J2ME 기반 모바일 응용 SW의 단위 테스트 도구 개발[i]

윤석진*, 김철홍, 신규상,
한국전자통신연구원
e-mail:sjyoon, kch, gsshin@etri..re.kr

A Study on Unit Test Tool for Mobile Application Software based on J2ME

Seok-Jin Yoon**, Chul-Hong Kim, Gyu-Sang Shin
Embedded SW Research Division, ETRI

요 약

최근 PDA와 휴대폰 등 모바일 디바이스의 급격한 증가로 모바일 응용 SW 개발이 주목을 받고 있다. 다양한 방법과 기술들이 모바일 응용 SW 개발을 지원하기 위해 개발되었으며, 다양한 언어가 J2ME 또는 Brew 플랫폼에서 SW 개발을 위해 제시되었다. 그러나 이러한 환경에서 SW 테스트는 큰 관심을 받지는 못하였다. 그 동안 모바일 응용 SW의 기능 테스트에 대한 몇몇 연구 성과들이 발표되어 왔고, 성능, 사용성, 스트레스 테스트 등에 관련된 연구들이 주로 진행되어 왔다. 특히 단위 테스트는 개발자의 역량에 의존하는 방식으로 진행되어 왔다. 본 논문에서는 모바일 응용 SW를 비즈니스 로직 측면과 GUI 측면으로 구분하여 각각에 대한 단위 테스트 방법을 제시하고자 한다.

1. 서론

최근 국산 휴대폰은 치열한 경쟁 상황에서 시장적시성을 만족하기 위하여 3-4개월의 짧은 기간에 개발하여 출시함에 따라, 탑재된 SW의 품질과 신뢰성이 떨어지고 있으며, 모바일 컨버전스 추세와 함께 응용 서비스 요구가 다양해져서 모바일 SW의 크기도 매년 20-30% 정도씩 증가함에 따라 SW 개발이 점점 더 어려운 환경에 처하고 있다.

특히, 모바일 SW는 운영환경이 매우 다양하고, 많은 제약 조건들을 가지고 있어, 모바일 SW 테스트는 일반 데스크탑 SW의 테스트보다 어렵다. 모바일 SW는 호스트에서 개발되고, 다양한 특성들을 가지고 있는 모바일 디바이스에 탑재된다. 따라서 호스트상의 에뮬레이터에서 테스트되고, 최종적으로 타켓 디바이스 상에서 테스트되어야 한다.

모바일 응용 SW는 GUI 중심으로 비즈니스 로직은 비교적 단순하다. 비즈니스 로직 테스트를 위한 도구는 기존 JUnit를 확장한 공개 SW 버전을 확장하여 개발하고, GUI 테스트 도구는 아직은 미성숙한 분야로 GUI 화면 그래픽을 비트맵을 비교하는 방식의 도구로 구현하고자 한다.

2. 관련 연구

2.1 모바일과 데스크탑 환경의 테스트 차이점

모바일 SW는 데스크탑 SW보다 단순하고, 크기가 작다. 모바일 환경은 단순한 JVM과 HW 계층인 디스플레이 장치, 네트워크 포트 등에 접근하기 위한 API를 제공한다[1]. 이러한 환경에서 프로그래밍과 테스팅은 서버 환경보다 더욱더 어렵다. 휴대폰은 각자 다른 HW 환경, 화면크기와 칼라 수, 메모리 크기와 전력 등을 가지고 있다. 또한 같은 어플리케이션이라도 설치되는 타겟 디바이스에 따라 조금씩 다르게 동작 할 수도 있다. 모바일과 데스크탑 SW 테스트의 차이점은 <표 1>과 같이 요약된다.

<표 1> 모바일/데스크탑 SW 테스트 비교

요소	모바일	데스크탑
테스트 기록	GUI 이벤트 기록을 위한 기능을 제공하지 못해, 사용자 인터랙션에 대한 에뮬레이션이 불가능	GUI 이벤트 기록을 위한 java.awt.Robot 제공
배치 자동화	디바이스 상에 배치와 테스트의 지루한 수작업	배치가 완전히 자동화되어 있고, 테스트도 비교적 자동화되어 용이함
테스트 환경 차이점	디바이스 간에 차이점이 있고, 모든 디바이스 상에서 테스트가 요구됨	개발과 배치/테스트가 동일하고, 특정 OS에 의존하는 경우가 거의 없음
API	API 표준화가 아직 미성숙	API 표준이 성숙되어 있고, 다양한 벤더로부터 JVM 이식가능

i 본 연구는 정보통신부 및 정보통신연구진흥원의 IT신성장동력핵심기술개발사업의 일환으로 수행하였음. [2007-S032-01, 다중 플랫폼 지원 모바일 응용 S/W 개발환경 기술 개발]

2.2 로직 단위 테스트

단위 테스트의 목적은 모듈이 제대로 구현되었는지 시험하는 것으로 모듈이 어떤 작업을 수행하는지를 나타낸 설계 명세를 근거로 외부 관점의 테스트(이를 블랙 박스 테스트라고함)를 할 수도 있고, 프로그램의 내부를 살펴서 논리 흐름을 체크하는 화이트 박스 형태를 취할 수도 있다[2].

단위 테스트는 대부분 비공식적으로 다루어 프로그램 개발자에게 일임하는 경우가 많다. 프로젝트의 공식적인 절차로 다룰 경우는 테스트 계획에 무엇을 테스트하며, 어떤 결과를 예상하고 있는지 작성하여 설계자와 프로그래머가 사인한 후 테스트 케이스를 수행시킨다. 단위 테스트의 완료 시점은 코드 커버리지를 사용하는 경우가 많다. 원시코드 문장이 한번 이상, 또는 분기에 대한 조건이 모두 수행되었는지 확인하여 단위 테스트가 완료되었는지 확인한다[2].

모바일 환경에서의 SW 단위 테스트 도구로는 J2MEUnit[3], JMUnit[4], Sony Ericsson Mobile JUnit[5] 등이 있다. 3개의 도구는 JUnit을 확장한 것으로 사용법이 비슷하다. JUnit Helper 메소드가 서브 클래스로 제공되는데, 예를 들면, assertTrue, assertFalse, assertEquals 등이다. JUint에 익숙한 개발자들은 핵심 API가 같기 때문에 어려움 없이 새로운 단위 테스트를 작성할 수 있다.

차이점은 모바일 플랫폼인 J2ME에서는 자바 데스크탑 플랫폼인 J2SE에서 제공하는 리플렉션(reflection) 기능이 없기 때문에, JUnit과 같이 리플렉션을 활용한 테스트 도구를 개발하기 어렵다. 단위 테스트 도구에서 리플렉션 기능은 테스트 개발자에게 테스트 클래스 및 테스트 메소드의 추가를 용이하게 한다. 리플렉션 기능이 없다면 테스트 결과를 수집하고, 테스트를 수행하는데 있어서 테스트를 추가할 때마다 개발자가 수작업으로 관련된 테스트를 수정해야한다.

기존의 모바일 SW 단위 테스트 도구 중에서 기능상으로는 Sony Ericsson Mobile JUnit이 가장 뛰어나지만, 자체 플랫폼과 에뮬레이터만 지원하고 있어 국내의 개발 환경에 적용하기 어렵다.

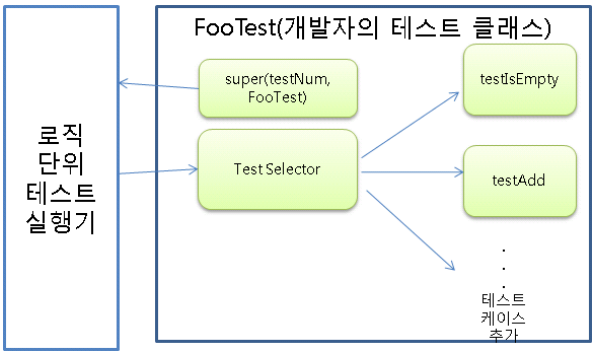
3. 모바일 응용 SW 단위 테스트 도구의 구성

3.1 비즈니스 로직 단위 테스트 도구

모바일 SW 개발 환경에서는 리플렉션 기능이 없기 때문에 일반적인 JUnit과 같이 리플렉션을 활용한 테스트 도구를 개발하기 어렵다. 또한 에뮬레이터와 연동되어 동작할 수 있는 기능이 필요하다.

기존의 모바일 환경을 지원하는 단위 테스트 도구 중에 기능상으로는 소니 에릭슨의 Mobile JUnit이 가장 뛰어나지만 자사 플랫폼만을 지원하고 소스가 공개되어 있지 않아 국내 모바일 SW 개발 환경에 적용하여 이용할 수가 없다.

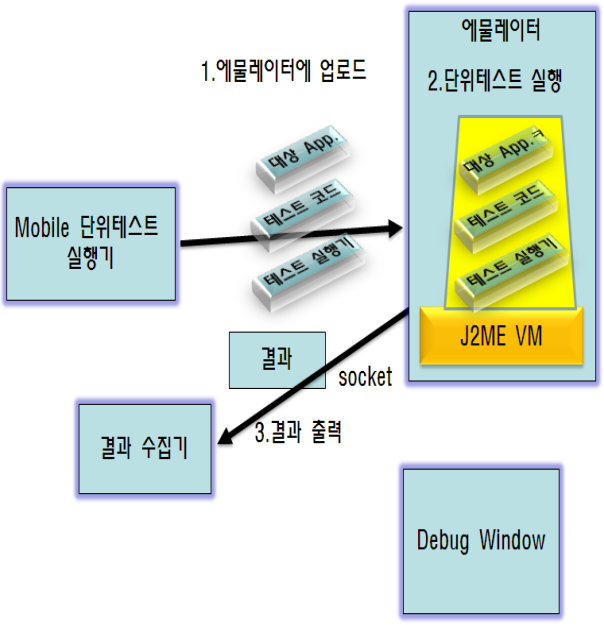
본 로직 단위 테스트 도구는 소스 코드가 공개된 JMUnit을 기반으로 구현하고자 한다. JMUnit은 지원되지 않는 리플랙션 기능을 해결하기 위해 생성자에 super 메소드(테스트 개수, “테스트 클래스명”)와 같은 방법을 사용하는데, 본 도구도 이러한 방법을 사용하여 리플렉션 기능을 구현한다.



<그림 1> 로직 단위 테스트의 구성

<그림 1>은 JMUnit을 이용한 단위 테스트 코드의 구성을 설명한다. 우선 테스트 클래스는 생성자를 통해서 테스트 클래스에 포함되어 있는 테스트 케이스의 개수와 클래스명 정보를 알려준다. 이리하여 로직 단위 테스트 실행기는 테스트 선택기 메소드를 자동으로 호출하여 테스트를 수행하게 된다. 해당 결과는 테스트 실행기가 수집하여 테스트 결과를 보고하게 된다.

이와 같은 단위 테스트를 현재 국내에서 많이 사용하고 있는 에뮬레이터에서 동작시키기 위해서는 <그림 2>와 같은 구조의 단위 테스트 실행기와 결과 수집기가 구성되어야한다.



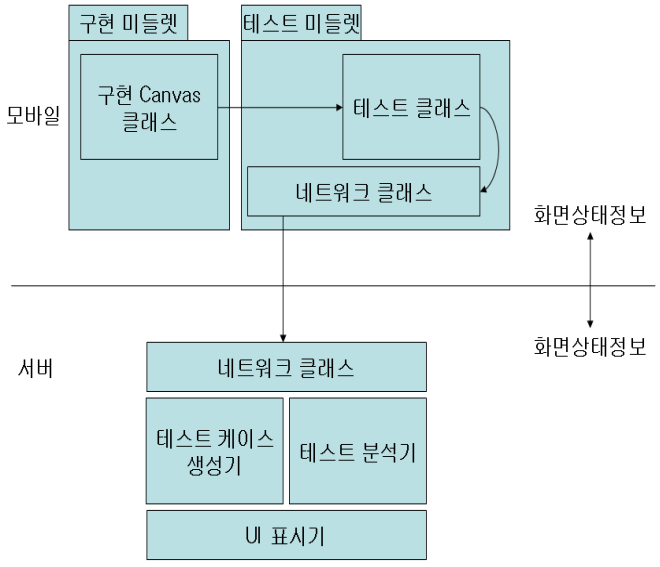
<그림 2> 일반 에뮬레이터와의 통합

3.2 GUI 단위 테스트 도구

GUI 단위 테스트는 테스트 수행에 앞서 서버 측에서 테스트 케이스 생성기를 통하여 테스트할 항목들 즉, 화면 상태(state), 이벤트 정보, 출력 이미지 등으로 구성된 테스트 케이스를 생성한다. 모바일과 서버의 상태 정보를 동기화하기 위하여 개발자는 서버 측에 모바일 상에서 구축한 상태 정보(객체 또는 변수)를 입력하여 준다.

테스트 방법으로 모바일(또는 에뮬레이터) 상에서 처리되는 작업은 화면에 출력된 그래픽, 현재 화면 상태 정보(객체 또는 변수), 그리고 이벤트 정보(키 입력, 자동전환) 등과 같은 정보들을 네트워크를 통하여 서버로 전송한다.

서버측은 네트워크를 통하여 들어온 정보들을 분석하여 그래픽은 이미지 파일로 저장하고, 상태와 이벤트 정보는 XML 형식으로 데이터를 저장한다. 테스트 수행은 모바일 상에서 전송되어온 데이터를 테스트 케이스 생성기에서 생성한 데이터와 비교 후 일치하는 정보가 있으면 이미지를 비교한다. 이를 통하여 화면에 출력된 이미지와 테스트 케이스 생성기에서 생성한 예상 이미지의 상태가 같다면, GUI 테스트가 성공하였다고 판단한다. 하나의 테스트 케이스를 수행하면 서버 측의 UI에 테스트 수행 결과를 표시하여 성공과 실패를 표현한다.



<그림 3> GUI 단위 테스트 개념도

4. 결론

본 논문에서는 모바일 응용 SW를 비즈니스 로직과 GUI 측면으로 나누어 각각에 대한 단위 테스트 방법 개발을 위한 개념을 제시하였다. 비즈니스 로직 단위 테스트는 공개 SW 도구인 JMUnit을 기반으로 지원되지 않는 리플렉션 기능을 추가하여 확장 개발하고, GUI 테스트 도구는 테스트 오라클인 예상 그래픽 이미지를 사전에 저장하여 테스트 실행 후 생성된 그래픽 이미지를 비트맵 단위로 비교하는 기법을 제시하였다. 향후 계획은 이러한 단위 테스트 방법을 이용하여 테스트 케이스를 개발자 입장에서 좀더 빠르고 쉽게 생성할 수 있는 테스트 생성 기술의 개발이 필요하다.

참고문헌

[1] Dawid Weiss and Marein Zduniak, Automated Integration Tests for mobile Application in Java 2 Micro Edition, LNCS, Springer Belin, pp478-487, 2007.

[2] 소프트웨어 테스트 전문기술, TTA, 2003.

[3] http://j2meunit.sourceforge.net

[4] http://sourceforge.net/projects/jmunit

[5] Unit Testing with Sony Ericsson Mobile Unit, Sony Ericsson, Sep 2006.

[6] Atif M. Memon, A Comprehensive Framework for Testing Graphical User Interfaces, Ph.D. Dissertation, Univ. of Pittsburgh, 2001.

[7] Alex Ruiz and Yvonne Wang Price, Test-Driven GUI Development with TestNG-Abbot, IEEE Software, May/June 2007.

[8] Juichi Takahashi and Yoshiaki Kakuda, Effective Automated Testing: A Solution of Graphical Object Verification, IEEE ATS, 2002.