http://dx.doi.org/10.3745/KIPSTD.2012.19D.2.203

XML을 이용한 안드로이드 GUI 위젯의 기능 테스트 자동화

마 영 철[†] · 최 은 만[††]

요 약

캡처 리플레이 기법은 GUI 테스팅을 위하여 흔히 사용되는 자동화 방법이다. 하지만 안드로이드 플랫폼 애플리케이션 테스팅에는 직접 이용하기 어렵다. 그 이유는 좋은 테스팅 프레임워크가 제공되고 있지만 GUI 요소들과 위젯 이벤트 처리 부분이 자동 링크되지 않기 때문이다. 캡처 리플레이 테스팅 도구가 없다면 명세서를 이용하여 테스팅 시나리오를 만들고 일일이 수동으로 GUI 요소들과 연결하여 테스팅 하여야 한다. 이 논문은 현재 사용되고 있는 안드로이드기반 GUI 자동화 테스트 기법보다 발전되고 최적화된 캡처 리플레이 방법을 제안하고 이의 효율성을 증명하기 위한 것이다. XML을 이용하여 주요 위젯기반 원소를 추출하기 위한 기술들을 정립하고 위젯기반 API 이벤트 처리 방법을 고안하였다. 제안한 방법으로 캡처 단계에 모니터링하여 클릭 이벤트가 일어나는 형태를 추적한 후 리플레이 단계에서 활성화 된 위젯을 API 이벤트와 상태를 교환하여 테스트 케이스를 생성한다.

키워드 : 테스트 자동화, GUI 테스팅, XML

Functional Test Automation for Android GUI Widgets Using XML

Ma Yingzhe[†] · Choi Eun Man[††]

ABSTRACT

Capture-and-replay technique is a common automatic method for GUI testing. Testing applications on Android platform cannot use directly capture-and-replay technique due to the testing framework which is already set up and technical supported by Google and lack of automatic linking GUI elements to actions handling widget events. Without capture-and-replay testing tools testers must design and implement testing scenarios according to the specification, and make linking every GUI elements to event handling parts all by hand. This paper proposes a more improved and optimized approach than common capture-and-replay technique for automatic testing Android GUI widgets. XML is applied to extract GUI elements from applications based on tracing the actions to handle widget events. After tracing click events using monitoring in capture phase test cases will be created by communicating status of activated widget in replay phase with API events.

Keywords : Test Automation, GUI testing, XML

1. 서 론

스마트폰의 빠른 보급과 함께 모발 소프트웨어의 개발이 증가하고 있다. 모발 소프트웨어 개발은 기능 위주였던 전통 데스크톱 소프트웨어 개발과는 달리 좋은 사용자 경험을 기초로 GUI에 집중된다. 특히 모발 디바이스에 대한 사용자의 상호작용 경험을 더 좋게 하기 위하여 고해상도 디스플레이와 같은 높은 시각적 효과를 요구한다. 즉 사용자들의 기대치가 높아지기 때문에 개발자들에게 더 복잡한 기능을 구축하여야 한다는 부담을 가중시켜왔다.

사용자 입장에서는 좋은 GUI로 인해 소프트웨어를 더 쉽게 사용할 수 있게 되었지만 개발자에게는 GUI 소프트웨어 개발 과정에 여러 가지를 고려하게 만들었다. 예를 들면 사용 용이성(usability)만이 아니라 입력 방식, 즉 터치 입력 때문에 오는 오류에 대한 안정성(stability) 및 강인성(robustness) 등의 품질을 높이기 위해 GUI 테스팅을 수행하여야 한다. 모발 소프트웨어를 개발할 때 이와 같은 품질을 추구하지만 전통적인 소프트웨어 테스팅 기법만으로 만족할 수가 없다. 왜냐하면 기존 테스팅 기법은 코드의 구조와 실행경로 및 빈도가 사용되지만 GUI 테스팅에서는 이러한 방법이 잘 적용되지 않는다[1]. 또한 모발 소프트웨어의 GUI 개발은 빠르게 변하는 프로토타이핑 모델을 사용하기 때문에 소프트웨어가 계속 업그레이드되고 GUI에도 지속적인 변경을 초래하게 된다[2]. 따라서 회귀 테스트가 매우 중요한 작업이 된다.

† 준 회 원 : 비원플러스 연구원
†† 정 회 원 : 동국대학교 컴퓨터공학과 교수(교신저자)
논문접수 : 2011년 11월 1일
수 정 일 : 1차 2011년 12월 29일, 2차 2012년 1월 30일
심사완료 : 2012년 1월 30일

회귀 테스트 작업을 할 때 테스트 케이스의 생성, 보관 및 평가하는 작업을 수동으로 한다면 많은 시간이 소비되고, 또한 테스트 범위 유지에도 많은 시간이 소비된다[1]. 이러한 이유로 GUI 회귀 테스트에는 자동화된 도구가 절대적으로 필요하다. 특히 스마트폰 중의 대표적인 플랫폼인 안드로이드 폰을 위한 소프트웨어 개발과 테스팅 작업에서는 이러한 도구가 필수적으로 요구된다. 다행히 Google에서는 개발 SDK의 발표와 동시에 새로운 안드로이드 테스팅을 위한 프레임워크(Android Test Framework)를 제공하였다.

안드로이드 기반의 테스팅 프레임워크와 테스트 슈트는 JUnit에 기반을 두고 있다[17]. 그러나 안드로이드 컴포넌트는 일반적인 JUnit을 이용한 테스팅 방법과 같이 테스트 케이스를 만들어서 테스트할 필요가 없다. 왜냐하면 안드로이드 JUnit의 확장된 부분이 이미 특정한 컴포넌트 테스트 케이스 클래스를 제공하기 때문이다. 테스트 케이스 클래스들은 보조 메서드를 제공하고 목(Mock) 객체와 메서드를 만들어준다.

드물지만 자동 프레임워크(ATF: Automatic Testing Framework)를 적용한 잘 알려진 안드로이드 테스트 프레임워크 두 가지 있다. 명세기반 시나리오 기능 테스트 프레임워크인 Robotium과 명세기반 통합 테스트 프레임워크인 Positron이다. 두 가지 프레임워크의 테스트 실행 방법에는 공통점이 있다. 테스트 대상 애플리케이션(Application Under Test)의 소스 코드가 필요하다는 점과 기존 블랙박스 기능 테스트 기법을 사용한다는 점이다[7]. 이런 특징은 테스트 케이스 생성 및 유지 등 중요한 작업을 기존 기법처럼 수동으로 만들게 하며 이는 테스팅 작업에 많은 노력이 들게 한다.

이 논문의 주요 목적은 현재 사용되고 있는 안드로이드 기반 GUI 자동화 테스트 기법보다 발전되고 최적화된 캡처 리플레이(Capture-Replay) 방법을 제안하고 이의 효율성과 호환성을 증명하기 위한 것이다. 이를 위하여 캡처 리플레이 기법을 적용한 안드로이드 자동화 테스트 방법을 먼저 살펴보고 캡처 단계에 기록된 스크립트를 통해 리플레이 단계에 테스트 케이스를 자동으로 생성하고 테스트를 실행하는 방법을 기술한다.

기존의 캡처 리플레이 기법의 의미를 반영하여 XML을 이용한 주요 위젯기반 요소를 추출 방법을 고안한다. 또한 위젯기반 API 이벤트 처리 방법을 제안한다. 제안한 방법을 적용하여 캡처 단계에서 모니터링한 터치 이벤트가 일어나는 형태를 추적한 후 리플레이 단계에서 위젯의 속성과 이벤트를 스크립트로 전환함으로써 테스트 케이스를 생성한다.

이 논문은 총 다섯 장으로 구성되어 있다. 2장에서는 안드로이드 GUI 테스팅 현황과 기존 테스팅 기술들을 살펴보고 3장에서는 제안한 개선 기법을 실현하기 위한 개념과 기술을 설명한다. 4장에서는 3장에서 언급한 기법과 시험사례, 위젯 기반으로 요소를 추출하는 것부터 테스팅 결과 검증에 이르기까지 설명하고, 테스팅의 효율성과 호환성, 위젯식별의 정확성을 살펴본다. 마지막으로 5장에서 논문의 결론을 맺는다.

2. GUI 자동화 테스트 기법

GUI 테스트 자동화는 80년대 말부터 연구하기 시작했다. 현재 GUI 자동 테스트에 사용되는 세 가지기법은 캡처 리플레이 기법, 명세 기반 테스트 기법, 베타 테스트 기법이다.

캡처 리플레이 기법은 사용자가 소프트웨어를 작동해서 일어나는 마우스 이벤트와 키보드 이벤트를 스크립트 언어[5]로 기록하고 기록된 스크립트를 재생하여 소프트웨어의 GUI 테스트에 사용한다. 현재 대부분 GUI 테스트 자동화 도구들은 이런 방식을 이용한다. 그러나 캡처 리플레이 과정은 테스터의 능력에 전적으로 의존한다. 다양한 시나리오를 이용하여 테스트하기 위해 많은 기록 과정이 필요하며, GUI를 변경한다면 같은 작업을 반복해야 한다. 또한 이벤트를 기록할 때 사소한 실수도 테스트 수행에 치명적인 영향을 끼칠 수 있다. 캡처 리플레이 기술은 단순히 마우스 이벤트와 키보드 이벤트의 내용만을 기록하기도 하지만 테스트 명세가 없다면 많은 양의 테스트 케이스를 감당하기 어려울 것이다.

위와 같은 단점을 보완하기 위하여 명세 기반 테스트 기법이 만들어졌다[8]. 명세 기반 기법은 GUI의 명세를 기반으로 애플리케이션을 테스트한다. 테스트 명세는 테스트 케이스와 테스트 스크립트를 비교하기 때문에 예상되는 전체 소프트웨어의 동작이나 기능을 간결한 방법으로 표현한다. 엄격한 명세 기반 테스트 수행은 요구사항이나 디자인명세가 일괄적이고 완벽하며 분명하게 기술될 것을 요구한다. 명세 기반 테스트는 정적인 세부분석을 통해서 전형적이고 중요한 시나리오를 얻기 위한 기초를 제공한다. 요구분석 명세에 정확한 시스템 동작이 명시되면 테스터는 테스트 케이스를 조직적으로 얻을 수 있다. 하지만 명세 기반 GUI 테스트는 높은 수준의 GUI 명세를 요구한다. 이런 명세를 얻는데 많은 노력이 필요하고 얻어진 명세를 목표 애플리케이션에 연결하여 직접적 테스트를 수행하기가 어렵다[9].

베타 테스트는 GUI 테스트 중에 가장 인기 있는 방법이다. 베타 테스트는 베타 버전의 소프트웨어 복사본을 출시하여 일반 사용자들에게 테스트의 일부, 즉 GUI 테스트를 수행하게 하는 것이다. 그 예로 마이크로소프트사는 Windows 95 베타 버전의 소프트웨어 복사본을 약 400,000개를 출시하였다[2]. GUI 테스트 방법 중 가장 많은 오류를 찾을 수 있는 방법이지만 비전문적인 테스터들로 구성되어 그 전문성 이 떨어지고 많은 일반 사용자를 구하고 교육시키는 데에 어려움이 따른다.

2.1 안드로이드 기반 GUI 테스트 프레임워크

안드로이드 모바일 플랫폼은 달빅 가상 머신을 제공하면서 일반적인 GUI 테스트 도구 중 <표 1>의 테스트 프레임워크를 제공하고 있다.

단위 수준의 명세 기반 테스트 기법은 클래스 내부를 코드 수준으로 들여다 볼 수 있는 방법을 지원한다. 대표적인

〈표 1〉 안드로이드 기반 테스트 프레임워크

W : White, B : Black, A : Auto, R : Regression			
프레임워크 이름	적용 단계	테스트 기법	제공자
JUnit x	단위 테스트	W	구글
Android Test Framework(ATF)	모두	W, B, R	구글
Positron[16]	통합 테스트 및 기능 테스트	W, A	개인별
Robotium	통합 테스트 및 시스템 테스트	B, A	개인별
Sikuli	기능 테스트	A(GUI)	시쿨리
몽키 테스트	스트레스 테스트	A(무작위 이벤트)	구글

단위 테스트 프레임워크는 JUnit이다. 테스트 명세 기준으로 클래스의 구조 안에 있는 메소드와 변수, 비교에 필요한 테스트 오라클을 중심으로 테스트 케이스를 작성한다[2].

안드로이드 애플리케이션은 네 가지 기본 구성요소, 즉 액티비티, 서비스, 방송 수신자(Broadcast Receiver), 콘텐츠 제공자이다. 이렇게 다양한 구성 요소가 있는 상황에서는 기존 JUnit 프레임워크를 적용할 수 없다. 이것이 바로 안드로이드 테스트 프레임워크가 나온 이유이다.

GUI 테스트에 대하여 논하려면 안드로이드 기반 GUI 구성 요소를 소개하여야 한다. 안드로이드 애플리케이션의 화면을 구성하는 주요 단위는 액티비티이다. 주로 액티비티는 뷰로 구성된 유저 인터페이스를 화면에 표시하고 사용자의 입력을 처리하는 역할을 한다. 그러나 액티비티 자체는 화면에 직접 보이지 않으며 액티비티 안의 뷰가 바로 사용자에게 보이는 실체다[16]. 뷰는 안드로이드의 사용자 인터페이스를 구성하는 핵심 컴포넌트로서 화면상의 사각 영역을 차지하며 UI 요소, 예를 들어 에디트 텍스트, 텍스트 뷰, 버튼, 체크 박스, 라디오 버튼, 이미지 뷰 등을 그려주고 사용자로부터 입력을 받아들인다. 뷰는 위젯을 직접 보이는 것과 직접 보이지 않으며 다른 뷰를 담는 컨테이너 역할을 하는 것이 있다.

기존 안드로이드 테스트 프레임워크는 모든 테스트 상황을 다 포함하고 있지만 어떤 특정한 문제를 해결할 수가 없다. 특히 GUI 테스트를 하는 경우에 원래의 JUnit을 이용한 방법으로는 진행이 번거롭고 효율성이 낮은 단점이 있다. 따라서 안드로이드 GUI 테스트를 하기 위한 특수한 테스트 도구들이 출시되었고 이를 살펴본다.

Robotium GUI 테스트 프레임워크는 JUnit을 기반으로 작동하며 안드로이드 테스트 플랫폼 기술을 적용하여 애플리케이션의 동작 사나리오를 작성하는 테스트 도구이다. 개발 명세를 통해 애플리케이션 실행 흐름을 파악하고 GUI 테스트 시나리오를 자동 추출한다. 추출한 시나리오를 기반으로 자동화 테스트를 실행한다. 테스트 결과를 시나리오 기준으로 JUnit을 통해서 확인할 수 있다.

Sikuli는 GUI의 스크린샷을 사용하여 시각적으로 테스트하는 자동화된 도구이다[18]. Sikuli는 Sikuli 스크립트, Jython에 대한 시각적 스크립팅 API를 포함하고 있고 쉽게 스크린샷과 시각적인 스크립트를 작성하기 위하여 통합 개발 환경이 포함되어 있다. 안드로이드를 기반으로 하는 특수 GUI 테스트 도구는 아니지만 Sikuli 스크립트 도구를 이용하여 안드로이드 실행 화면을 캡처하면 자동화 테스트를 실행할 수 있다.

위에서 언급한 프레임워크를 이용하여 시스템 수준의 기능별 GUI 테스트 시험을 자동화 하려면 필히 명세를 중심으로 테스트 시나리오를 만들어서 작업하여야 한다[15]. 시나리오를 구성할 때는 주로 사용사례나 다이어그램을 적용하는데 특히 사용자 동작의 연속적인 순서를 사용 사례로 파악하고 이를 이용하여 테스트 사례를 생성한다. 다음에 테스트 데이터와 테스트 오라클을 묶어서 테스트 케이스를 만든다. 따라서 테스트를 자동화하기 위해서 테스트 케이스를 이용하여 사용자 동작을 모의하고 수동 테스트처럼 재현한다. 이런 작업을 제대로 하려면 명세를 잘 정리하여야 한다. 또한 테스트 케이스도 이에 맞게 바꾸어야 한다. 이상 모든 작업을 모두 수동으로 한다면 똑같은 방식으로 관리하고 유지하여야 하기 때문에 테스트 결과의 정확도와 효율성이 낮아진다.

2.2 캡처 리플레이 기법

캡처 리플레이 과정은 사용자가 발생시키는 이벤트와 실행 화면을 기록한 후 기록한 내용을 재현 단계로 이루어진다. 즉 테스트 시나리오에 의하여 AUT를 수동으로 테스트해보는 동시에 기록하는 것이 캡처다. 기록된 스크립트를 통하여 테스트 케이스를 생성하고 이를 이용해서 테스트를 다시 실행하는 것을 재현(replay)이라 부른다. 테스트 결과는 캡처 단계에 얻은 예상 화면을 기준으로 재현 단계에 받은 실제 화면과 비교하는 방식으로 확인된다[14].

캡처 리플레이 기법은 단순하기 때문에 이 기술을 이용해서 테스트 도구를 구현하기가 용이하다. 하지만 이 기술은 앞에 서술했던 세 가지 중요한 문제점을 가지고 있다. 비록 캡처 리플레이 기법은 실제 명세 기반을 도입하여 이런 문제들을 완화될 수 있겠지만 높은 수준의 GUI 명세를 기술하는데 많은 노력이 필요하다. 또한 대부분 테스트는 AUT가 이미 존재하는 상황에 진행된다. 캡처 리플레이 기법은 이런 단점으로 안드로이드 기반 자동화 테스트를 에뮬레이

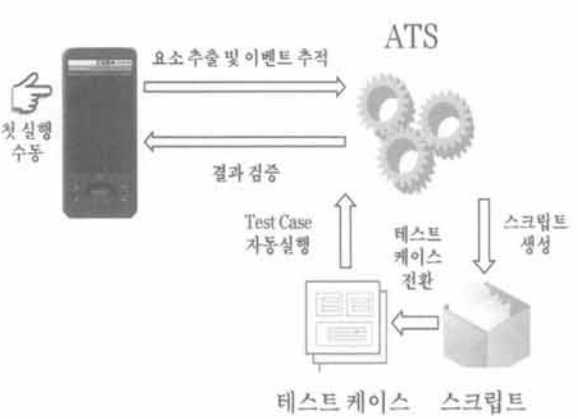
터에서 실행할 때 그대로 적용할 수가 없다. 또한 안드로이드 애플리케이션 GUI 테스트를 실제 디바이스에서 실행할 때도 역시 기존 캡처 기법은 이용할 수가 없다. 또한 기존 캡처 기법처럼 좌표 값을 기록하면 다양한 모바일 디바이스에 대한 호환성이 낮아지는 단점이 있다. 더구나 생성된 산출물은 스크립트만 있고 테스트 케이스가 없기 때문에 실제 디바이스에서 동작되지 않아 기존 레코드 리플레이 기법을 사용할 수 없다. 마지막으로 기존 스크립트는 많은 정보를 최적화 하지 못하고 수많은 스크립트를 유지하여야 하며 따라서 효과적으로 재사용할 수가 없다.

본 논문에서는 이러한 문제점을 해결하기 위하여 캡처 리플레이를 개선한 안드로이드 위젯의 기능 테스트 자동화 기법을 제안하였다. 제안한 기법은 명세 기반 GUI 시험을 위하여 모든 뷰 노드를 추출하고 이를 통해서 자바 테스트 케이스를 자동 생성하고 재현한다.

3. GUI 위젯 기능 테스트 자동화 설계 및 구현

안드로이드 기반의 GUI 위젯 기능 테스트 자동화 방법을 소개하기 위하여 기존의 캡처 리플레이 기법을 적용하여 테스트 케이스 산출물까지 확장하여 개선하는 방법에 대하여 소개한다. 안드로이드 위젯 기능 테스트 자동화 실행 절차는 (그림 1)과 같다. 여기서 ATS(Automatic Test System)를 통해서 스크립트 생성 단계가 바로 구체적인 캡처 작업이 될 수 있다. 즉 캡처작업은 실행 화면의 레이아웃 요소 추출을 시작하는 동시에 모니터링 프로그램을 이용하여 이벤트 좌표 값을 기록한다. 요소 레이아웃 구조와 이벤트 좌표 값 리스트까지 연결하여 스크립트를 자동으로 생성한다. 이러한 스크립트를 이용하여 자동 전환된 테스트 케이스로 실제 디바이스에서 테스트 시험을 실행한다. 실행된 결과의 검증은 시각적인 표현을 통해 직관적으로 이해할 수 있는 기법을 지원한다.

제안된 방법은 우선 테스트 스크립트를 생성하고 이를 테스트 케이스로 전환하며 마지막으로 시험결과 검증을 하는 세 가지 세부 작업으로 나눌 수 있다. 시험 결과 검증 결과 오류가 발견되었다면 위치를 찾아 디버깅을 하게 된다. 프로그램 수정 후에 회귀테스트를 위하여 테스트 케이스를 다시 만들 필요가 없다. 이미 캡처된 UI 인터랙션으로 생성한 스크립트가 있기 때문이다. 만일 프로그램 수정으로 인하여 일부 테스트 케이스가 바뀌어야 한다면 그 부분에 대한 스크립트를 수정하여 회귀 테스트를 용이하게 할 수 있다.



(그림 1) 위젯 기능 테스트 자동화 실행 절차

3.1 위젯기반 레이아웃 요소 추출

안드로이드 기반 애플리케이션의 위젯 정보를 추출하려면 안드로이드 시스템 내부와 외부 시스템 사이의 정보를 교환할 수 있는 통로를 만들어서 내부 정보를 밖으로 출력할 수 있는 방법이 필요하다. 따라서 안드로이드 기반 GUI 위젯을 테스트하기 위한 첫걸음은 어떤 통신 방식으로 외부 시스템과 내부 시스템을 연동할 것인가를 연구하는 것이다.

안드로이드 SDK 중에는 강력한 도구들이 많다. 그 중에 hierarchyViewer라는 시뮬레이터는 애플리케이션 레이아웃의 요소를 추출해줄 수 있다. 이 도구를 이용하면 현재 실행하고 있는 애플리케이션 레이아웃을 트리형태로 시각적으로 보여준다. 각 위젯은 트리상의 노드로 표현되며, 노드를 클릭하면 해당 위젯의 세부 정보를 보여준다. 이 세부정보를 통하여 해당 위젯의 오류를 눈으로 확인할 수 있다. 간단한 계산기 애플리케이션의 위젯을 hierarchyViewer를 사용하여 추출한 결과는 (그림 2)와 같다.

(그림 2) hierarchyViewer로 추출한 레이아웃 일부 캡처화면

(그림 2)의 왼쪽은 예시 애플리케이션을 에뮬레이터 위에서 실행시킨 화면이다. 사용자가 숫자 버튼을 이용하여 연산자를 입력할 수 있다. 화면 오른쪽에 각 연산 버튼을 클릭하면 더하기, 빼기, 곱하기, 나누기 연산을 구할 수 있다. 사각형 ① 안의 버튼들이 추출하여야 하는 위젯이다. 버튼들은 각각이 선형 레이아웃 나열방식으로 나타내었다. 오른쪽은 추출한 위젯을 트리형태로 보여준 결과이다. 추출된 위젯은 트리의 노드로 표현되며 각 노드에서 해당 위젯 ID, 위젯 자료유형과 위젯 해시코드 등 테스트를 위한 정보를 얻을 수 있다.

레이아웃 위젯요소를 추출하려면 에뮬레이터에 AUT가 설치되어 있어야 한다. 우선 자동화 테스트 시스템 ATS를 실행하고 현재 연결할 수 있는 디바이스를 탐지한다. 실제

(그림 3) 레이아웃 요소 추출 절차

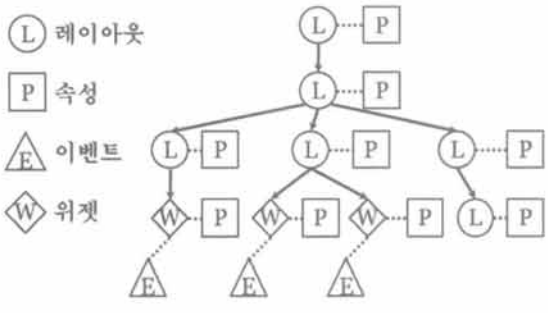
디바이스가 없고 에뮬레이터를 적용한 경우에는 탐지 결과가 emulator-xxx로 표시되는데 xxx는 에뮬레이터의 전화번호이다. 다음 단계는 연결할 수 있는 디바이스를 찾고 ATS가 ADB(Android Debugging Bridge) 명령어를 이용해서 AUT를 디바이스에 설치해준다. 설치된 후에 자동으로 실행된 AUT는 최초 실행 상태에서 레이아웃 위젯 요소들을 추출하기 위하여 표준 소켓 통신 방식으로 안드로이드 시스템과 ATS 사이의 통신을 채널을 설정한다.

(그림 4)는 정규화 시키기 위한 정규 표현식 패턴이다. 모든 위젯의 추출한 결과를 보면 전치사 'android.widget.'로 시작한다. 점(.)에서 '@'까지 사이의 모든 내용을 '.+?'이라는 매칭 캐릭터를 이용하여 간과시킬 수 있다. 위젯 해시 코드, 위젯 Id와 위젯 테스트 등 정보도 이와 같은 방식으로 추출할 수 있다.

```
14    protected final String DecorViewExp = "PhoneWindow$DecorView";
15    protected final String LinearLayoutExp = "LinearLayout";
16    protected final String FrameLayoutExp = "FrameLayout";
17    protected final String TableLayoutExp = "TableLayout";
18    protected final String TableRowExp = "TableRow";
19
20    protected final String widgetTypeExp = "android.widget..+?@";
21    protected final String widgetHashExp = "@.+?\\s";
22    protected final String widgetIdExp = "id/..+?\\s";
23    protected final String widgetTextExp = "mText=.+?get";
```

(그림 4) 정규 표현식 매치 패턴

화면 요소 추출결과가 시각적으로 재현될 필요는 없다. 다만 일반 문자열 유형으로 얻어 나중 단계인 테스트 스크립트 생성을 위하여 추상 레이아웃 위젯 정보 스트링들을 트리구조로 저장할 필요가 있다. 생성된 레이아웃 위젯 트리 구조는 (그림 5)와 같다. 모든 요소의 레이아웃과 각 위젯들은 모두 트리 노드로 표현된다.

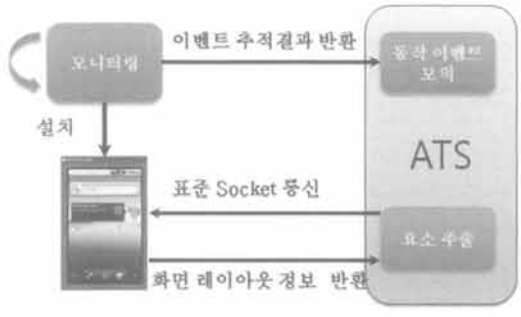


(그림 5) 추상 레이아웃 요소 트리

3.2 이벤트 모니터링

안드로이드 GUI에서 발생하는 이벤트는 터치 스크린과 키보드 입력이다. 터치 이벤트는 손가락으로 누르는 동작과 손가락을 떼는 동작, 누르면서 옮기는 동작이 있다. 세 가지 터치 이벤트를 이용해서 서로 이어지는 복잡한 이벤트를 만들어 낼 수 있다. 마찬가지로 키패드를 사용하는 경우에 사용자가 키패드를 누르고 키를 떼어 주는 두 가지 동작 이벤트가 있다.

외부 시스템은 내부 안드로이드 시스템에서 발생하는 이러한 이벤트를 동적으로 감지할 수 없다. 따라서 이런 이벤트들을 감지할 수 있는 모니터링 프로그램의 도움을 받아야 한다. 모니터링 프로그램은 사용자가 동작시킨 이벤트의 위치와 종류까지 추적하며, 추적한 결과는 스크립트 생성 단계에서 중요하게 사용된다. 또한 이벤트가 어떤 위젯에서 발생하였는지를 정확하게 알고 있으면 재현 단계에서 어느 위젯을 동작시키고 어떤 이벤트가 일어나야 되는지를 자동으로 알 수 있다.



(그림 6) 이벤트 모니터링 절차

앞서 소켓 통신 방식으로 내부 안드로이드 시스템과 외부 시스템을 연동하여 동적으로 레이아웃 요소를 추출하고 실행과정 중에 실시간 포커스를 주었던 위젯 정보도 같이 뽑을 수 있다. 하지만 출력한 정보 중에 일어난 이벤트 정보는 포함되지 않았다. 따라서 위젯과 이벤트 사이의 관계를 이어주기 위해서 추출된 요소 트리와 모니터링 프로그램에서 추적한 이벤트들의 리스트를 연결시켜야 한다. 다시 말하면 소켓 통신 방식을 통하여 동적으로 추출된 요소의(트리 노드) 정보로 해당 위젯의 유형이나 자료 값을 얻을 수 있다. 동시에 모니터링 프로그램을 통하여 하위레벨의 원시 이벤트들을 수집, 조합하여 상위 레벨의 위젯 이벤트를 알아 낼 수 있다.

3.3 테스트 케이스 전환 방법

앞서 설명한바와 같이 안드로이드 애플리케이션은 고립된 시스템 환경에서 실행되기 때문에 테스트 케이스를 외부 시스템에서 내부 안드로이드 시스템으로 전송해 주는 수단이 필요하다. 안드로이드 애플리케이션을 디버깅할 때 ADB를 많이 적용하는데 이를 이용하여 테스팅을 위한 특정한 동작을 시뮬레이션해 주고 이를 내부 시스템으로 보내 주는 방법을 사용하면 가능하다.

테스트 케이스는 스크립트와 안드로이드의 레이아웃 트리 구조를 참조하여 자동으로 만들어 낼 수 있다. 먼저 스크립트를 만들기 위하여 추상 트리와 모니터링 프로그램을 통하여 추적한 이벤트들을 통합시킨다. 즉 위젯과 상응하는 이벤트들을 연결시켜주는 작업이다. 안드로이드 레이아웃의 위젯을 추출한 결과 XML과 이벤트 추적 결과 XML을 통합시킨 결과는 (그림 7)과 같다. 통합된 XML파일을 표준화하여 안드로이드 GUI 테스트 자동화 스크립트를 생성할 수 있다.

```
<TreeNode>
  <Layout id="" type="LinearLayout" />
  <Property id="44ee6340" text="Left Operand" type="Label" />
  <Event>...</Event>
</TreeNode>
<TreeNode>
  <Layout id="44ee6820" type="LinearLayout" />
  <Property id="textView1" text="2" type="EditText" />
  <Event level="low" type="single"><axis scrollX="0" scrollY="0" /></Event>
</TreeNode>
<TreeNode>
  <Layout id="" type="LinearLayout" />
  <Property id="44ee7910" text="Right Operand" type="Label" />
  <Event>...</Event>
</TreeNode>
<TreeNode>
  <Layout id="44ee6340" type="LinearLayout" />
  <Property id="textView1" text="1" type="EditText" />
  <Event level="low" type="single"><axis scrollX="0" scrollY="0" /></Event>
</TreeNode>
<TreeNode>
  <Layout id="" type="LinearLayout" />
  <Property id="" text="" type="" />
  <Event>...</Event>
</TreeNode>
<TreeNode>
  <Layout id="44eeb300" type="LinearLayout" />
  <Property id="plusbtn" text="+" type="Button" />
  <Event level="low" type="single"><axis scrollX="0" scrollY="0" /></Event>
</TreeNode>
```

(그림 7) 정보 통합시킨 후 XML 파일

ATS 자동 테스트 시스템은 위와 같이 생성된 XML을 통해서 실행 테스트 케이스를 생성한다. 앞서 플레이 기법은 키 이벤트 시뮬레이션과 터치 이벤트 시뮬레이션 두 가지가 있다고 설명하였다. 우선 키 이벤트 모의 테스트 케이스의 생성 결과가 (그림 8)에 있다. 추상 레이아웃 트리를 통해서 요소 각각의 순서위치를 확인한 후, 방향키 이벤트 모의 방법을 이용하여 안드로이드 GUI 테스트 자동화 테스트 케이스를 생성한다. 생성된 테스트 케이스는 최종적으로 자동화 테스트를 할 때 사용된다.

```
public void test_key_02(){
    //2
    ary_key.add(new KeyTouch(KEYCODE.KEYCODE_DPAD_DOWN));
    ary_key.add(new KeyTouch(KEYCODE.KEYCODE_DPAD_DOWN));
    ary_key.add(new KeyTouch(KEYCODE.KEYCODE_DPAD_RIGHT));
    ary_key.add(new KeyTouch(KEYCODE.KEYCODE_DPAD_CENTER));
    //-
    ary_key.add(new KeyTouch(KEYCODE.KEYCODE_DPAD_RIGHT));
    ary_key.add(new KeyTouch(KEYCODE.KEYCODE_DPAD_RIGHT));
    ary_key.add(new KeyTouch(KEYCODE.KEYCODE_DPAD_DOWN));
    ary_key.add(new KeyTouch(KEYCODE.KEYCODE_DPAD_CENTER));
    //1
    ary_key.add(new KeyTouch(KEYCODE.KEYCODE_DPAD_UP));
    ary_key.add(new KeyTouch(KEYCODE.KEYCODE_DPAD_LEFT));
    ary_key.add(new KeyTouch(KEYCODE.KEYCODE_DPAD_LEFT));
    ary_key.add(new KeyTouch(KEYCODE.KEYCODE_DPAD_LEFT));
    ary_key.add(new KeyTouch(KEYCODE.KEYCODE_DPAD_CENTER));
    //=
    ary_key.add(new KeyTouch(KEYCODE.KEYCODE_DPAD_DOWN));
    ary_key.add(new KeyTouch(KEYCODE.KEYCODE_DPAD_DOWN));
    ary_key.add(new KeyTouch(KEYCODE.KEYCODE_DPAD_DOWN));
    ary_key.add(new KeyTouch(KEYCODE.KEYCODE_DPAD_RIGHT));
    ary_key.add(new KeyTouch(KEYCODE.KEYCODE_DPAD_RIGHT));
    ary_key.add(new KeyTouch(KEYCODE.KEYCODE_DPAD_CENTER));
    //
}
```

(그림 8) 키 이벤트 모의 테스트 케이스 생성

터치 이벤트 시뮬레이션은 먼저 추상 레이아웃 트리를 통해서 각 위젯이 실행된 순서와 터치 이벤트 정보, 좌표 값을 얻는다. (그림 9)의 사례는 추적한 좌표 값을 이용하여 생성된 '2-1='와 '2+1=' 사례에 대한 테스트 케이스이다. 모든 터치 포인트의 집합은 PointTouch 유형의 리스트에 묶여서 실행될 때 테스트 메소드를 호출하고 리스트 중에 등록된 터치 포인트를 하나씩 실행하면 모의 터치 이벤트가 일어난다. Test 생성자 메소드는 테스트 슈트와 비슷한 역할을 한다. 즉, 테스트 메소드들이 순서대로 모여서 하나의 테스트 슈트를 구성한다.

```
public TouchTestCase(){
    test_01();
}

public void test_01(){// 2 - 1=
    ary_touch.add(new PointTouch(136,199));
    ary_touch.add(new PointTouch(268,264));
    ary_touch.add(new PointTouch(49,212));
    ary_touch.add(new PointTouch(211,381));
}

public void test_02(){// 2 + 1=
```

(그림 9) 터치 이벤트 모의 테스트 케이스 생성

4. GUI 위젯 기능 테스트 실행 및 결과 검증

본 논문에서 제안한 안드로이드 GUI 위젯 기능테스트 자동화 캡처 리플레이 기법을 실험하기 위해 오류가 포함된 안드로이드 응용프로그램 계산기(CalculatorIX)을 이용하였다. 먼저 기능 시험을 위하여 사용사례 슬라이스를 중심으로 시험 사례를 작성한다. 연산기능을 시험하기 위해 우선 버튼들의 이벤트가 잘 일어나는지를 확인하여 야 한다. 만약 버튼을 누른 후 화면입력창에 나타나는 값이 예상 결과와 다르다면 이 버튼은 오류가 있는 것이 분명하다. 그러므로 연산기능을 테스트하기 전에 버튼 이벤트 동작이 잘 구동되는지를 확인 하여야 한다. 따라서 버튼들의 이벤트 테스트를 올바르게 진행할 수 있는 여부가 본 논문의 ATS 자동화 테스트 시스템에서 가장 중요한 부분이다. ATS를 이용하여 안드로이드 시스템에서 사용되는 위젯을 정확하게 식별, 추출하는 것을 테스트한 결과 사용하는 대부분의 위젯에 대하여 정확히 식별됨을 확인하였다. 불일치된 버튼은 추출된 위젯 정보 중에 Unicode '√'를 인식하지 못하는 때문인데 이것이 발견된 결함이다. 그 외에 애플리케이션을 개발 할 때 사용하는 위젯은 에디드 텍스트, 버튼, 라디오 버튼, 레이팅 바, 시크 바, 스피너 등도 테스트하고 결과를 검증하였다.

연산기능 시험은 버튼 이벤트 시험보다 더 복잡하다. 우선 첫 번째 피연산자를 입력하고 연산자 버튼을 누른다. 다음 두 번째 피연사자를 입력하고 (=) 버튼을 누르면 결과가 출력된다. 이 과정을 따라서 연산자 기능시험을 진행하였고 계산 결과를 구하는 (=) 버튼의 동작시험도 같이 진행하였다. 단순한 계산결과를 구하는 시험보다는 블랙박스 테스트

기법을 적용하여 경계 테스트를 수행하는 것이다. 앞서 각 버튼들은 이미 테스트했기 때문에 결과 값이 예상 결과 값과 일치하지 않으면 계산 기능에 오류가 있는 것이다.

〈표 2〉 안드로이드 GUI 위젯 식별 결과

위젯 Value	추출 ID	추출 Value	위젯 속성	추출 결과	정확도
계산기					
1	btn1	1	Button	Button	일치
2	btn2	2	Button	Button	일치
3	btn3	3	Button	Button	일치
4	btn4	4	Button	Button	일치
5	btn5	5	Button	Button	일치
6	btn6	6	Button	Button	일치
7	btn7	7	Button	Button	일치
8	btn8	8	Button	Button	일치
9	btn9	9	Button	Button	일치
0	btn0	0	Button	Button	일치
C	cbtn	C	Button	Button	일치
CE	cebtn	CE	Button	Button	일치
.	pointbtn	.	Button	Button	일치
+	plusbtn	+	Button	Button	일치
-	minusbtn	-	Button	Button	일치
*	mulbtn	*	Button	Button	일치
/	divbtn	/	Button	Button	일치
=	equalbtn	=	Button	Button	일치
+/-	signumbtn	+/-	Button	Button	일치
x^	powerbtn	x^	Button	Button	일치
√x	rootbtn	?^	Button	Button	불일치
라디오 그룹					
Snack	snack	Snack	Radio Button	Radio Button	일치
Breakfast	breakfast	Breakfast	Radio Button	Radio Button	일치
Lunch	lunch	Lunch	Radio Button	Radio Button	일치
Dinner	dinner	Dinner	Radio Button	Radio Button	일치
All of them	all	All of them	Radio Button	Radio Button	일치
Rating 바					
☆☆☆	ratingbar1	n/a	Rating Bar	Rating Bar	일치
☆☆☆☆☆	ratingbar2	n/a	Rating Bar	Rating Bar	일치
☆☆☆☆☆	small_rating bar	n/a	Rating Bar	Rating Bar	일치
☆☆☆☆☆	indicator_rat ingbar	n/a	Rating Bar	Rating Bar	일치
Seek 바					
n/a	seek	n/a	Seek Bar	Seek Bar	일치
Spinner					
n/a	spinner1	n/a	Spinner	Spinner	일치
n/a	spinner2	n/a	Spinner	Spinner	일치

명세기반 블랙박스 테스트 기법을 적용한 결과는 숫자 출력 문제 때문에 불일치로(예, 정수3과 실수3.0) 표시되었다. 하지만 실제 테스트 결과 값이 예상 결과 값과 같기 때문에 계산기 코드를 수정할 때 숫자를 출력하는 부분 코드만 수정하면 된다. 여기서 알 수 있듯이 본 논문 제안한 캡처 리플레이 방법은 높은 자동화 테스트 정확도를 유지할 수 있다. 연산자 자동화 테스트 결과는 <표 3>와 같다.

〈표 3〉 계산기 연산자 버튼 기능 테스트 결과

계산기 경계 테스트					
No.		테스트 데이터	예상 결과	실제 결과	검증 결과
1	버튼 +	2 +1	3	3.0	불일치
		10^308+1	1.0E308	1.0E308	일치
2	버튼 −	2 - 1	1	1.0	불일치
		10^(-308)-1	-1	-1.0	불일치
3	버튼 x	2 * (-2)	-4	-4.0	불일치
		10^308*10^2	Error!	Error!	일치
4	버튼 ÷	2 / (-2)	-1	-1.0	불일치
		10^(-308)/10^2	0	0.0	불일치
5	버튼 =	N/A	N/A	N/A	N/A

모바일 애플리케이션을 개발할 때 또 한 가지 중요한 이슈는 다양한 디바이스에서 GUI 기능 테스트가 필요하다는 점이다. 왜냐하면 디바이스마다 해상도 등의 차이가 있기 때문이다. 모든 해상도에서 테스트하려면 많은 시간이 소요되며, 기존 기법들로 자동테스트를 수행할 때 해상도가 달라지면 정확한 테스트를 할 수가 없다. 제안한 방법을 이용하면 다양한 해상도에도 불구하고 위젯의 기능 테스트를 자동으로 실행할 수 있다.

〈표 4〉 다양 해상도에 호환성 테스트 결과

에뮬레이터 (IBM X41 + Android 2.2)					
No.		테스트 데이터	해상도	키 이벤트/ 시간	터치 이벤트/시간
1	+	2 + 1	320*480	가능 / 63s	가능 / 7s
			240*320	가능 / 62s	오류 값
2	−	2 - 1	320*480	가능 / 69s	가능 / 7s
			240*320	가능 / 69s	오류 값
3	×	2 * (-2)	320*480	가능 / 93s	가능 / 8s
			240*320	가능 / 91s	오류 값
4	÷	2 / (-2)	320*480	가능 / 99s	가능 / 8s
			240*320	가능 / 100s	오류 값

<표 4>의 테스트 결과를 보면 계산기의 연산자 기능테스트를 할 때 HVGA(320*480)해상도에서 키 이벤트 기법과 터치 이벤트 기법은 모두 올바른 결과를 구할 수 있다. 하지만 에뮬레이터의 해상도를 QVGA(280*320)으로 바꾸고 테스트 케이스를 이용하여 다시 실행하면 키 이벤트 기법으로 생성된 테스트 케이스만 똑같은 결과를 얻을 수 있고 터치 이벤트 기법으로 생성된 테스트 케이스는 오류 값이 얻어진다. 터치 이벤트 기법은 좌표 값에 기반을 두어 테스트를 실행하기 때문에 해상도가 달라졌을 때 잘못된 결과가 얻어지는 것이기 때문이다. 하지만 본 논문에서 제안한 키 이벤트 기법은 좌표 값에 기반을 두지 않기 때문에 다양한 해상도 모드에서 사용가능하다.

5. 결론 및 향후 연구

이 논문에서 제안한 안드로이드 테스트 자동화 기법은 XML을 이용하여 기존 GUI 캡처 리플레이 방법을 개선한 것이다. 개선된 안드로이드 테스트 자동화 기법은 애플리케이션 레이아웃 요소추출 방법과 동작 이벤트 추적 방법을 이용하여 테스트 케이스를 자동으로 생성한다. 생성된 테스트 케이스를 ADB를 이용하여 재실행하면 실제 테스트 결과를 구할 수 있다. 예상 결과와 실제 결과를 비교하면 애플리케이션 내부에서 문제가 있는지 알 수 있다.

본 논문에서는 다양한 테스트 경우를 커버하기 위하여 키 이벤트 시뮬레이션과 터치 이벤트 시뮬레이션 두 가지 테스트 케이스를 실행방법을 제안하였다. 키 이벤트 시뮬레이션은 버튼이나 메뉴의 선택 이벤트들을 확인하기 위하여 사용하며 터치 이벤트 시뮬레이션은 디바이스 해상도의 차이 등 에 따른 이상 현상을 확인하기 위하여 사용한다.

이 논문에서 제안된 자동화 테스트 방법은 화면전환이 빈번한 경우 요소 추출에 오류를 일으킬 수 있어 단일 화면에서 장점을 보이고 있으며, 향후 연구에서는 이러한 화면 전환에서 생기는 문제들을 해결할 필요가 있다. 그리고 본 논문에서 제안된 방법을 적용한 자동화 테스트 도구의 개발도 다음 연구단계에서 고려하여야 할 내용이다.

참 고 문 헌

[1] A. M. Memon, M. E. Pollack and M. L. Soffa, "Hierarchical GUI test case generation using automated planning," IEEE Transactions on Software Engineering, Vol.27, No.2, pp.144-155, 2001.

[2] A. M. Memon, "GUI Testing: Pitfalls and Process," IEEE Computer, Vol.35, pp.87-88, 2002.

[3] Automated GUI Testing, Tessella Support Services plc, January 1999, http://www.tssp.co.uk/Literature/Supplements/autogui.htm

[4] A. Jaaskelainen, M, Katara, A, Kervinen, M, Maunumaa, T. Paakkonen, T. Takala and H. Virtanen, "Automatic GUI Test Generation for Smart Phone Applications - an Evaluation," ICSE - Companion 2009. 31st International Conference on Software Engineering - Companion Volume, pp.112-122, 2009.

[5] B. Mu, MK. Zhan, LF. Hu, "Design and Implementation of GUI Automated Testing Framework Based on XML," WCSE '09 Proceedings of the 2009 WRI World Congress on Software Engineering, Vol.4, pp.194-199, 2009.

[6] C. Lowell and J. Stell-Smith, "Successful Automation of GUI Driven Acceptance Testing," Lecture Notes in Computer Science, Springer-Verlag Heidelberg, Vol.2675, pp.331-333, 2003.

[7] J. Chen and S. Subramaniam, "Specification-based Testing for GUI-based Applications," Software Quality Journal, Vol.10, No.3, pp.205-224, 2002.

[8] M. Assem, A. Keshk, N. Ismail, H. Nassa, "Specification - Driven Automated Testing of Java Swing GUIs Using XML," ITI 5th International Conference on Information and Communications Technology, pp.84-88, 2009.

[9] M. A. Abdel Salam, A. E. Keshk, N. A. Ismail, H. M. Nassar, "Automated Testing of Java Menu-Based GUIs Using XML Visual Editor," ICCES '07. International Conference on Computer Engineering & System, pp.313-318, 2007.

[10] W. K. Chen, T. H. Tsai, and H. H. Chao, "Integration of Specification based and CR-based Approaches for GUI Testing," AINA'05, 19th International Conference on Advanced Information Networking and Applications, Vol.1, pp.967-972, 2005.

[11] Y. Sun and E. L. Jones, "Specification-Driven Automated Testing," Proceedings of the 42ed Annual Southeast Regional Conference, pp.140-145, 2004.

[12] 김상형, 안드로이드 프로그래밍 정복, 한빛미디어, 2010.

[13] 권호철, 김주성, 이창건, 하은용, "안드로이드 모바일 플랫폼에서의 사용자 인터페이스 자동 전환 기술의 개발," 한국 정보과학회 학술 발표 논문집, Vol.36, No.2(B), pp.436-440, 2009.

[14] 황선명, 윤석진, "이미지 플로우 기반의 모바일 GUI 테스트 도구에 관한 연구," 정보처리학회논문지D, Vol.15-D, pp. 347-354, 2008.

[15] 황선명, 김정종, "시나리오기반의 모바일 어플리케이션 소프트웨어 GUI 테스트 방법," 한국산학기술학회논문지, Vol.9, pp.681-689, 2008.

[16] Android Testing Framework, http://developer.android.com

[17] JUnit, http://www.junit.org/

마 영 철

e-mail : randy0725@gmail.com

2008년 중국 빈주 학원 컴퓨터공학과(학사)

2011년 동국대학교 컴퓨터공학과(공학석사)

2011년~현 재 비원플러스 연구원

관심분야 : 소프트웨어 테스트, 모발 보안

최 은 만

e-mail : emchoi@dgu.ac.kr

1982년 동국대학교 전산학과(학사)

1985년 한국과학기술원 전산학과(공학석사)

1993년 일리노이 공대 전산학과(공학박사)

1985년~1988년 한국표준과학연구원 연구원

1988년~1989년 데이콤 주임연구원

2002년 카네기멜론대학 소프트웨어공학과정 연수

2000, 2007년 콜로라도 주립대 전산학과 방문교수

1993년~현 재 동국대학교 컴퓨터공학과 교수

관심분야 : 소프트웨어 설계, 소프트웨어 테스트, 프로세스와 메트릭, Program Comprehension, AOP