듈에 대해 컴파일 시간을 측정하였다. 각각의 모듈에 대해 클래스를 추가하고 테스트 케이스를 추가하면서 배치파일을 이용하여 컴파일 시간을 측정한 결과 본 논문에서 구현한 테스트 도구는 기존 도구에 비해 20% 이상 컴파일 시간이 줄어듦을 보였다. XUnit 기반 유닛 테스트 도구와 달리 본 논문에서 구현한 테스트 도구는 테스트 케이스가 프로그램 원시코드와 분리되므로 원시코드의 컴파일 시간이 줄어 둘 수 밖에 없다. 또한, 코드의 수정 량이 같다고 가정할 때 프로그램 원시코드의 수정보다는 스크립트의 수정이 더 용이하므로 컴파일 시간이 동일하다 하더라도 기존 도구에 비해 제안 설계 기법은 생산성에 있어 더 효율적이며, 프로그램 원시 코드와 별도의 문서로 관리되는 테스트 케이스는 재 활용성이 높다.

3. 결론

본 논문에서는 테스트 주도 개발에서 테스트 케이스 변경 시 테스트 대상의 메타 정보와 자바 스크립트를 이용하여 빠르게 변경된 테스트를 수행할 수 있는 유닛 테스트 도구의 설계 기법을 제안하였다.

제안한 설계 기법에 따라 테스트 도구를 구현함으로써 제안 설계 기법의 가용성을 검증하였고 기존의 테스트 도구와의 비교 평가를 통해 본 논문에서 구현한 테스트 도구는 재 컴파일이 필요 없거나 컴파일 시간이 줄어듦을 보였다.

향후 연구로는 다양한 Mock 오브젝트나 유저 인터페이스를 지원하고 테스트 대상에 대한 표준 인터페이스를 사용하는 개선된 테스트 도구를 설계하여 다양한 테스트 환경에 적용할 것이다.

참고문헌

[1] Boby George and Laurie Williams, " A structured experiment of test-driven development," Information and Software Technology, Vol. 46, Issue. 5, pp. 337-342, 2004.

[2] David Janzen and Hossein Saiedian, " Test-Driven Development:Concepts, Taxonomy, and Future Direction," The flagship publication of the IEEE Computer Society, Vol. 38, no. 9, pp. 1-3, 2005.

[3] Hans Wasmus and Hans-Gerhard Gross, " Evaluation of Test-Driven Development," Delft University of Technology Software Engineering Research Group Technical Report Series, 2007.

[4] Charles D. Allison, " The simplest unit test tool that could possibly work," Vol. 23, Issue. 1, pp.183-189, 2007.

[5] Panagiotis Louridas, " JUnit: Unit Testing and Coding in Tandem," IEEE SOFTWARE, Vol. 22, no. 4, pp. 12-15, 2005.

[6] 김영상, 백창현, 박승규, " 유닛 테스트 자동화 도구를 위한 프레임워크 설계," 한국정보과학회 한국컴퓨터종합학술대회 논문집 pp. 184-186, 2006.

[7] 김희진, 최병주, 윤석진, "테스트 주도 개발(TDD)에서의 모바일 응용 소프트웨어 성능 테스트 방안," 한국정보과학회 학술발표논문집 제34권 제2호(B), pp.143-146, 2007.

[8] Oystein Hallaraker and Giovanni Vigna, " Detecting malicious JavaScript code in Mozilla," Engineering of Complex Computer Systems, pp. 85-94, 2005.