ISSN 2383-630X(Print) / ISSN 2383-6296(Online)
Journal of KIISE, Vol. 42, No. 8, pp. 1022-1030, 2015. 8
http://dx.doi.org/10.5626/JOK.2015.42.8.1022

안드로이드 어플리케이션 개발에서 퍼미션 분석을 사용한 다양한 테스트 환경 조건 생성 기법

(Testing Android Applications Considering Various Contexts Inferred from Permissions)

송 광 식 † (Kwangsik Song)　한 아 림 †† (Ah-Rim Han)　정 세 훈 † (Sehun Jeong)　차 성 덕 ††† (Sungdeok Cha)

요 약 최근에 제조되는 스마트폰들이 점점 다양한 인터페이스 장비와 사용자 주변 환경을 인식하는 센서 장비를 포함하게 되면서, 어플리케이션을 개발 시 주어진 장비들을 활용함으로 인해 증가된 개발 및 테스팅 복잡도를 효과적으로 제어하는 것이 중요하게 되었다. 이는 기존의 이벤트 기반 혹은 사용자가 지정한 입력만을 사용하여 테스트를 수행하는 연구로는 한계가 있음을 의미하며 대상 어플리케이션과 상호작용하는 주변 기기를 특정하여 이들로부터 구체적인 실행 환경 조합을 체계적으로 생성하는 연구가 필요하고 할 수 있다. 본 연구에서는 안드로이드 어플리케이션과 함께 배포되는 퍼미션 정보를 기반으로 어플리케이션의 기능에 영향 주는 주변 기기들을 분석한 후 이를 기반으로 외부 환경 조건들을 자동으로 생성하는 방법을 제안한다. 마지막으로 오픈소스에 본 연구기법을 적용하여 코드 커버리지가 향상됨을 보임으로써 본 연구의 효과성을 검증하였다.

키워드: 안드로이드 어플리케이션 테스팅, 퍼미션, 상황 인지, 다양한 환경 조건

Abstract The context-awareness of mobile applications yields several issues for testing, since mobile applications should be able to be tested in any environment and under any contextual input. In previous studies of testing for Android applications as an event-driven system, many researchers have focused on using generated test cases considering only Graphical User Interface (GUI) events. However, it is difficult to find failures that could be detected when considering the changes in the context in which applications run. It is even more important to consider various contexts since the mobile applications adapt and use the new features and sensors of mobile devices. In this paper, we provide a method of systematically generating various executing contexts from permissions. By

· 이 논문은 2015년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. NRF-2014R1A1A2054098)
· 본 연구는 미래창조과학부 및 정보통신기술진흥센터의 대학ICT연구센터육성지원사업의 연구결과로 수행되었음(IITP-2015-H8501-15-1012)
· 이 논문은 2015 한국 소프트웨어공학 학술대회에서 '안드로이드 어플리케이션 개발에서 퍼미션 분석을 사용한 테스트 케이스의 다양한 테스트 환경 조건 생성 기법'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 고려대학교 전산학과
kwangsik_song@korea.ac.kr
gifaranga@korea.ac.kr
†† 정 회 원 : 고려대학교 전산학과 연구교수(Korea Univ.)
arhan@korea.ac.kr
(Corresponding author임)
††† 종신회원 : 고려대학교 전산학과 교수
scha@korea.ac.kr

논문접수 : 2015년 3월 17일
(Received 17 March 2015)
논문수정 : 2015년 6월 1일
(Revised 1 June 2015)
심사완료 : 2015년 6월 11일
(Accepted 11 June 2015)

정보과학회논문지 제42권 제8호(2015. 8)

referring to the lists of permissions, the resources used by the applications for running Android applications can be easily inferred. To evaluate the efficiency of our testing method, we applied the method on two open source projects and showed that it contributes to improve the statement code coverage.

Keywords: Android application testing, permissions, context-awareness, various contexts

1. 서 론

최근 스마트폰 어플리케이션들 중에는 사용자 주변 환경을 파악 및 이용하여 과거에는 지원하지 못했던 외부 장치를 활용한 기능들을 제공하고 있는 사례가 늘고 있다. 이러한 경향은 스마트폰 운영체제의 발전과 함께 하드웨어 제조업체들이 풍부한 주변기기(입/출력, 통신 및 센서 하드웨어)를 경쟁적으로 제공하게 되면서 시작되었고, 최근 웨어러블 디바이스(wearable device)의 등장으로 인해 그 경향은 더욱 확대 될 전망이다. 이러한 상황 인지(context-aware) 어플리케이션[1,2] 개발 환경에서는 개발자들이 다양한 기능을 제공하는 어플리케이션을 제작할 수 있다는 장점이 있는 반면, 어플리케이션에 대하여 기본적인 사용자와의 상호작용 이외에도 다양한 환경 및 스마트폰 하드웨어 구성의 조건에 대하여 테스팅을 해야 하는 어려움이 있다[3].

기존에 모바일 어플리케이션 테스팅(mobile application testing)에 널리 사용되는 GUI(Graphical User Interface)이벤트 기반 테스팅에 대한 연구들[4-8]에서는 실제 어플리케이션을 대상 하드웨어에 탑재한 후 시스템 테스팅을 수행할 수 있고 자동화가 용이하다는 장점이 있지만, 상황 인지형 어플리케이션(context-aware application)이 대응해야 할 각종 환경 조건은 효과적으로 생성 하지 못한다는 한계점이 있다. 또 다른 스마트폰 어플리케이션 테스팅 방법은 전문가가 어플리케이션의 사용 형태(usage profile)를 분석하여 어플리케이션 결함을 발견하는데 용이하다고 생각되는 실행 환경 조건을 선정해 수동으로 적용하는 연구가 있다[9,10]. 그러나 이러한 방법들은 체계적인 적용이 어렵고, 특히 고려해야 할 환경 조건이 많은 경우에는 어떤 환경 조건을 먼저 적용하는 것이 효율적인지 대한 답을 주지 못한다는 단점이 있다. 따라서 각 어플리케이션에 영향을 줄 수 있는 환경 요인을 분석하여 다양한 실행 환경 조건을 자동으로 생성해 주는 기법이 필요하다.

본 연구에서는 안드로이드[11] 어플리케이션의 퍼미션(permission) 분석한 정보를 이용하여 체계적으로 다양한 실행 환경(executing context) 조건을 고려한 안드로이드 어플리케이션 테스팅(testing) 방법을 제안한다. 퍼미션이란 각각의 안드로이드 어플리케이션들이 접근할 수 있는 주변기기들에 대하여 정의된 선언으로, 어플리케이션은 퍼미션에 정의된 주변기기들에만 API를 통하여 접근할 수 있으므로[12,13] 어플리케이션의 기능이나 사용 형태에 대한 사전 지식 없이도 그와 관련 있는 환경 조건을 유추할 수 있는 근거를 제공한다. 게다가 퍼미션 정보는 안드로이드 어플리케이션 패키지 바이너리 파일과 함께 별도의 XML(eXtension Markup Language) 파일 형태로 배포되므로 분석이 용이하다는 장점도 있다. 본 논문에서는 퍼미션 분석을 통해 파악된 어플리케이션과 상호작용하는 디바이스 정보를 활용해 가능한 상태 조합, 즉 외부 환경으로부터 들어오는 이벤트들이 디바이스 센서(sensor)들에 의하여 감지되고 이에 따라서 변경된 디바이스의 가능한 상태 조건들을 체계적으로 구축하는 방법을 제안한다. 이는 궁극적으로는 다양하게 실행 환경 조건을 변경해가면서 GUI 이벤트 기반으로 자동으로 생성된 테스트케이스들을 실행하여 다양한 환경 변화에 대한 테스팅(testing)을 수행할 수 있도록 한다.

본 연구에서 제안하는 환경 조건 생성 절차는 다음과 같다.

a. 퍼미션 정보로부터 구체적인 실행 환경을 만들기 위해 각 퍼미션에 수록될 수 있는 주변기기 종류별로 기기들이 가질 수 있는 상태 후보 군을 미리 정의한다.
b. 이 정보를 이용하여 테스팅 대상 어플리케이션의 퍼미션 정보에 따라 각 주변기기들이 가질 수 있는 상태를 후보군에서 뽑아 조합하여 환경 조건을 생성한다.
c. 마지막으로 본 연구에서 만들어진 환경 조건과 어플리케이션과 사용자간의 상호작용을 기반으로 하는 기존의 어플리케이션 GUI 테스팅 기법을 활용하여[6] 상황 인지 테스트 케이스 집합을 생성한다.

본 연구에서는 결함 발견 확률을 높이기 위해 개발자의 경험이나 주변기기의 특징에 따라 각 환경 조건 조합의 우선순위를 매기는 방법도 제안하였다.

본 연구에서 제안하는 방법의 효과성(effectiveness)을 보이기 위하여, 오픈 소스 안드로이드 어플리케이션 2종을 대상으로 코드 커버리지(code coverage)를 측정하는 실험을 수행하였다. 코드 커버리지 같은 경우는 직접적으로 환경 조건을 고려한 효과성을 나타내는 지표는 아니지만 높은 코드 커버리지가 결함 탐지에 긍정적인 영향을 미치므로[14] 이 지표를 통해 간접적으로 성능의 향상을 보여줄 수 있다. 실험 수행 결과, 기존 GUI 이벤트 기반 테스팅 기법 만 적용했을 때 보다 퍼

미션을 분석한 결과를 함께 적용하였을 때 더 높은 코드 커버리지를 달성함을 보일 수 있었다.

본 논문의 구성은 다음과 같다. 2장에서는 외부 환경을 고려한 이벤트 기반의 모바일 어플리케이션 테스팅 연구에 대하여 설명한다. 3장에서는 안드로이드 퍼미션의 역할과 필요성에 대해 설명한다. 4장에서는 퍼미션 분석을 통해 테스트 환경 조건들을 만들 수 있는 방법을 자세히 설명한다. 5장에서는 제안한 방법의 효과성을 보이기 위한 실험 과정 및 결과를 기술한다. 마지막으로, 6장에서는 결론과 향후에 대해 논의한다.

2. 관련 연구

사용자로부터 주어지는 이벤트를 중심으로한 GUI 이벤트 기반 테스팅 연구[4-8]는 예전부터 있어왔다. 하지만, 모바일 어플리케이션은 상황 인지형 어플리케이션으로써 어플리케이션이 실행되면서 외부에서 들어오는 이벤트를 감지(sense)하고 반응(react)하는 등 주위 환경과도 많은 상호작용(interaction)을 하기 때문에 이러한 특성을 고려하여 테스팅하는 것이 중요한데 이를 고려한 연구는 많지 않은 실정이다.

외부 환경 이벤트를 고려하는 중 가장 최근 연구로 D. Amalfitano의 연구[9]를 들 수 있는데, 이 논문에서는 모바일 어플리케이션의 결함 발현을 유도하기 위한 외부 환경 이벤트 시나리오를 제안하고 이를 기존 GUI 이벤트 기반 테스트 케이스 자동 생성 도구[6]에 구현하였다. 개선된 테스트 케이스 자동 생성 도구는 우선 대상 어플리케이션의 GUI를 분석하여 사용자 입력 테스트 케이스를 자동 생성한 뒤 이를 다양한 외부 환경 시나리오에 따라 설정된 안드로이드 가상 디바이스 상에서 수행한다. 이 연구는 모바일 어플리케이션의 외부 환경 변화를 테스팅에 고려했다는 점에서 본 논문의 목적과 유사하다고 할 수 있다. 하지만 이 논문은 외부 환경 이벤트 시나리오를 전문가가 경험과 직관을 바탕으로 수동 생성하는 반면, 본 논문에서 소개하는 방법은 어플리케이션마다 함께 배포되는 퍼미션 정보를 분석하여 체계적으로 외부 환경 조건을 생성한다는 점에서 차이가 있다. 또한 이 연구는 안드로이드 어플리케이션의 실행과 관련된 외부 환경 조건을 체계적으로 분석하기 위해서 소스코드에 선언되어 있는 인텐트 이벤트 핸들러(intent event handler)를 분석하여 해당 어플리케이션이 반응하는 외부 환경을 유추하고 이를 바탕으로 테스트 케이스를 생성하였다. 소스 코드를 분석하는 이러한 방법은 어플리케이션이 어떤 외부 이벤트에 반응하는지에 대한 명확한 정보를 얻을 수 있다는 장점이 있으나 소스코드가 꼭 필요하다는 점에서 적용 대상이 제한된다. 이에 반해 본 연구에서 제안하는 방법은 소스코드와 상관없이 어플리케이션과 배포되는 퍼미션 XML을 분석하여 외부 환경조건을 체계적으로 생성한다는 점에서 차별 점이 있다.

3. 안드로이드 퍼미션

안드로이드 운영체제의 보안 시스템을 구성하는 퍼미션은 특정한 API(Application Program Interface) 호출이나 주변기기에 접근하는 권한을 관리한다[15,16]. 안드로이드 어플리케이션 개발을 위해 선언할 수 있는 퍼미션의 종류는 Wi-Fi, Bluetooth, 오디오 등의 스마트폰에 내장된 기기에 대한 항목에서부터 위치 정보, 주소록, 통화목록 접근 등 개인정보와 관련된 부분까지 다양하게 준비되어 있으며 개발자는 어플리케이션이 제 기능을 수행하는데 필요한 주변기기나 개인 정보의 종류 등에 근거하여 필요한 퍼미션을 안드로이드 Manifest XML 파일에 선언한 후 어플리케이션 바이너리 파일과 묶어 배포한다. 이렇게 선언된 퍼미션 목록은 어플리케이션이 사용자의 하드웨어에 설치될 때 그림 1과 같은 사용자 승인 과정을 통과하게 되며 이 과정을 통해 사용자는 자신이 설치하려고 하고 있는 어플리케이션이 하드웨어나 개인 정보에 어떤 영향을 미칠 수 있는지 평가하여 최종적으로 설치 여부를 결정한다. 따라서 퍼미션 분석을 통해 모바일 어플리케이션이 사용하는 주변 기기들(devices)에 대하여 쉽게 유추할 수 있다.

본 논문에서는 대상 어플리케이션의 환경 조건에 영향을 미치는 주변 기기들로 현대 스마트폰에 탑재되는 다양한 센서(예. WIFI, GPS)와 기타 장비(예. 배터리, USB, SD-card)들을 선정하고 이들과 관련된 퍼미션을 중점적으로 분석하였다.

4. 퍼미션을 이용한 환경 조건 생성에 기반한 안드로이드 어플리케이션 테스팅 방법

이 장에서는 그림 2의 순서에 따라 안드로이드 퍼미션들을 분석하고 그 결과를 이용해 테스팅에 활용할 수 있는 환경 조건을 자동으로 도출하는 방법을 설명한다. 우선 안드로이드 퍼미션을 분석하여 각 퍼미션이 상징하는 주변 환경의 상태 후보 군을 정의하였다. 그 다음 상태 후보 군을 조합하여 구체적인 환경 조건을 생성함과 동시에 각 환경 조건을 평가하여 결함 발현 가능성이 높으리라 기대되는 순서대로 우선 순위를 할당한다.

4.1 안드로이드 퍼미션 분석

안드로이드 퍼미션을 통해서 실행 환경 조건을 생성하기에 앞서, 퍼미션과 관련된 외부 환경, 즉 각 퍼미션이 관장하는 주변기기의 종류와 각 주변기기가 어플리케이션에 전달할 수 있는 상태 값을 분석하였다. 이 분석은 *BLUETOOTH* 퍼미션과 같이 그 이름에서 관련

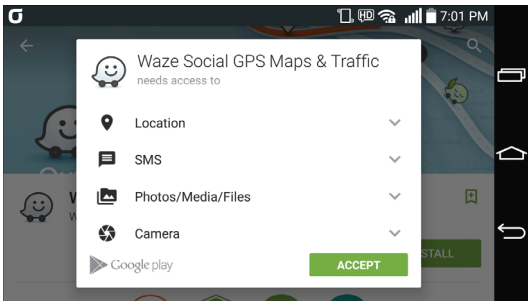

그림 1 안드로이드 어플리케이션 설치 시 사용자로부터 퍼미션을 받는 화면

Fig. 1 Snapshot of acquiring permission(s) from a user when installing an application

하드웨어를 쉽게 유추할 수 있는 사례도 있지만 *ACCESS_FINE_LOCATION* 퍼미션과 같이 추상적인 사례도 있다. 이 퍼미션은 스마트폰의 상세 위치 정보에 접근하기 위한 퍼미션으로써 지도나 사용자의 위치를 활용하는 어플리케이션에 흔히 사용되며, Wi-Fi, GPS, Radio 3가지 주변기기에 제한적으로 접근한다.

표 1은 이와 같은 분석을 통해 구체화 한 퍼미션 중 일부를 요약한 것이다. 표의 처음 두 열은 퍼미션의 이름과 역할을 기술한 것이고 세 번째 열에서 관련된 주변기기 및 그 주변기기들이 가질 수 있는 상태 후보 군을 제시하였다. 대부분의 주변기기 상태는 ON, OFF 와 같이 기초적인 가용성 여부로 표현할 수 있으며 이를 통해 퍼미션을 활용하는 어플리케이션이 그와 관련된 주변기기들의 모든 가능한 가용 상태 조합에 대해서 정상적인 작동을 하는지 여부를 확인하는 환경 조건을 생

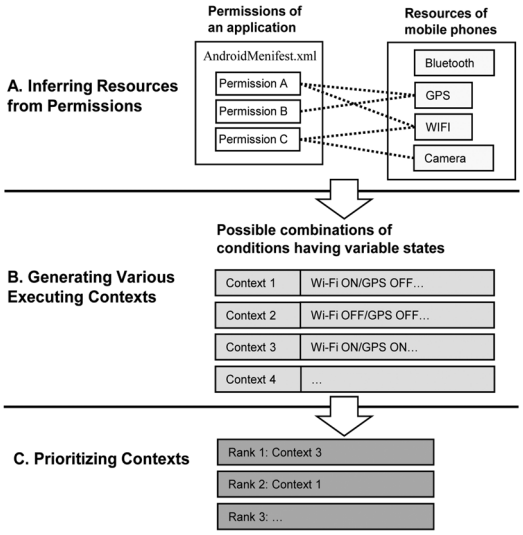

그림 2 퍼미션을 이용한 다양한 환경 조건 생성 방법 개요

Fig. 2 Overview for generating various contexts after analyzing permissions

성할 수 있다. 퍼미션에 따라서 상태 후보 군은 더 세분화 될 수도 있다. 예를 들어 GPS는 ON, OFF 뿐만 아니라 Fail, 즉 켜져는 있지만 위치 정보를 가져오는데 실패한 경우도 상태 후보에 추가할 수 있다. 마지막으로 표의 버전 열은 각 퍼미션이 필요로 하는 안드로이드 운영체제의 최소 버전 정보이다.

4.2 환경 조건 조합 및 우선 순위화

앞서 분석한 퍼미션-주변기기 관계 및 각 주변기기가 가질 수 있는 상태 후보들을 이용하여 대상 어플리케이

표 1 퍼미션 별로 관련된 주변기기와 각 상태 후보들 예시

Table 1 List of permissions and related devices with their possible states

Permission	Description	Related Devices [Possible States]	Version
ACCESS_FINE_LOCATION	Measures accurate location	Wi-Fi[on|off], GPS[on|off], Radio[on|off]	Android 1.0 ~
INTERNET	Accesses Internet	Wi-Fi[on|off], Radio[on|off]	Android 1.0 ~
CAMERA	Accesses camera	Camera Enable, [Flash on|off], SD-card [free|full]	Android 1.0 ~
BLUETOOTH	Enables Bluetooth	Bluetooth [on|off]	Android 1.0 ~
WRITE_CALL_LOG	Modifies user's call log	Radio [on|off]	~ Android 4.0.3
WRITE_EXTERNAL_STORAGE	Modifies a SD-card	SD-card [free|full]	~ Android 1.5
BIND_DEVICE_ADMIN	Controls user's device	Camera Enable, [Flash on|off], SD-card [free|full], Wi-Fi[on|off]	Android 2.2.x ~
VIBRATE	Accesses a vibration motor	Vibrator[on|off]	Android 1.0 ~
NFC	Enables NFC feature	NFC [on|off]	Android 2.3 ~
FLASHLIGHT	Enables a flash light	Flash [on|off]	Android 1.0 ~
CHANGE_NETWORK_STATE	Modifies networking state	Wi-Fi[on|off], GPS[on|off], Radio[on|off]	Android 1.0 ~
CAPTURE_VIDEO_OUTPUT	Enables screen capturing	LCD [on|off], Camera Enable	Android 1.0 ~

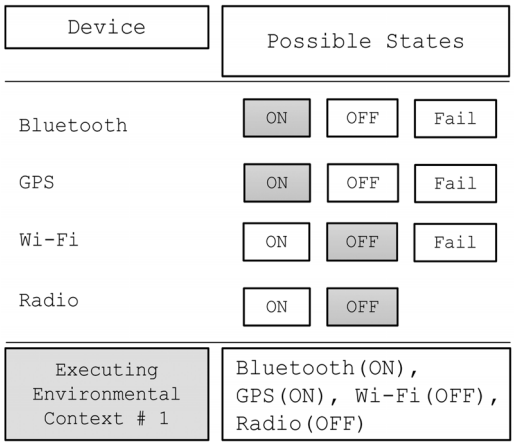

그림 3 환경 조건 조합 생성 예제

Fig. 3 Example of generating an executing context-combination of conditions having variable states

션의 퍼미션 정보에 맞는 환경 조건을 생성할 수 있다. 본 연구에서는 퍼미션을 통해 유추된 주변기기들이 가질 수 있는 모든 상태 조합에 대하여 테스트 실행 환경을 생성하였다. 예를 들어 그림 3의 경우는 퍼미션 분석을 통해 해당 어플리케이션의 실행에 영향을 줄 수 있는 주변기기로 Bluetooth, GPS, Wi-Fi, Radio를 도출해 낸 경우이다. 최종적인 환경 조건 후보 군은 각 주변기기의 상태 조건 가능성을 모두 조합하여 생성되며 해당 예제의 경우에는 최대 54가지 환경 조건 후보군을 생성하였다.

이러한 조합법은 구현과 자동화가 간편하고 그 과정을 이해하기 쉽다는 장점이 있으나 퍼미션 조합에 따라서 환경 조건 후보 군의 크기가 빠르게 커질 수 있다는 단점 또한 있다. 그러므로 어떤 환경 조건 후보를 우선적으로 테스팅에 적용해야 하는지를 결정하는 문제가 중요하다. 게다가 환경 조건 후보들은 실행 환경 조건 설정 값을 나타내며 이는 단독으로 테스트 케이스로써 사용 될 수 없다.

본 논문에서는 중요한 환경 조건들을 구분하고 실행 환경 조건들을 우선순위화 하기 위하여 2가지 기준을 제안한다. 첫 번째 기준은 각 주변 기기의 상태가 실 사용 환경에서 얼마나 쉽고 자주 바뀔 수 있는지에 착안한 것이다. 예를 들어 GPS와 Wi-Fi의 경우에는 사용자의 위치나 실내/외 여부에 따라 그 상태가 자주 바뀌므로 이 주변기기를 활용하는 어플리케이션이 이 변화에 얼마나 잘 대응하는지 여부가 중요하다. 이와 상대적으로 SD-card나 센서(sensor)는 하드웨어에 큰 손상이 없는 한 항상 접근할 수 있는 상태이다. 이를 바탕으로 표 2와 같이 각 주변기기 별 가중치를 정하여 가중치가 높은 주변기기의 상태 후보들을 먼저 테스트 케이스로 적용하게끔 환경 조건 조합의 우선순위를 결정할 수 있다.

표 2 주변 기기 별 가중치 설정 예제

Table 2 Weights per devices related to permissions

Device	Weight
GPS	User modifiable (3)
Wi-Fi	User modifiable (3)
Radio	User modifiable (3)
SD-card	User unmodifiable (2)
Sensor	Read only (1)

두 번째 기준은 각 주변기기 상태 후보 군의 의미에 따라 가중치를 부여하는 방법이다. 예를 들어, 표 3은 각 퍼미션 별 정상 시나리오(success scenario) 수행 가능 여부에 따라 각 상태 후보군의 가중치를 할당하였다. 즉, 정상적인 모든 설정을 확인하는 것이 가중치가 높고, 그렇지 않은 것은 가중치를 낮게 할당하여 구분하였다. 이와 같은 활용 예는 모든 기능에 대하여 정상 수행 가능한 환경에서 먼저 테스트 하고 난 뒤에 예외사항 발생이 가능한 환경에서 테스트를 수행하려고 할 때 유용하다.

환경 조건의 최종 우선순위 값은 각 주변기기 별로 가중치와(표 2 참조) 사용 빈도에 따른 가중치(표 3 참조) 가운데서 테스팅 목적에 따라 선택적으로 사용이 가능하다. 예를 들어, 세부 위치 정보와 저장장치에 대한 퍼미션을 필요로 하는 어플리케이션의 환경 조건들의 가중치를 계산할 때, 첫 번째 방법으로는 GPS, Wi-Fi, Radio는 조작 가능하므로 3점, SD-card는 조작이 어려우므로 2점이 부여되므로 앞선 세 주변기기의 상태 조합이 먼저 검사된다. 반면에, 두 번째 기준을 적용하면 사전 정의된 값에 따라 상태 후보 조건 별로 3~5점이 부여된다. 이렇게 계산하여 나온 결과 값은 높을수록 우선 순위가 높은 환경 조건이다.

표 3 상태 후보 군의 사용 빈도에 따른 가중치 설정 예제

Table 3 Weights of executing contexts according to used scenarios

Permission	Weight	Device[State]
ACCESS_FINE_LOCATION	3	GPS[on] Wi-Fi[on|off] Radio[on|off]
	2	GPS[off] Wi-Fi[off] Radio[on|off]
	2	GPS[off] Wi-Fi[on] Radio[on|off]
WRITE_EXTERNAL_STORAGE	2	Memory[free]
	1	Memory[full]

지금까지 본 논문에서 제안한 환경 조건을 생성하고 정렬하는 방법을 설명하였다. 이 결과는 사용자 GUI 이벤트 생성 기법 등을 활용한 테스트 케이스와 합쳐져 하나의 온전한 테스트 케이스를 이룬다. 이를 통해 기존 안드로이드 어플리케이션 테스팅에서 단순히 가정하고 있었던 어플리케이션 실행과 관련된 환경 조건들을 다양하고 체계적으로 가정함으로써 실행 환경 변화와 관련된 다양한 결함을 발견할 수 있다. 다음 5장에서는 제안한 방법의 성능을 검증하기 위해서 오픈 소스 안드로이드 어플리케이션 2종에 대해 실험한 결과를 설명한다.

5. 평가

본 논문에서 제안한 어플리케이션 실행 환경 분석기법이 실제 테스팅에 미치는 영향을 보이기 위해 오픈 소스 안드로이드 어플리케이션에 적용하였다. 비록 제안한 방법은 블랙박스(black-box) 테스팅에 활용될 것을 목표로 하였으나 본 사례연구에서는 정량적인 성능 측정을 위해 화이트박스(white-box) 테스팅에서 사용되는 문장 커버리지(statement coverage) 척도를 활용하여 기존 GUI 이벤트 기반 테스트 케이스 생성 기법을 활용했을 때보다 환경조건을 더 고려했을 때의 커버리지 변화량을 측정하였다.

5.1 실험 방법

우선 테스팅 대상이 되는 안드로이드 어플리케이션으로 Open Camera[17]와 Angulo[18]를 선정하였다. 두 어플리케이션의 특징은 표 4에 요약하였다. 이 두 어플리케이션은 스마트폰의 하드웨어 장치를 활용하고 있으며 소스 코드가 공개되어 있어서 문장 커버리지를 통한 성능 비교가 가능하였다. 또한, 안드로이드 버전은 아이스크림 샌드위치(Ice cream sandwich)-4.0.x 버전을 이용하였는데, 이는 GPS, Wi-Fi, SD-card, Camera와 같은 주변 기기에 대하여 퍼미션 정보를 기술할 수 있는 환경이다.

실험은 그림 4와 같이 수행되었다. 선정한 안드로이드 어플리케이션의 퍼미션을 분석하여 실행 환경 조건을

표 4 대상 어플리케이션 설명
Table 4 Characteristic for experimental subjects

LOC	Permission	Feature
Open camera [17]		
3790	*ACCESS_FINE_LOCATION* *WRITE_EXTERNAL_STORAGE* *CAMERA*	Camera
Angulo [18]		
621	*CAMERA* *VIBRATE*	Angle measurement tool

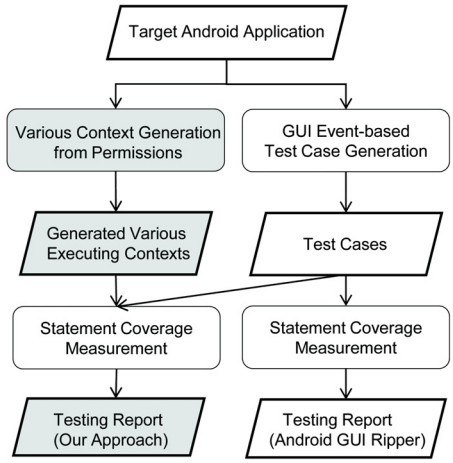

그림 4 실험 설계 개요
Fig. 4 Overview of experimental design.

표 5 실행된 테스트 케이스와 환경 조건의 수
Table 5 Generated number of test cases and executing contexts

Subject (# Test Cases from Android GUI Ripper)	Permission	# States
	# Executing Contexts	
Open camera (22)	*ACCESS_FINE_LOCATION*	8
	WRITE_EXTERNAL_STORAGE	2
	CAMERA	2
	32 (=8X2X2)	
Angulo (43)	*CAMERA*	2
	VIBRATE	2
	4 (=2X2)	

생성한 뒤 이를 표 5의 기준에 따라 정렬한 결과를 GUI 이벤트 기반 테스트 케이스 생성 도구인 Android GUI Ripper[6,8]가 생성한 테스트 케이스와 결합한 결과를 실험군으로 설정하여 문장 커버리지를 측정하였다. 대조군으로는 환경 조건 고려 없이 Android GUI Ripper 단독으로 생성한 테스트케이스를 활용하였다.

Android GUI Ripper는 Open Camera, Angulo 어플리케이션에 대해서 각각 22, 43개의 테스트 케이스를 생성하였다. 또한 본 논문에서 제안한 방법은 퍼미션을 분석하여 동일한 어플리케이션에 대하여 표 5와 같이 각각 32, 4개의 실행 환경 조건을 생성하였다.

표 6은 Open Camera에 대하여 생성된 다양한 조건의 환경 조건을 표 3의 기준에 따라 정렬한 결과이다. 생성한 실행 환경 조건들을 살펴보면, 대상 어플리케이션은 세부 위치정보, 외장 저장장치, 카메라에 대한 퍼미션을 가지고 있으므로 {Location(Wi-Fi(on), GPS (off),

표 6 Open Camera 어플리케이션에 대해서 생성 및 정렬 된 환경 조건
Table 6 Generated and ranked executing contexts for Open Camera [17]

ACCESS_FINE_LOCATION		WRITE_EXTERNAL_STORAGE		CAMERA		Total weights	Rank
Wi-Fi on, GPS on, Radio on	3	SD-card free	2	Camera on	2	7	1
Wi-Fi on, GPS off, Radio on	2	SD-card free	2	Camera on	2	6	2
Wi-Fi off, GPS on, Radio on	2	SD-card free	2	Camera on	2	6	2
Wi-Fi off, GPS off, Radio on	1	SD-card free	2	Camera on	2	5	3
Wi-Fi on, GPS on, Radio on	3	SD-card full	1	Camera on	2	6	2
Wi-Fi on, GPS on, Radio on	3	SD-card free	2	Camera off	1	6	2
Wi-Fi on, GPS on, Radio on	3	SD-card full	1	Camera off	1	5	3
WI-FI on, GPS off, Radio on	2	SD-card full	1	Camera on	2	5	3
WI-FI on, GPS off, Radio on	2	SD-card full	1	Camera off	1	4	4
WI-FI on, GPS off, Radio on	2	SD-card free	2	Camera off	1	5	3

Radio(off)), SD-card(full), Camera(on)}로 표현되는 실행 조건을 생성할 수 있고, Angulo에 대해서도 동일한 방식으로 실행 환경 조건을 생성할 수 있다.

5.2 결과

그림 5의 그래프에서 볼 수 있듯이 Open Camera에 대해서 하나의 실행 환경에서 Android GUI Ripper를 기반으로 생성한 테스트 케이스를 실행하면 45% 문장 커버리지를 보이는 반면, 퍼미션을 기반으로 32개의 실행환경에서 테스트 케이스를 실행하면 문장 커버리지를 47%로 향상시킬 수 있었다. 예를 들어 그림 6은 Open Camera 에서 본 연구에서 제안하는 기법을 사용하기 전과 후의 코드 커버리지 리포트 중 일부이다. 이 리포트에서 각 라인의 수행 여부를 나타내는 두 번째 열을 보면 알 수 있듯이, 퍼미션 분석을 통해 Camera 의 가용성 여부를 환경 조건으로 활용하여 실행 환경 조건을 다양화한 테스트케이스를 실행함으로써, Camera 상태가 off 일

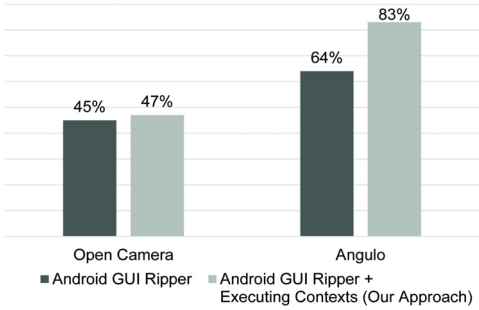

그림 5 문장 커버리지 결과
Fig. 5 Results of statement coverage

Lines	Cover	Class Preview$1RotatedTextView, Method onDraw (Canvas)
2291	✓	else if(**camera == null**) {
2296	✗	p.setColor(Color.WHITE);
2297	✗	p.setTextSize(14 * scale + 0.5f);
2298	✗	p.setTextAlign(Paint.Align.CENTER);
2299	✗	int pixels_offset = (int) (20 * scale + 0.5f);
2306	✓	}

(a) 환경 조건을 고려 전 (Before considering executing contexts)

Lines	Cover	Class Preview$1RotatedTextView, Method onDraw (Canvas)
2291	✓	else if(**camera == null**) {
2296	✓	p.setColor(Color.WHITE);
2297	✓	p.setTextSize(14 * scale + 0.5f);
2298	✓	p.setTextAlign(Paint.Align.CENTER);
2299	✓	int pixels_offset = (int) (20 * scale + 0.5f);
2306	✓	}

(b) 환경 조건을 고려 후 (After considering executing contexts)

그림 6 Open Camera[17]에서 환경 조건을 고려하여 추가적으로 수행된 코드
Fig. 6 Additionally executed source codes by considering executing contexts in Open Camera[17]

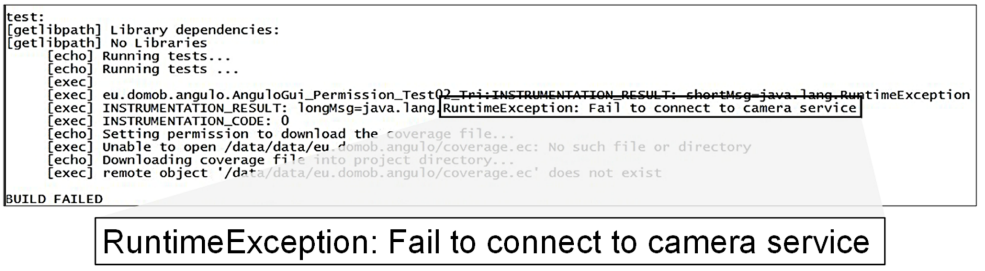

그림 7 런타임 예외상황: 카메라 서비스 연결 오류
Fig. 7 Runtime exception: fail to connect to camera service in Angulo [18]

때만 수행되는 코드를 부가적으로 커버할 수 있다.

Angulo에서도 마찬가지로 Android GUI Ripper가 생성한 테스트 케이스에 퍼미션 분석을 통한 테스트 케이스를 추가하면, 문장 커버리지를 64%에서 83%로 향상시킬 수 있었다. 그림 7은 Angulo에서 본 연구에서 제안하는 기법을 이용함으로써, 어플리케이션이 이유 없이 멈추거나 (freeze), 에러를 발생하거나 (error occurrence), 다운되는(crash) 경우를 찾을 수 있음을 보여주는 예제이다.

결과적으로, 본 연구에서 제안하는 기법을 이용하면, 디바이스 상태 조건에 따라 수행 여부가 결정되는 코드 부분에 대해 더 많이 테스트 할 수 있었고, 이는 결과적으로 코드 커버리지를 높이는데 기여하였다.

5.3 평가 위험요소(Threats to validity)

Android GUI Ripper 는 태생적으로 GUI 기반의 테스팅 방법으로, 백 엔드(backend) 및 모든 컴포넌트들을 커버하는데 한계가 있다. 본 연구에서는 이를 극복하기 위해 실제 사용이 많은 기능과 오류가 많이 발생할 것 같은 시나리오를 수작업으로 판별하여 Android GUI Ripper가 생성한 테스트케이스를 보강하였지만 이 또한 충분히 대상 어플리케이션의 기능을 충분히 테스트하기에는 부족하였다. 이는 어플리케이션 기능을 테스트하는 테스트케이스가 충분하지 못하면 환경 조건 추가로 인한 테스트 케이스 품질 향상이 드러나기 어려움을 의미한다. 다른 한편으로는, 커버리지 측면에서는 테스트케이스 자체의 품질(quality)가 자체가 좋아야 하며, 테스트케이스의 완전성이 코드 커버리지에 매우 절대적(dominant)이라는 것을 알 수 있었다.

다양한 환경 조건을 고려한 테스트 수행 방법의 효과성(effectiveness)을 보이기 위하여 본 실험에서는 코드 커버리지를 이용했다. 코드 커버리지는 결함 탐지 성능을 직접적으로 측정하는데 사용하는 지표는 아니지만, 높은 코드 커버리지가 결함 탐지에 긍정적인 영향을 주는 상관관계[14]가 있다. 그러므로 본 논문에서는 제안하는 방법의 성능을 간접적으로 보여주기 위한 지표로 이를 활용하였으며 차후에 결함 탐지 성능을 직접적으로 측정할 수 있는 다른 검증 척도(measure) 적용을 고려하고 있다. 이를 위하여 추후 향후 연구로 실제 오픈 소스의 변경 이력(revision history)를 참고하여, 실제 일어난 결함들 중 본 연구를 이용하여 찾아낼 수 있었던 결함들을 분석할 계획이다.

6. 결론 및 향후 연구 계획

본 논문에는 안드로이드 어플리케이션에서 퍼미션 분석에 근거한 체계적인 실행 환경 조건 생성 방법을 제안하였다. 이는 어플리케이션과 그를 둘러싼 실행 환경 사이의 복잡한 상호 작용을 테스팅하려고 할 때 사용자 GUI 이벤트 패턴만을 사용하는 연구는 부족한 점에서 기인한 것이다. 따라서 본 연구에서는 안드로이드 어플리케이션마다 제공되며 그 어플리케이션이 어떤 주변 기기를 사용하는지에 대한 단서를 제공하는 안드로이드 퍼미션 정보를 활용하여 그 어플리케이션 테스팅에 활용될 수 있는 환경 조건을 생성하였다. 생성된 환경 조건은 기존 GUI 이벤트 기반 테스트 케이스와 합쳐져 환경 변화를 고려한 안드로이드 어플리케이션 테스트 케이스를 작성할 수 있었다. 제안한 기법의 효과성을 보이기 위해 오픈 소스 안드로이드 어플리케이션 2종에 적용하여 코드 커버리지가 기존 GUI 이벤트 기반의 테스트케이스만 고려하여 테스팅한 결과보다 향상되었음을 확인할 수 있었다.

향후 연구는 테스트 케이스를 생성해내기 위한 퍼미션들의 정의를 확장하는 부분과 개발자를 위한 자동화 분석 툴을 만드는 것이다. 이를 통해 퍼미션 분석에 의한 환경 조건에 근거하여 놓치기 쉬운 버그들을 찾아낼 수 있는 테스트 케이스를 자동으로 생성할 수 있다. 또한 실험과 관련하여, 코드 커버리지 뿐 아니라 결함 검출 능력에 대하여도 본 연구가 기여한다는 점을 보일 계획이다.

References

[1] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a Better Understanding of Context and Context Awareness," *Handheld and Ubiquitous Computing*, Vol. 1707, pp. 304-307, Jan. 1999.

[2] M. Wang, J. Yuan, H. Miao, and G. Tan, "A Static Analysis Approach for Automatic Generating Test Cases for Web Applications," *Proc. of Computer Science and Software Engineering*, Vol. 2, pp. 751-754, Dec. 2008.

[3] A. K. Maji, K. Hao, S. Sultana, and S. Bagchi, "Characterizing Failures in Mobile OSes: A Case Study with Android and Symbian," *Proc. of the 21st IEEE International Symposium on Software Reliability Engineering*, pp. 249-258, Nov. 2010.

[4] Monkey, [Online]. Available: http://developer.android.com/tools/help/monkey.html

[5] MonkeyRunner, [Online]. Available: http://developer.android.com/tools/help/monkeyrunner_concepts.html

[6] D. Amalfitano, A.R. Fasolino, P.Tramontana, S. DeCarmine, and A. M. Memon, "Using GUI Ripping for Automated Testing of Android Applications," *Proc. of the 27th IEEE/ACM International Conference on Automated Software Engineering*, pp. 258-261, Sep. 2012.

[7] Z. Liu, X. Gao, and X. Long, "Adaptive Random Testing of Mobile Application," *2nd International Conference on Computer Engineering and Technology*, Vol. 2, pp. 297-301, Apr. 2010.

[8] D. Amalfitano, A.R. Fasolino, P. Tramontana, S. DeCarmine, and G. Imparato, "A Toolset for GUI testing of Android Applications," *Proc. of the 28th IEEE International Conference on Software Maintenance*, pp. 650-653, Sep. 2012.

[9] D. Amlfitano, A.R. Fasolino, P. Tramontana, and N. Amatucci, "Considering Context Events in Event-Based Testing of Mobile Applications," *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops*, pp. 126-133, Mar. 2013.

[10] W. Du and A.P. Mathur, "Vulnerability Testing of Software System Using Fault Injection," *Technical report, COAST, Purdue University*, pp. 1-20, Apr. 1998.

[11] Google. Android Open Source Project, http://source.android.com/, Jul. 2014.

[12] R. Johnson, Z. Wang, C. Gagnon, and A. Stavrou, "Analysis of Android Applications' Permissions," *2012 IEEE Sixth International Conference on Software Security and Reliability Companion*, pp. 45-46, Jun. 2012.

[13] X. Wei, L. Gomez, I. Neamtiu, and M. Faloutsos, "Permission Evolution in the Android Ecosystem," *Proc. of the 28th Annual Computer Security Applications Conference*, pp. 31-40, Dec. 2012.

[14] W.E. Wong, J.R. Horgan, S. London, and A.P. Mathur, "Effect of Test Set Size and Block Coverage on the Fault Detection Effectiveness," *Software Reliability Engineering*, pp. 230-238. Nov. 1994.

[15] K.W.Y. Au, Y.F. Zhou, Z. Huang, and D. Lie, "PScout: Analyzing the Android Permission Specification," *Proc. of the 2012 ACM Conference on Computer and Communications Security*, pp. 217-228, Oct. 2012.

[16] Android Developer, System Permissions, [Online]. Available: http://devel oper.android.com/intl/ko/guide/topics/security/permissions.html, Jul. 2014.

[17] Open Camera, [Online]. Available: http://opencamera.sourceforge.net

[18] Angulo, [Online]. Available: http://www.domob.eu/projects/angulo.php

송 광 식

2005년 동국대학교 컴퓨터공학과 졸업(학사). 2015년 고려대학교 컴퓨터학과 졸업(석사). 2005년~현재 LG전자 SW 공학연구소 연구원. 관심분야는 테스트 설계, 테스트 자동화, 안드로이드 테스팅

한 아 림

2004년 서강대학교 컴퓨터학과 졸업(학사). 2007년 한국과학기술원(KAIST) 전산학과 졸업(전산학 석사). 2013년 한국과학기술원(KAIST) 전산학과 졸업(전산학 박사). 2013년~현재 고려대학교 융합소프 트웨어연구소 연구교수. 관심분야는 소프트웨어 설계 품질 평가 및 개선, 리팩토링 자동화

정 세 훈

2010년 고려대학교 컴퓨터학과 졸업(학사) 2012년 고려대학교 컴퓨터학과 졸업(석사) 2012년~현재 고려대학교 컴퓨터학과 박사과정 재학중. 관심분야는 소프트웨어 시스템 안전성 검증 기술

차 성 덕

1983년 UC Irvine 전산학과 졸업(학사) 1986년 UC Irvine 전산학과 졸업(전산학 석사). 1991년 UC Irvine 전산학과 졸업(전산학 박사). 1991년~1994년 The Aerospace Corporation 기술 위원. 1994년~2008년 한국과학기술원 교수. 2009년~2012년 고려대학교 고신뢰 융합 소프트웨어공학 연구센터장. 2008년~현재 고려대학교 교수. 관심분야는 소프트웨어 신뢰성 보장 및 테스팅, 요구공학, 웹 보안 등