모바일 어플리케이션을 테스트하기 위한 아키텍처 모델

노명기° 류성열
숭실대학교 컴퓨터학과
infinite@selab.ssu.ac.kr°, syrhew@comp.ssu.ac.kr

Test Architecture Model of Mobile Applications

MyoungKi Roh° SungYul Rhew
Dept. of Computer Science, Soong Sil University

요 약

모바일 어플리케이션은 플랫폼이 제공하는 API를 통해서 만들어진다. 모바일 어플리케이션을 제작하고 운용하는데 모바일 플랫폼이 차지하는 비중은 아주 높다. 따라서 모바일 어플리케이션의 플랫폼 종속성은 높다고 할 수 있다. 플랫폼 종속적인 모바일 어플리케이션을 테스트하기 위해서는 해당 플랫폼에 대한 정보를 가지고 있어야 한다. 본 논문에서는 플랫폼 종속적인 모바일 어플리케이션을 해당 플랫폼에 대한 종속성을 감소시켜 다양한 플랫폼 환경에서 테스트 할 수 있는 아키텍처를 제시한다. 또한 이를 일바탕으로 특정 플랫폼에서 수행된 테스트 데이터가 다른 플랫폼에서도 사용될 수 있도록 테스트 데이터의 재사용을 향상시킬 수 있는 테스트 아키텍처를 제시한다.

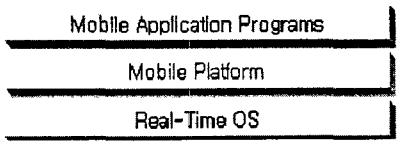
1. 서 론

무선인터넷에 대한 연구와 개발이 활발히 이루어지고 있는 상황이다. 그러나 무선인터넷에 대한 표준 제정에 힘을 기울이고 있으나 아직까지 세계적인 표준이 정해지지 않았다. 따라서 현재 무선인터넷 분야는 다양한 기준이 존재하고 이에 따라 여러 가지 플랫폼이 제공되고 있다. 이러한 다양한 환경에서 개발된 어플리케이션을 테스트 하는 것은 쉽지 않은 일이다. 다양한 환경은 테스트 소요시간을 예측하기 어렵게 만들며 비용 증가의 원인이 된다. 그리고 다양한 플랫폼은 모바일 어플리케이션 테스트 자동화의 걸림돌이 된다. 특정 플랫폼에서 개발된 모바일 어플리케이션은 해당 플랫폼에서 제공하는 API를 이용해서 개발되고 테스트를 할 때에도 해당 플랫폼에서 제공하는 API의 정보를 알고 있어야 한다. 그리고 수행된 테스트 결과 역시 해당 플랫폼에 종속성을 가지고 있다. 해당 플랫폼에 종속적인 테스트 데이터는 다른 플랫폼에서 재사용하기 어려운 일이므로 테스트 데이터의 재사용성은 떨어진다 할 수 있다. 앞에서 언급한 두 가지 문제를 해결하기 위해서는 다양한 환경에서 모바일 어플리케이션 테스트를 할 수 있는 테스트 환경이 우선적으로 구축되어야 하며, 나아가서 해당 플랫폼에 종속적인 테스트 데이터를 다른 플랫폼에서 사용할 수 있도록 만들어주어야 한다. 다양한 환경에서 테스트가 가능하고 그 결과로써 나온 테스트 데이터가 플랫폼에 상관없이 사용될 수 있다면 테스트 데이터의 재사용성은 향상될 수 있다. 본 논문에서는기존의 테스트 아키텍처를 확장함으로써 플랫폼의 종속적이지 않은 테스트와 테스트 데이터를 이용하는 방법을 제시하고자 한다.

2. 관련연구

2.1 모바일 계층구조

모바일의 계층 구조는 각 플랫폼 마다 약간씩 상이한 구조로 되어있지만 기능적인 면으로 계층을 나누면 <그림 1>과 같은 4가지의 계층으로 분류할 수 있다.



<그림 1> 모바일 계층구조

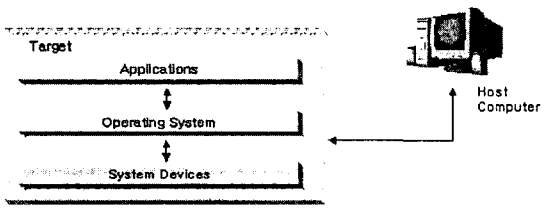
Handset H/W 계층은 칩 셋(chip set)이 탑재되어있는 물리적인 하드웨어 계층을 말한다. Real-Time OS 계층은 지정된 시간 제한 내에 확실한 출력을 보장하는 운영체계 계층이다. 플랫폼에 따라 다양한 RTOS가 탑재되는데 Handset H/W 계층을 제어하고 Mobile Platform 계층에서의 서비스를 제어하는 역할을 담당한다. Mobile Platform 계층은 통신, 멀티미디어, 게임 등 Mobile Application Programs의 개발 환경을 지원하기 위한 API들을 제공하며 Mobile Application Programs를 관리하고 실제로 실행되는 계층을 말한다. Mobile Application Programs 계층은 해당 플랫폼에서 제공하는 API를 이용하여 사용자에게 서비스되는 응용프로그램을 말한다[1][2].

모바일 플랫폼은 다양한 기준을 따르고 있다. 모바일 어플리케이션을 테스트하기 위해서는 우선 모바일 어플리케

이션의 해당 플랫폼에 대한 정보가 필요하다.

2.2 내장형 소프트웨어 테스트 아키텍처

모바일 어플리케이션은 내장형 소프트웨어(Embedded Software)의 하나이다. 내장형 소프트웨어는 일반 소프트웨어와는 달리 실시간성, 고신뢰성, 저전력을 요구하는 특성을 가지며 통상적으로 개발 플랫폼과 타겟 플랫폼이 일치하는 않는다. 이러한 개발 환경을 "교차-개발환경(Cross-Development Environment)"이라고 부른다[3]. <그림 2.1>은 일반적인 내장형 소프트웨어의 개발 환경을 나타낸다.



<그림 2.1> 교차-개발환경

따라서 내장형 소프트웨어를 테스트할 때에도 교차-개발환경을 만족시켜주어야 한다. 내장형 소프트웨어 테스트 아키텍처(Embedded Software Test Architecture)는 <그림 2.2>과 같다[4][5].

■ 사용자 인터페이스(GUI)

사용자와 상호 작용하여 테스트 진행 과정을 관리하고, 테스트 과정에서 사용자가 직접 개입하여 테스트 데이터나 테스트 결과 판정을 위한 예상 결과 값 등을 조정할 수 있다. 또한 테스트 결과를 사용자에게 보여주는 기능을 가지고 있다.

■ 프로그램 분석기(Program Analyzer)

테스트 대상 프로그램 분석을 통해 테스트 정보를 추출한다.

- 테스트 데이터의 자료형 정보
- 테스트 데이터의 상수값 정보
- 프로그램 제어 흐름 정보
- 이벤트 정보
- 프로그램 제어 흐름 모니터링 정보
- 이벤트 모니터링 정보

■ 테스트 환경 설정기(Test Setup)

테스트 환경 설정기에서는 타겟 사이드로 사용될 논리적 노드의 물리적 위치를 지정한다.

■ 테스트 드라이버

테스트 드라이버를 생성한다.

■ 테스트 수행기(Test Engine)

타겟 사이드에서 수행될 테스트를 수행하고 관리하는 역할을 한다. 또한 테스트 수행이 끝난 후에는 테스트 결과를 수집하여 테스트 보고서를 생성하는 자료를 준비한다.

■ 테스트 조정기(Test Coordinator)

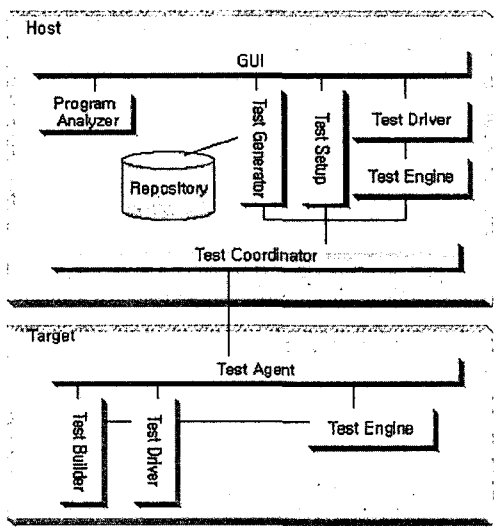
타겟 사이드에서 테스트 수행의 동기화를 책임지고 타겟 사이드의 테스트 수행 중 발생하는 이벤트를 보고 받아 테스트 생성기에서 생성한 이벤트 시퀀스에 근거하여 계속 진행 시킬 지의 여부를 판단하여 타겟 사이드에 지시하는 역할을 수행한다.

■ 테스트 빌더(Test Builder)

타겟 사이드에서 테스트가 가능한 실행 코드를 생성한다.

■ 테스트 중개기(Test Agent)

호스트 사이드와 타겟 사이드 사이에서 정보를 전송하거나 받는 역할을 한다.



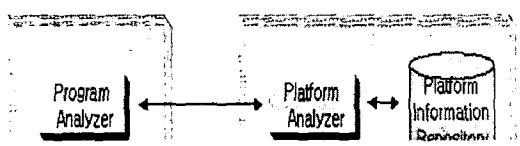
<그림 2.2> 내장형 소프트웨어 테스트 아키텍처

3. 내장형 소프트웨어 테스트 아키텍처의 확장

3.1 모바일 플랫폼 분석 계층 확장

모바일 어플리케이션은 해당 플랫폼에서 제공하는 API에 따라 개발되기 때문에 모바일 어플리케이션을 테스트하기 위해서는 타겟 테스트 소스 속에 플랫폼이 제공하 는 AP

정보를 포함해야 한다. 타겟 테스트 소스 안에 해당플랫폼의 API를 제공해주기 위해서 플랫폼 정보를 저장하는 리파지토리를 구축할 필요가 있다. 다양한 플랫폼 정보를 접근하기 용이하게 하기 위해서 리파지토리는 XML을 기반으로 한다. 플랫폼 정보 리파지토리에서 테스트하고자 하는 어플리케이션에 해당하는 플랫폼 정보를 가져오기 위한 중간 계층(Platform Analyzer)을 두어야 한다. Platform Analyzer는 테스터가 테스트 하고자 하는 모바일 어플리케이션의 플랫폼을 선택하면, 플랫폼 리파지토리에서 해당 플랫폼의 정보를 검색하고 해당 정보를 Program Analyzer로 전달하는 역할을 한다. Program Analyzer는 전달받은 정보를 타겟 테스트 소스 안에 삽입할 수 있도록 해준다 <그림 3.1>. Platform Analyzer 계층 계층을 추가함으로써 테스터는 하나의 툴 안에서 타겟 보드의 교체만으로 플랫폼을 고려하지 않고 다양한 모바일 어플리케이션을 테스트 할 수 있게 된다.

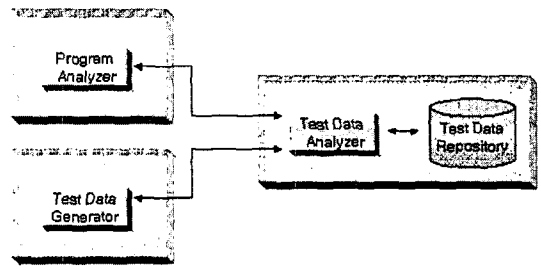


<그림 3.1> 모바일 플랫폼 분석 계층

3.2 테스트 데이터 생성 계층 확장

테스트에 사용될 테스트 데이터를 자동 생성할 수 있으면 테스트 데이터를 수작업으로 생성하는 것보다 시간이 단축되기 때문에 테스트의 효율을 높일 수 있다. 테스트 데이터를 자동 생성하기 위해 과거에 테스트가 수행된 테스트 결과나 테스트를 하려는 데이터가 리파지토리에 저장되어 있어야 한다. 리파지토리에 저장된 정보를 이용하여 타겟 테스트 소스에 맞는 테스트 데이터를 생성할 수 있다.

모바일 플랫폼들은 동일한 기능 또는 유사한 기능을 다른 형식의 API로 제공한다. 모바일 어플리케이션은 플랫폼에서 제공하는 API를 종속적으로 이용한다. 따라서 모바일 어플리케이션은 플랫폼에 종속적이기 때문에 수행된 테스트의 결과도 플랫폼에 종속적인 정보를 담고 있다. 그러므로 플랫폼을 달리할 때 특정한 플랫폼에서 수행된 테스트의 결과를 재사용하기는 어렵다. 본 논문에서 이러한 상황을 해결할 수 있는 방법을 제시한다. 특정한 플랫폼에서 수행된 테스트 결과를 리파지토리에 저장할 때 테스트 결과를 분석하여 재사용이 가능한 형태로 변환시키는 계층을 둔다. 이 계층(Test Data Analyzer)은 수행된 테스트 결과를 리파지토리에 변환하여 저장하는 기능과 테스트 데이터를 생성할 때에 리파지토리에서 정보를 가져와 특정한 플랫폼에 맞게 변환하는 기능을 수행한다.



<그림 3.2> 테스트 데이터 분석 계층

4. 결 론

모바일 플랫폼에 종속적인 모바일 어플리케이션을 테스트할 때 프로그램 분석기(Program Analyzer)가 테스트 타겟 소스를 분석하는 시점에서 플랫폼 분석기(Platform Analyzer) 계층을 두어 해당 플랫폼에 정보를 가져오게 함으로써 플랫폼 종속적인 면을 감소시킬 수 있다. 플랫폼에 개방적인 플랫폼 분석기 계층을 추가함으로써 테스터는 모바일 어플리케이션에 대해 유연하게 대처할 수 있다.

데이터 분석기(Data Analyzer) 계층에서 플랫폼 종속적인 테스트 결과를 다른 플랫폼에서도 사용이 가능하게 하도록 변환시켜 저장함으로써 테스트 결과의 재사용성이 향상되고 테스트 시간을 단축시켜 전반적인 테스트의 효율을 증대시키는 효과를 가져 올 수 있다.

참고문헌

[1] "모바일 플랫폼 표준 WIPI", 한국정보통신기술협회, 2002.12

[2]" BREW White Paper, BREW and J2ME-A Complete Wireless Solution for Operators Committed to Java", QUALCOMM.

[3] Bart Broekman and Edwin Notenboom, "Testing Embedded Software", Addison Wesley, 2003

[4] "분산 및 내장형 소프트웨어 테스팅 기술", 정보통신부, 2002.10.30

[5] Alan Hartman and Kenneth Nagin, "Model Driven Testing-AGEDIS Architecture Interfaces and Tools", European Conference on Model Driven Software Engineering, 2003.12.11