안드로이드 어플리케이션의 화이트박스 테스트를 위한 액티비티기반의 상태도 생성기법

백태산, Ajay Kumar Jha, 이우진
경북대학교 전자전기컴퓨터학부
e-mail : sans79@gmail.com

Activity based state diagram generation for White-box testing of Android applications

Tae San Baek, Ajay Kumar Jha, Woo Jin Lee
School of EECS, Kyungpook National University

요 약

본 논문에서는 안드로이드 어플리케이션의 화이트박스 테스트를 위해 안드로이드의 특성인 액티비티의 라이프사이클을 고려하여 상태도를 생성하는 기법에 대한 연구이다. 상태도를 생성 하기 위해 소스코드에서 역공학을 통해 생성된 호출그래프에 어플리케이션내의 여러 액티비티의 라이프사이클정보를 추가하여 액티비티 상태도를 생성한다.

1. 서론

안드로이드를 기반으로 하는 스마트폰의 보급률이 상승하고 안드로이드 OS 활용범위가 사회 전반으로 넓어지고 있는 추세이다. 다양한 기기에 적용이 되고 이와 함께 다양한 기능을 하는 안드로이드 어플리케이션이 하루가 다르게 출시되고 있다.그러나 어플리케이션에서 발생한 문제점은 안드로이드 OS 에 영향을 줄 수 있으므로 개발자가 개발단계에서 미리 자체 테스트를 통하여 잠재된 문제점을 발견하여 품질 향상을 위해 소프트웨어 테스팅 도구에 대한 연구 개발이 필요하다. 안드로이드 어플리케이션은 기존의 자바 어플리케이션과는 다른 구조를 가지고 있으며 모바일 디바이스와 같이 제한된 하드웨어 환경에서 동작하도록 최적화 되어 있다.

본 논문에서는 안드로이드 어플리케이션을 역공학을 통해 생성된 정보에 안드로이드 액티비티(Activity)정보를 고려하여 테스트 케이스를 생성하기 위한 액티비티기반의 상태도 생성 기법에 대하여 알아볼 것이다. 본 논문은 다음과 같이 구성된다. 2 절에서는 안드로이드 테스팅에 대한 기존 방법과 안드로이드의 특징에 대해 기술하고, 3 절에서는 안드로이드 액티비티 정보를 기반으로 하여 테스트 케이스 및 테스트 시퀀스를 생성하는 방법을 제시한다. 마지막으로 4 절에서는 결론을 맺는다.

2. 관련연구

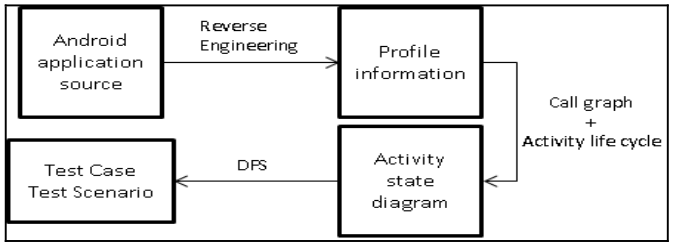
안드로이드는 테스팅을 위해 JUnit[1], Monkey[2], MonkeyRunner[3]의 세가지를 기본적으로 제공하고 있다. 안드로이드의 단위 테스트를 위해서는 JUnit 을 지원한다. 다만 기존 JVM 에서 수행하던 JUnit 을 수행하는 것은 타겟에서 수행하는것에 대한 보장을 하지 못하므로, Android Test Case Class 를 별도로 제공한다. 또한, 단순한 API 를 테스트하는 것 외에 Instrumentation based testing 기법을 제공한다. 사용자의 행위에 대한 내용을 코드로 표현하여 기능 테스트를 한다. Monkey 는 안드로이드의 무작위 이벤트를 타겟에 보내 에러가 발생하는지를 또는 어느 정도의 시간을 버티는지를 테스트하는 스트레스 테스트를 할 수 있다. MonkeyRunner 는 액션 좌표값과 액션 이벤트를 Jython 으로 스크립트형태의 코드를 제작하여 기능 테스트를 할 수 있는 도구이다.

안드로이드 소프트웨어는 4 개의 컴포넌트로 구성되어 있다. 4 개의 컴포넌트는 액티비티, 서비스, Broadcast Receiver, Content Provider 이다. 안드로이드는 메인함수와 같은 유일한 진입점이 따로 있는 것이 아니며 처음으로 생성되는 인스턴스 생성자가 실질적인 진입점 역할을 한다. 컴포넌트 중 본 논문에서 사용되는 액티비티는 안드로이드 어플리케이션을 구성하는 가장 기본적인 빌딩블록으로 한 화면을 차지하면서 뷰로 구성된 유저 인터페이스를 화면에 표시하고 사용자의 입력을 처리하는 역할을 수행하고 하나의 어플리케이션은 여러 개의 다른 화면을 구성하는 서로 다른 액티비티로 구성이 되어 있다.

기존에 연구된 모바일 소프트웨어 테스팅 방법들로는 위에서 언급한 액티비티라는 특성을 가진 안드로이드 어플리케이션에 대한 테스팅을 할 수 없다[7].

3. 안드로이드 액티비티기반의 상태도 생성

본 논문에서는 테스트케이스 생성을 위해, 정적분석기를 이용하여 역공학 기법으로 단위 프로그램(함수, 프로시저, 액티비티 등)들 사이의 호출관계를 나타내는 호출그래프(call graph)를 우선 생성한다. 이 그래프의 노드는 단위 프로그램을, 간선은 단위 프로그램 사이의 호출을 의미한다[4].

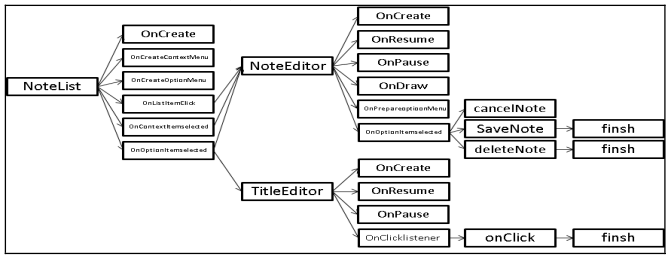


(그림 1) 안드로이드 테스트 케이스 생성 절차

액티비티 기반의 테스트 케이스를 생성하기 위해 먼저, 소스코드에서 역공학을 통해 액티비티 기반의 호출 그래프를 생성하고 호출 그래프와 액티비티 라이프사이클 정보를 추가하여 액티비티 상태도를 생성한다. 그리고 액티비티 상태도에서 깊이 우선 탐색으로 테스트 케이스를 생성할 수 있다(그림 1). 본 논문에서는 안드로이드 SDK 에 포함되어 있는 샘플코드 중 메모장 어플리케이션에 대한 테스트케이스 생성을 위한 액티비티 기반의 상태도 생성기법에 대해서만 다룬다.

3.1 액티비티 기반의 호출 그래프 생성

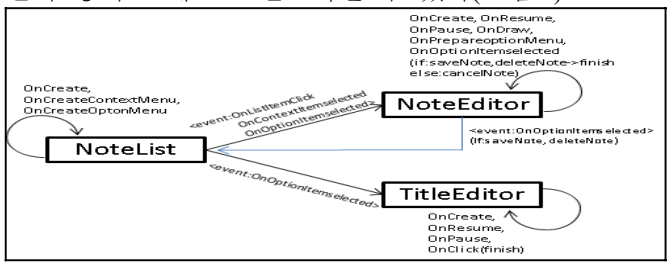
호출그래프 생성은 이미 많은 연구가 이루어져 있고 다양한 도구들이 존재하고 있다. 이러한 기술을 이용하여 안드로이드 메모장 샘플 소스코드에서 호출 그래프를 그림 2 와 같이 생성한다.



(그림 2) 호출 그래프

3.2 액티비티 기반의 호출 그래프 생성

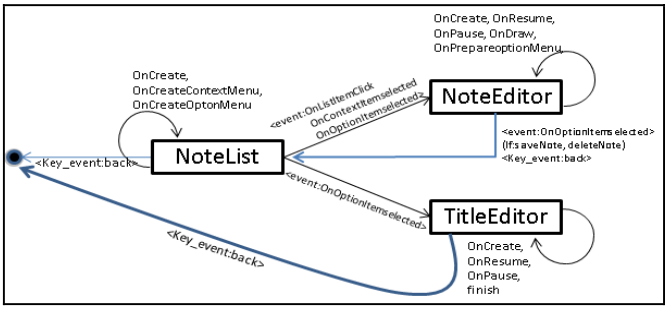
이렇게 생성된 호출 그래프에서 하나의 액티비티가 다른 액티비티와 연결시키는 노드를 두 액티비티 사이의 간선으로 연결하고 그 외의 노드들은 해당 액티비티의 사이클(cycle) 간선으로 연결하여 액티비티 기반의 상태 그래프로 간소화할 수 있다(그림 3).



(그림 3) 간소화된 상태도

3.3 액티비티 기반의 호출 그래프 생성

간소화된 그래프는 액티비티의 라이프사이클 특성[5]이 포함되어 있지 않다. 각각의 액티비티들은 서로 다른 라이프사이클을 가지고 있고 이에 따라 이전 액티비티로 이동하는 순서가 달라지게 된다. 이런 액티비티의 라이프사이클 정보를 추가하면 시작 액티비티도 라이프사이클이 존재하기 때문에 이전 상태가 존재하여야 하는 중요한 문제점이 발생하게 된다.



(그림 4) 액티비티 상태도

이를 위해 간소화된 상태도(그림 3)에 소스코드에서 구현되어 있지 않은 back_Key 이벤트에 따른 이전 액티비티로의 이동 간선을 추가한다. 어플리케이션의 첫번째 액티비티 또는 다른 액티비티에서 이전상태로 이동시 어플리케이션 실행 이전 상태로 돌아가기 위한 임의의 초기상태를 추가하여 안드로이드의 특성을 고려한 액티비티 상태도(그림 4)를 완성할 수 있다. 생성된 안드로이드의 액티비티 라이프사이클을 고려한 액티비티 상태도를 선형 그래프로 변환하여 깊이 우선 탐색방법[6]으로 테스트케이스를 생성하여 상태기반 화이트박스 테스트에 활용할 수 있다.

4. 결론

본 논문에서는 안드로이드 어플리케이션의 기본적인 빌딩블록인 액티비티의 라이프사이클을 고려하여 액티비티 상태도를 생성하는 방법에 대하여 제안하였다. 이러한 방법으로 생성된 상태도에서 테스트케이스를 생성하여 안드로이드 어플리케이션에 대한 상태기반 화이트박스 테스팅을 할 수 있다.

ACKNOWLEDGEMENT

"이 논문은 2010 년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No.2010-0010606)."

참고문헌

[1] JUnit, http://www.junit.org/taxonomy/term/6

[2] Monkey, http://developer.android.com/guide/developing/tools/monkey.html

[3] Monkey UI/Application Excerceiser, http://developer.android.com/guide/developing/tools/monkey.html, May 2010

[4] D Grove, G DeFouw, J Dean, C Chambers, "Call graph construction in object-oriented languages," ACM SIGPLAN Notices, 1997

[5] Android activity lifecycle,"http://developer.android.com/reference/android/app/Activity.html#ActivityLifecycle," May 2010

[6] R Tarjan, "Depth-first search and linear graph algorithms," 12th Annual Symposium on switching and automata theory, 1971

[7] Marek Janicki, Mika Katara, Tuula Paakkonen, "Obstacles and opportunities in deploying model-based GUI testing of mobile software : a survey," Softw. Test. Verif. Reliab published online in Wiley Online Library, May 2011