https://doi.org/10.5392/JKCA.2021.21.08.403

안드로이드 앱 GUI 테스트 자동화 툴 개발 방법에 관한 연구

A Study on the Development Method of Android App GUI Test Automation Tool

박세준, 김규정
숭실대학교 글로벌미디어학부

Se-jun Park(pnk25@naver.com), Kyu-jung Kim(kyu@ssu.ac.kr)

요약

모바일 앱의 수가 기하급수적으로 늘어남에 따라 앱 개발과정에서 이루어지는 테스트의 자동화가 중요해지고 있다. 앱을 출시하기까지 다양한 유형의 테스트를 통해 반복적인 검증이 이루어지는데, 다양한 유형의 테스트 중 GUI 테스트를 중심으로 본 연구는 진행되었다. 안드로이드의 GUI 테스트 자동화 툴을 개발하기 위해서 안드로이드 앱 테스트의 UI Control 16가지와 Material Design Guideline에 대해서 기초 자료를 수집하였다. 그 후 기존 GUI 테스트 자동화 툴에 대해서 스크린 캡처 테스트 기반 2가지 툴과 소스코드 분석 테스트 기반 4가지 툴 분석을 하였다. 분석을 통해 기존의 GUI 테스트 자동화 툴들은 시각적 설계, 인터페이스 사용용이성, 컴포넌트 배치에 대해서는 고려하지 않고 있다는 점을 파악하였다. 이러한 기존 툴의 미비점을 보완하고자 새로운 GUI 테스트 자동화 툴 개발 방법으로 컴포넌트 선정이나 관리 분석 그리고 컴포넌트별 소스코드 예시로 탐색 메뉴나 버튼, 아이템 그룹화나 리스트에 관한 방법을 제시하였다. 본 연구는 GUI 테스트의 새로운 개발 방향을 제시함으로써 개발사의 안정적인 앱 배포에 기여할 수 있다는 점에서 의의가 있다.

■ 중심어 : | 안드로이드 | 앱 | GUI 디자인 | 테스트 자동화 툴 | 머터리얼 디자인 가이드라인 |

Abstract

As the number of mobile apps increases exponentially, automation of tests performed in the app development process is becoming more important. Until the app is released, iterative verification is performed through various types of tests, and this study was conducted focusing on the GUI test among various types of tests. This study is meaningful in that it can contribute to the stable app distribution of the developer by suggesting the development direction of the GUI test. To develop Android's GUI test tool, I collected basic data before presenting the development method by researching Android's UI controls and Material design guideline. After that, for the existing GUI test automation tool, two tools based on screen capture test and four tools based on source code analysis test were studied. Through this, it was found that existing GUI test tools don't consider visual design, usability, and component arrangement. In order to supplement the shortcomings of existing tools, a new GUI test automation tool development method was presented based on the basic data previously studied.

■ keyword : | Android | App | GUI Design | Test Automation Tool | Material Design Guideline |

접수일자 : 2021년 04월 07일
수정일자 : 2021년 04월 28일
심사완료일 : 2021년 05월 03일
교신저자 : 박세준, e-mail : pnk25@naver.com

I. 서 론

1. 연구 목적과 필요성

데이터 분석 플랫폼인 앱애니(App Annie)에 따르면 2020년 전세계 모바일 앱 다운로드 수, 소비자 지출, MAU(월간 순수 이용자 수)가 역대 최대치를 기록했다. 구체적으로 소비자 지출은 1120억 달러를, 다운로드 수는 1300억 회를 넘어섰다. 이렇게 많은 앱이 단시간에 개발되는 상황에서는 앱 내에 오류를 포함할 확률이 높아진다. 이러한 오류를 줄이기 위해서는 다양한 유형의 테스트가 개발 과정에서 이루어져야 한다.

본 연구에서는 다양한 유형의 테스트 중 GUI 테스트에 초점을 두고 있다. 구글은 안드로이드에 대한 메뉴, 대화상자, 창 배치 등의 GUI 제약사항을 정의하는 가이드라인을 제공한다. 앱이 이러한 가이드라인을 준수하지 않는 경우 구글이 앱 배포를 중단하거나 앱을 삭제시킬 수 있기 때문에 GUI 가이드라인의 준수는 소프트웨어를 안정적으로 배포하기 위해서 매우 중요하다.

소프트웨어 테스트 자동화 관련 국내외 논문 6편[1-6]에 따르면, 가이드라인 준수 여부 확인 방식인 자동 테스트 방식은 시간과 비용이 절감되고, 높은 정확도로 다양한 테스트를 진행할 수 있다는 장점이 있어 연구가 필요함을 알 수 있다. 그럼에도 최근 국내 관련 논문은 앱에 초점을 두지 않은 연구[7][8], 기존 테스트 툴에 기댄 연구[9], 앱 GUI에 초점을 두었으나 일부 테스트 기능에 집중한 연구[10][11]라는 점에서 한계를 지닌다. 따라서 앱 GUI 테스트 전반의 자동화에 대한 연구가 필요하다.

2. 연구 방법과 범위

연구 방법은 다음과 같다. 2장에서 Henry Muccini의 연구[12]를 통해 다양한 유형의 소프트웨어 테스트와 함께 GUI 테스트에 대해서 정리한다. 그리고 안드로이드 UI Control과 가이드라인에 대해서 연구함으로써 4장의 툴 개발의 기초 자료를 수집한다. 3장에서는 기존의 GUI 테스트 자동화 툴에 대해서 검토함으로써 4장에서 개발하는 툴의 필요성을 도출한다. 4장에서는 새로운 GUI 테스트 자동화 툴 개발 방법을 제시한다. 연구 범위는 다음과 같다.

첫째, OS는 안드로이드를 범위로 정하여 연구를 진행하였다. 구글의 안드로이드는 시장 점유율이 70% 이상으로 세계에서 가장 많이 사용된다. 또한 점유율 27%로 2위인 iOS의 디바이스 종류 29개 대비 10,707개[13]로 작동 환경이 다양하다. 이러한 작동 환경의 다양성은 GUI 테스트를 어렵게 만든다. 따라서 안드로이드에서 작동하는 앱의 GUI 테스트 자동화 툴 개발 방법을 연구할 필요가 있다.

둘째, GUI 테스트는 소스코드 분석 접근 방식과 캡처 접근 방식에 기반한다. 따라서 3장에서도 스크린 캡처 방식과 소스코드 분석 테스트로 분류하여 기존 툴을 연구하였다. 기존 GUI 테스트 자동화 툴은 웹과 모바일 환경의 테스트를 지원하며 현재까지도 주기적인 업데이트가 되고 있는 툴을 기준으로 선정하였다. 스크린 캡처 테스트 툴은 Ranorex, Sikuli를 선정하였으며, 소스코드 분석 툴은 Appuim, Espresso, SelenDroid, UI Automator를 범위로 하여 연구를 진행하였다.

셋째, GUI 테스트 자동화 툴 개발 방법의 경우 개발 명세와 컴포넌트별 예시 소스코드를 제시하였다. 개발 명세는 개발 언어 선정, 컴포넌트 selector, Event & Time operators, Property 관리, 컴포넌트 분석, 서버와의 통신을 기준으로 연구하였다. 예시 소스코드는 많이 사용되는 하단 탐색 메뉴, Raised 버튼, Floating 버튼, ListView, 메뉴, 아이템 그룹화, 입력 필드 크기를 기준으로 제시하였다.

II. 안드로이드 앱 테스트

1. 앱 테스트 유형

앱 테스트는 출력 값이 예상 값과 일치하는지 확인하고, 테스트 케이스를 통해 버그가 없는지 확인하는 일련의 과정을 일컫는다[14]. 최소한의 앱 품질을 보장하기 위해서는 다양한 유형의 앱 테스트가 이루어져야 한다. Henry Muccini의 연구[12]에서는 모바일 앱의 특성으로 연결성, 제한된 리소스, 자율성, 사용자 인터페이스 등 10가지를 제시하고 있다. [표 1]은 해당 연구에서 제시한 작동 환경의 특성에 따른 테스트 방법을 정리한 것이다.

표 1. 작동 환경의 특성에 따른 테스트 방법

특성	테스트 방법
연결성	네트워크 신뢰성 및 보안성 테스트
제한된 자원	기능 및 성능에 대한 모니터링
자율성	에너지 소비에 대한 모니터링
사용자 인터페이스	GUI(Graphic User Interface) 테스트
문맥 인지	문맥에 기반 한 기능 테스트
적응	적응의 정확성에 대한 실험
신규 개발 언어	화이트·블랙 박스 테스트, 바이트 코드 분석
신규 OS	적응 가능성 테스트, 신규 OS 테스트
기기와 OS 다양성	다양한 기기와 OS에서의 정상 작동 여부 테스트
터치 스크린	유용성 및 화면 터치에 대한 반응성 테스트

[표 1]에서 확인 가능한 다양한 유형의 작동 환경 특성 중 사용자 인터페이스에 집중하여 GUI 테스트를 본 연구에서는 중점적으로 다룬다. 이를 위해 이하에서는 안드로이드 UI control과 GUI 가이드라인을 연구하여, 4장에서의 GUI 테스트 자동화 툴 개발을 위한 기초 자료를 수집한다.

2. 안드로이드 UI Control

안드로이드는 터치 이벤트를 통해 작동하는 GUI 컴포넌트를 위한 자바 라이브러리를 기본적으로 제공하고 있다. 안드로이드에는 많은 UI Control들이 존재하는데, 앱 개발과정에서 가장 많이 사용되는[15] 9가지 UI Control를 정리하면 [표 2]과 같다.

표 2. 많이 사용되는 UI Control의 명칭별 용도

명칭	용도
TextView	텍스트를 화면에 표시하여 사용자가 볼 수 있도록 지원
EditText	사용자가 텍스트를 입력할 수 있도록 지원
Button	사용자가 클릭하자마자 일부 작업을 수행하도록 지원
ImageButton	버튼에 이미지를 표시하여 작업을 수행하도록 지원
ToggleButton	버튼의 ON/OFF 상태를 표시하도록 지원
RadioButton	사용자가 여러 항목 중 일부를 선택할 수 있도록 지원
RadioGroup	모든 버튼 중 하나만 선택할 수 있도록 지원
CheckBox	사용자에게 중복을 허용하여 선택할 수 있도록 지원
ProgressBar	확정과 불확정 상태가 존재, 진행 상황 표시 지원

상기 9가지 UI Control이 개발 과정에서 가장 중요한 GUI 요소이다. 비교적 적게 사용되는[15] 8가지 UI Control 목록을 정리하면 [표 3]과 같다.

표 3. 적게 사용되는 UI Control의 명칭별 용도

명칭	용도
Spinner	주어진 옵션 목록에서 특정 옵션을 선택하도록 지원
TimePicker	하루 중 특정 시간을 선택할 수 있도록 지원
DatePicker	달력을 보여주고, 특정일을 선택하도록 지원
SeekBar	특정 범위의 값을 선택하도록 지원
RatingBar	등급 및 별점을 매길 수 있도록 지원
AlertDialog	작은 팝업 윈도우로 작동하며, 경고 알림을 지원
Switch	ON, OFF 상태 중 하나를 선택하도록 지원
AutoComplete TextView	자동 완성형 EditText이며, 제한된 사항 중 선택하도록 지원

3. Material Design Guideline

사용자는 동일한 앱이라면 기기가 다르더라도 일관된 방식으로 보여지고 작동할 것을 예상한다[16]. 이러한 예상을 만족시키기 위해서 개발자와 디자이너는 호환성·성능·보안 등에 대한 품질 가이드라인을 준수해야 할 뿐만 아니라 시각적 패턴과 탐색 패턴에 대한 규칙인 Material Design Guideline도 준수해야 한다.

Material Design Guideline에서는 [표 2]와 [표 3]에서 정리한 사항 외에도 다양한 UI control에 대한 제약사항을 정의한다. 또한 명확성·가시성을 토대로 개발 과정 중 디자인 측면의 검토 사항에 대한 기준도 제시하고 있다. Material Design Guideline에서 제시하는 컴포넌트별 제약사항을 정리하면 [표 4]와 같다.

표 4. Material Design Guideline의 컴포넌트별 제약사항

컴포넌트	제약사항
List	알파벳순으로 정렬하여 탐색 용이하도록 제작
Menu	선택된 요소는 송신 요소 위에 표시, 항목 중복 불허
Button	버튼이 많으면 평면 대신 직사각형 버튼을 사용
Item Group	유사한 아이템들은 가까운 곳에 배치하여 제작
Input Field	허용 문자 수 이내는 파란 선, 초과는 빨간 선 표시
Click Area	개별 요소는 최소 48x48 dp, 요소간 거리는 8dp 이상
Floating Button	하단 바에 의해 버튼이 가려지지 않도록 제작
하단 탐색 메뉴	메뉴의 수가 2개 이상, 5개 이하로 구성되도록 제작

2장에서 연구한 안드로이드의 UI control 및 Material Design Guideline의 항목들은 4장에서의 GUI 테스트 자동화 툴 개발의 기초자료로 활용될 것이다. 이하 3장에서는 GUI 테스트를 위한 기존 툴을 스크린 캡처 테스트와 소스코드 분석 테스트로 분류하여 검토한다. 기존 툴에 대해 연구함으로써 그들과 차별성을 가지는 4장의 GUI 테스트 자동화 툴 개발 목적을 명확히 하고자 한다.

III. 기존 GUI 테스트 자동화 툴

1. 스크린 캡처 테스트 툴

1.1 스크린 캡처 테스트의 정의

스크린 캡처 테스트는 작동 화면을 캡처하여 다양한 데이터를 수집·저장하는 자동화 도구를 사용하여 주변 장치의 입력, 스크린 터치 등에 대해 행해지는 테스트를 뜻한다. 스크린 캡처 테스트는 영상 인식 기법과 영상 비교 기법에 기반을 두고 있다. 영상 인식 기법은 벤치마크로 활용되는 요소를 앱 내부의 윈도우에서 찾아 인식하는 방법이다. 영상 비교 기법은 작동 화면을 영상으로 간주하고, 일정 시간 다음의 영상과 비교하여 그 사이의 차이를 파악하는 방법이다.

스크린 캡처 테스트는 2단계로 행해진다. 첫째, 테스터는 스크린 캡처 테스트 툴을 사용하여 가능한 모든 테스트 케이스의 작업을 기록한다. 둘째, 툴은 앱의 기능성 테스트를 위해 모든 사용자 상호작용을 시뮬레이션하여 기록된 모든 작업을 다시 시작한다. 이하에서는 웹과 모바일 환경에서 모두 활용 가능하면서도, 지속적인 업데이트가 이루어지고 있는 스크린 캡처 테스트 툴인 Ranorex, Sikuli에 대해서 검토한다.

1.2 Ranorex

Ranorex는 무료로 시작했으나 현재는 유료인 GUI 테스트 자동화 툴이며, 블랙박스·크로스 브라우저·데이터·키워드·회귀분석·기능 테스트를 제공하고 있다. 테스트 케이스를 빠르게 정의하며, 직관적인 사용성을 지닌다는 장점이 있다.

Ranorex의 작동 방법은 다음과 같다. 첫째, 배포용 파일이나 실행 파일을 캡처한 화면을 영상으로 간주하여 테스트 케이스를 기록한다. 둘째, 기록 과정을 완료한 후 기록된 테스트 케이스에 대해 시뮬레이션을 진행하여 기록된 모든 작업을 재개한다. 끝으로 테스트 결과를 보고서 형태로 생성하여 개발자에게 제공한다.

1.3 Sikuli

Sikuli는 테스터가 화면 캡처를 통해 GUI 컴포넌트에 대해 수행해야 할 사항을 정의할 수 있는 자동화 툴이다. Sikuli는 GUI 컴포넌트의 형태나 이미지를 식별하는 방식으로 작동한다. 즉, 해당 툴은 테스트를 진행하는 동안 영상 비교 기법을 통해 화면 속 GUI 컴포넌트를 식별하고, 식별된 컴포넌트와 가장 유사한 컴포넌트를 찾는 방식으로 작동한다. GUI 컴포넌트를 ID값으로는 식별하지 않기에 소스코드가 테스트에 필요하지는 않다.

2. 소스코드 분석 테스트 툴

2.1 소스코드 분석 테스트의 정의

소스코드 분석 테스트는 앱 배포 전에 앱을 디버깅하기 위해 자동화된 테스트를 수행할 수 있도록 한다[17]. 소스코드 분석은 GUI 컴포넌트의 계층 구조를 분석하여 GUI가 올바르게 구성되었는지 점검하는 기법으로 정적·동적 분석이라는 2가지 유형으로 분류된다. 정적 분석은 실행 불가능한 환경에서 테스트하며, 프로그램 소스코드 자체를 점검한다. 반면에 동적 분석은 실행 가능한 환경에서 앱을 테스트하며, 블랙·화이트박스 테스트를 통해 웹·앱을 점검한다. 이하에서는 소스코드 분석 테스트 툴인 Appuim, Espresso, SelenDroid, UI Automator에 대해서 검토한다.

2.2 UI Automator, Espresso

UI Automator는 구글이 제공하는 개발 환경인 Android Studio 내장 안드로이드 테스트 프레임워크이다. 이는 개발 환경에 내장된 툴이라서 활용이 쉽고 실행 속도도 빠르다. UI Automator는 테스트 대상 앱에 대한 사용자의 Action을 시뮬레이션하기 위해 GUI 컴포넌트를 식별하면서 소스코드 분석을 기반으로 테스트를 진행한다.

Espresso도 UI Automator와 같이 구글에서 제공하는 Android Studio에 내장된 GUI 테스트 자동화 툴이다. Espresso는 앱의 작동·기능·내부 구조를 모두 검토하는 그레이 박스 테스트에 의한다. 또한 안드로이드 개발환경에 내장되어 있기 때문에 GUI 컴포넌트와 텍스트 리소스에 대한 레퍼런스를 직접 식별한다. 앱의 로드 시간을 알 수 있어서 Espresso의 API에는 물리적 버튼을 시뮬레이션하는 메소드(Method)가 존재한다는 점도 특징이다.

2.3 Appium, SelenDroid

Appium은 동일한 API를 사용하여 여러 플랫폼에서 테스트 케이스를 작성할 수 있도록 돕는 오픈소스 멀티 플랫폼 도구이다. Appium은 웹 드라이버 인터페이스를 사용하며, GUI 컴포넌트에 액세스하기 위해 소스코드 분석 기법에 기반하고 있다. Appuim은 테스트 스크립트를 실행할 때 명령들을 UI Automator 혹은 SelenDroid로 전송하여 안드로이드에서 작동할 수 있도록 한다.

SelenDroid 는 웹·앱 테스트를 위해 설계된 기존 프레임워크인 Selenium을 활용한다. 기능은 앞서 살펴본 Espresso와 UI Automator에 비해 다양하며, 브라우저 내장 플러그인을 사용하기 때문에 테스트 자동화를 위해 앱의 바이너리 코드를 수정할 필요가 없다. SelenDroid API를 사용하면 소스코드를 분석하고, GUI 컴포넌트를 반환함으로써 테스트를 자동화할 수 있다.

3. 소결

3장의 연구를 볼 때, 다음과 같은 사항을 알 수 있다. 첫째, 기존 툴들은 사용자 상호작용을 모방하거나 시뮬레이션하기 위해 자동화 된 테스트 케이스 생성과 함께 작동한다.

둘째, 기존 툴들은 대화형 컴포넌트와 해당 연결점을 찾아 앱의 구조를 검사하는 방식과 필요한 GUI 컴포넌트를 찾기 위해 화면을 캡처하는 방식으로 분류된다. 즉, 기존의 GUI 테스트 자동화 툴은 앱의 구조와 컴포넌트 식별에 초점을 두고 있으며, 시각적 설계, 인터페이스 사용 용이성, 컴포넌트 배치에 대해서는 고려하지 않고 있음을 알 수 있다.

모바일 앱은 기능적 완성도가 요구될 뿐만 아니라 시각적 매력도 있어야 한다. 이러한 시각적 매력을 포함하는 GUI 유용성은 OS 제공 기업이 제시한 가이드라인에 따라 달라진다. 따라서 개발 과정에서 기능적 완성도를 테스트할 뿐만 아니라 GUI 가이드라인을 실시간으로 확인할 수 있는 자동화 툴의 개발이 필요하다. 이를 통해 기존 툴에서 미비한 시각적 설계, 인터페이스 사용 용이성, 컴포넌트 배치에 대해서 고려할 수 있게 된다.

Ⅳ. GUI 테스트 자동화 툴 개발 방법

1. 개설

2장에서 구글은 GUI 컴포넌트에 대한 가이드라인을 제시하고 있음을 파악하였다. 그러나 많은 앱이 가이드라인 따르지 않아서 사용자와의 상호작용이 저하되고, 앱 간의 컴포넌트 불일치가 발생한다. 이를 방지하기 위해 Android Core UX-B1 품질 가이드[18]는 GUI 적합성 테스트를 요구하지만, 안드로이드 개발환경에서는 이러한 테스트하는 자동화 툴을 제공하지 않고 있다. 이러한 문제를 해결하기 위해 4장에서는 GUI 테스트 자동화 툴에 대한 개발 방법에 대해서 연구를 진행하였다.

2. 개발 명세

2.1 툴의 작동원리

해당 툴은 Client-Server 아키텍처로 구성되어 있다. [그림 1]은 툴이 작동할 때의 프로브와 인터프리터 사이 인터랙션에 대해서 표현한 것이다. 여기서 프로브는 현재 GUI 상태에 대한 정보를 처리하기 위해 서버에 전달하는 역할을 하는 클라이언트 코드를 뜻한다.

[그림 1]의 좌측은 클라이언트, 우측은 서버 측의 작동 방식이 표현되어 있다. 클라이언트 측인 Java 프로브는 GUI 속성 등을 Json 형식의 프로브 상태 보고서로 생성하여 서버에 전달한다. 서버 측에서는 전달 받은 보고서에 대해서 프로브 로그와 상호작용하며 Material Design Guideline을 기준으로 평가를 진행한다. 평가의 결과는 클라이언트에 메타 데이터로 반환되며, 대시보드를 통해 개발자에게 보여진다.

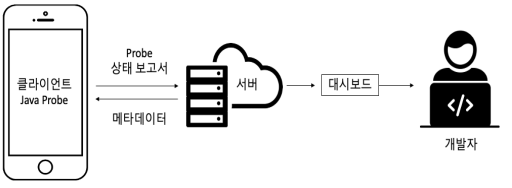


그림 1. 툴 작동원리

2.2 개발 언어 선정

GUI 테스트 자동화 툴(이하 툴)은 클라이언트-서버

아키텍처로 구성되며, 서버는 GUI 컴포넌트의 수집·평가를 지원한다. 이를 위해 사용자는 일차적인 시간 논리 확장에 기반 한 언어인 'LTL-FO+'를 사용한다[19]. 속성에는 경우에 따라 Javadoc 유형의 메타 데이터가 수반될 수 있으며, GUI 가이드라인에 따른 명령 집합은 메모리에 저장되고 고유한 식별자를 갖도록 한다.

2.3 컴포넌트 selector, 이벤트, Time Operator

툴에서 윈도우 컴포넌트는 속성을 표현하기 위해 선택되며, 속성은 선택된 1개 이상의 컴포넌트에 적용될 수 있다. 컴포넌트는 태그 및 ID 속성 값으로 선택하거나, jQuery 구문을 사용하여 객체 인스턴스 값으로 선택할 수 있다.

그리고 툴에서 사용자가 트리거(trigger)한 이벤트는 해당 컴포넌트의 속성으로 표현되며, Time operators는 일정 시간동안 동일한 컴포넌트의 상태를 평가할 때 사용된다.

2.4 Property 관리

앱 화면은 Activity 클래스를 상속하는 클래스들에 의해 구현된다. 프로브는 특정 Activity의 이름과 일치하는 속성을 전송하도록 서버에 요청하며, 서버는 해당 속성의 내용을 반환(return)한다.

2.5 컴포넌트 분석

프로브는 컴포넌트 내용을 분석하고 사용자가 트리거한 이벤트 관련 컴포넌트의 실시간 상태를 전송한다. Activity 간의 연결은 이벤트가 감지될 때, 프로브가 상위 View 객체의 부모와 자식 클래스를 반복적으로 탐색하여 연결 노드를 찾는다.

2.6 서버와의 통신

서버는 JSON 파일을 수신하고, 소유권 귀속 주체를 실시간으로 파악하여 해당 소유 주체의 메타데이터를 가이드라인과 오류 수준에 대한 링크로 포함시킨다. 현재 속성 정보는 GUI와 상호작용하여 테스터에게 시각적 피드백을 제공한다.

3. 컴포넌트별 소스코드 예시

3.1 하단 탐색 메뉴(Bottom Navigation Menu)

하단 탐색 메뉴(Bottom navigation menu)에 대한 가이드라인 테스트를 개발하는 방법은 다음과 같다. 2장의 [표 4]에 따르면 하단 탐색 메뉴는 2개 이상과 5개 이하를 동시 조건으로 만족해야 한다. 이를 코드로 표현하면 [그림 1]과 같다.

```
For each $x in $(BottomNavigationMenuView) (
    ($x's size is greater than  2)
     And
    ($x's size is less than  5)
).
```

그림 1. 하단 탐색 메뉴의 제약사항 코드화 예시

3.2 Raised 버튼

Raised Button은 버튼 아래에 그림자가 있는 버튼을 지칭하며, Flat Button과는 다르게 버튼이 튀어나온 형태이다. 세부사항이 많은 경우에는 Raised Button을 사용하는 것이 적절하다. 이에 대한 가이드라인 테스트를 개발하는 방법은 다음과 같다. [그림 2]의 상단에서는 RaisedButton의 가이드라인을 코딩하고, 하단에서는 z가 RaisedButton이니 테스트하도록 지시하였다.

```
We say that $x is RaisedButton when (
  ( (($x's width / the device's device-density) is greater than 47 ) And
    (($x's height / the device's device-density) is greater than 47)
  ) And
  ( Not ($x's background matches "RGB\(0,0,0\)")
    )
).

 For each $z in $(Button) (
 $z is RaisedButton
 ).
```

그림 2. Raised 버튼의 제약사항 코드화 예시

3.3 Floating 버튼

Floating Button은 [표 4]에 따르면 하단 바에 의하여 숨겨지거나 가려지지 않도록 제작되어야 한다. [그림 3]은 하단의 Snackbar에 Floating Button이 가려지지 않도록 코드로 구현한 것이다. Floating Button의 상단 높이와 Snackbar의 상단의 높이의 차이 값이 Floating Button의 높이보다 큰 것을 기준으로 했다.

```
Always (
    For each $x in $(#fab)  (
        For each $y in $(TextView)(
            If ($y's parent equals "SnackbarLayout")
            Then (($y's top  - $x's top) is greater than $x's height)
        )
    )
).
```

그림 3. Floating 버튼의 제약사항 코드화 예시

3.4 아이템 그룹화

[표 4]에 따르면 유사한 아이템 항목은 하나의 그룹으로 보아 서로 가까운 곳에 배치하여 제작해야 한다. [그림 4]에서는 유사한 아이템인 TextView들이 Slider 옆에 가지런히 놓이도록 하였다. 다양한 컴포넌트의 그룹화에 응용할 수 있다.

```
For each $z in $(RelativeLayout SeekBar) (
    There exists $x in $(RelativeLayout TextView) such that (
        $z's top equals $x's top
    )
).
```

그림 4. 아이템 그룹화의 제약사항 코드화 예시

3.5 입력 필드 크기

[표 4]에 따르면 Input Field는 문자의 수 제한 초과 시 빨간 선을, 초과 전에는 파란 선을 이용하여 제작해야 한다. 이를 위해 우선 빨간 선과 파란 선에 대해서 정의하였다. Input Field Size가 10을 초과하는 경우는 redline을, 11 미만인 경우에는 blueline으로 표시되도록 코드를 [그림 5]와 같이 작성하였다.

```
We say that $x and $y are redline when (
  ($x's length is greater than 10) And
  ($y's color matches "RGB\(255,0,0\)")
).

We say that $x and $y are blueline when (
  ( $x's length is less than 11)
  And ($y's color matches "RGB\(0,0,255\)")
).
```

그림 5. 입력 필드 크기의 제약사항 코드화 예시

3.6 ListView

ListView는 한글의 경우 가나다순으로, 영어의 경우 알파벳순으로 정렬되어야 한다. [그림 6]은 이러한 정렬에 대한 가이드라인을 코드로 구현한 것이다.

```
For each $x in $(ListView TextView) (
    For each $y in $(ListView TextView) (
        If ($y's top is greater than $x's top)
        Then ($y's text is greater than $x's text)
    )
).
```

그림 6. ListView의 제약사항 코드화 예시

3.7 메뉴

[표 4]에 따르면 메뉴는 선택된 메뉴 요소가 송신 요소 위에 표시되도록, 선택 항목 간 중복되지 않도록 제작해야 한다. Spinner 클릭 이벤트를 처리하기 위해 우선 'I click on Spinner'를 [그림 7]과 같이 작성하였다. 그 후에는 not equals에 대해서 [그림 8]과 같이 정의하였다. 이는 항목 간 중복 여부 판별에 활용된다.

```
We say that I click on Spinner when (
  There exists $b in $(Spinner) such that (
    $b's event is "ACTION_UP"
  )
).
```

그림 7. Spinner 클릭 이벤트 처리 예시

```
We say that $x and $y are not equals when (
    Not(
            $x's text equals $y's text
    )
).
```

그림 8. 선택 항목 간 중복 배제 처리 예시

Spinner를 클릭하는 이벤트가 발생하면 다른 item이 [그림 8]에서 정의한 'not equals'인지 [그림 9]와 같이 판독한다. 이를 통해 메뉴 아이템의 중복 여부를 판단할 수 있다.

```
If (I click on Spinner)
Then (
For each $x in $(Spinner item) (
    For each $y in $(Spinner item) (
    ($x's position equals $y's position) Or
        ( $x and $y are not equals)
    )
  )
).
```

그림 9. 메뉴의 제약사항 코드화 예시

4. 소결

4.1 툴 활용 예시

3.3에서 Floating Button은 하단 바에 의하여 숨겨

지거나 가려지지 않도록 소스코드를 작성하였다.

[그림 10]의 좌측은 하단 바에 의해 버튼이 가려지는 상황으로 상태 표시 상자가 빨간색으로 표시됨으로써 GUI 가이드라인을 위반했음을 알려준다. [그림 10]의 우측은 하단 바에 의해 버튼이 가려지지 않도록 GUI가 수정된 상황으로 상태 표시 상자가 초록색으로 표시됨으로써 GUI 가이드라인을 위반하지 않았음을 알려준다.

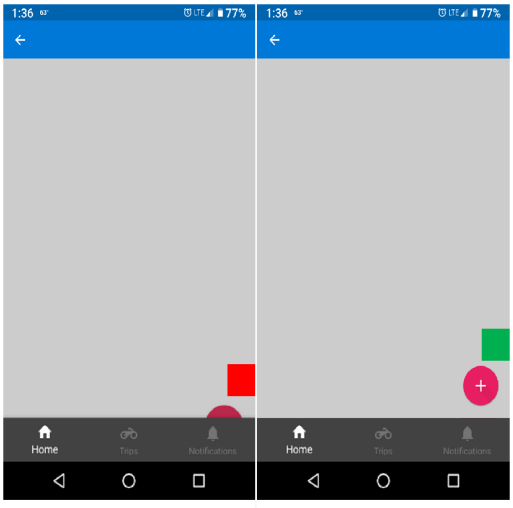


그림 10. Floating Button 관련 활용 예시

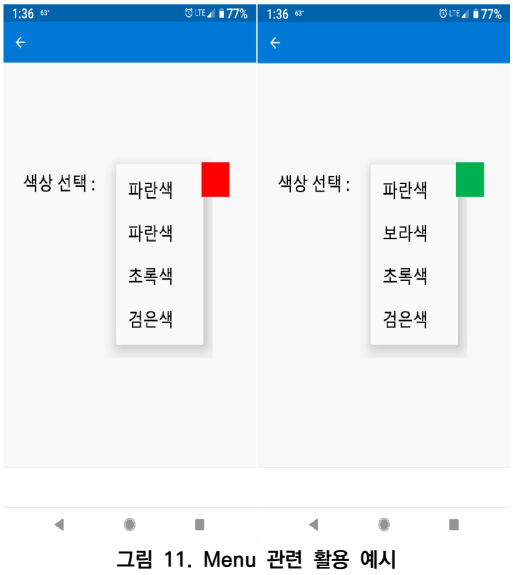


그림 11. Menu 관련 활용 예시

3.7에서 메뉴는 선택된 메뉴 요소가 송신 요소 위에 표시되도록, 선택 항목 간 중복되지 않도록 소스코드를 작성하였다. [그림 11]의 좌측은 선택 항목 중 '파란색'이 중복되고 있어 GUI 가이드라인을 위반한 상태이다. 따라서 상태 표시 상자가 빨간색으로 표시됨으로써 GUI 가이드라인을 위반했음을 알려준다. [그림 11]의 우측은 중복되었던 파란색 중 하나를 보라색으로 수정하여 중복되지 않게 된 상황으로 상태 표시 상자가 초록색으로 표시됨으로써 GUI 가이드라인을 위반하지 않았음을 알려준다.

4.2 새로운 툴의 이점

3장에서 기존의 GUI 테스트 자동화 툴은 앱의 구조와 컴포넌트 식별에 초점을 두고 있으며, 시각적 설계, 인터페이스 사용 용이성, 컴포넌트 배치에 대해서는 고려하지 않고 있음을 알 수 있다. 4장에서 제시한 개발 방법을 토대로 GUI 테스트 자동화 툴이 개발되는 경우에는 [표 5]와 같은 이점이 있다. Espresso, UI Automator, Appuim, SelenDroid는 앱 구조 파악 및 컴포넌트 식별을 할 수 있다. 그러나 시각적 설계, 인터페이스 사용성, 컴포넌트 배치의 적절성에 대해서는 고려하지 못한다. 새로운 툴은 시각적 설계, 인터페이스 사용성, 컴포넌트 배치의 적절성에 대해서 고려할 수 있다는 점에서 이점이 있다. 또한 SelenDroid에서 불가한 실시간 테스트가 가능하다는 점에서도 이점이 있다고 할 수 있다.

표 5. 기존 툴과 대비한 새로운 툴의 이점

	기존 툴	새로운 툴
앱 구조 파악 여부	○	○
컴포넌트 식별 여부	○	○
시각적 설계 고려 여부	×	○
인터페이스 사용성 고려 여부	×	○
컴포넌트 배치 고려 여부	×	○
실시간 테스트 여부	△	○

V. 결론

4장에서의 툴은 테스트 시간을 줄이고 앱 생산 생산성을 높이기 위해 안드로이드 GUI 가이드라인 위반 여

부를 판독하기 위한 자동화 툴 개발 방법을 제시하였다. 이를 통해 Android Core UX-B1 품질 가이드에서 요구하는 GUI 적합성 테스트가 이루어진다. 또한 3장에서 검토한 기존 GUI 테스트 자동화 툴에서 미비했던 시각적 설계, 인터페이스 사용 용이성, 컴포넌트 배치에 대해서도 고려할 수 있게 된다. 수많은 앱이 빠르게 개발되고, 소비되는 현 상황에서 본 연구는 GUI 테스트의 개발 방향을 제시함으로써 개발사의 안정적인 앱 배포에 기여할 수 있다는 점에서 의의가 있다.

후행 연구에서는 4장의 예시 소스코드 외에도 다양한 컴포넌트에 대한 코딩이 이루어져야 한다. 또한 개발 과정에서 실제로 활용되기 위해서는 많은 개발자들이 투입되어 본 연구의 개발 방법을 토대로 상용 툴 개발이 이뤄져야 할 것이다.

참 고 문 헌

[1] 김성용, 민대환, 임성택, 채봉수, “SW 테스트 자동화 구현을 위한 핵심성공요인에 관한 연구: 웹 어플리케이션 소프트웨어를 중심으로,” JITAM, 제27권, 제6호, pp.101-116, 2020.

[2] 김성용, 민대환, 임성택, “데이터 주도 접근법을 활용한 소프트웨어 테스트 자동화,” 한국IT서비스학회지, 제17권, 제1호, pp.155-170, 2018.

[3] 안정우, *암호 소프트웨어 개발 생명주기에 대한 자동화된 테스트 체계 연구*, 국민대학교 대학원, 박사학위논문, 2020.

[4] 최지훈, 박구락, 박원철, “V모델 기반의 테스트 자동화 설계 모델에 대한 연구,” 한국컴퓨터정보학회 학술발표논문집, 제28권, 제2호 pp.593-594, 2018.

[5] A. Sivji, “Software Testing Automation: A Comparative Study on Productivity Rate of Open Source Automated Software Testing Tools For Smart Manufacturing,” 2020 ICOS, Kota Kinabalu, Malaysia, pp.7-12, 2020.

[6] Dilara Ateşoğulları and Alok Mishra, “Automation Testing Tools: A Comparative View,” International Journal on Information Technologies & Security; Vol.12 Issue.4, p63-76, 2020.

[7] 정범진, 이정우, 홍창완, 안병구, “윈도우 환경에서의 GUI 기반 블랙박스 테스트 자동화 프로그램 도구,” 한국인터넷방송통신학회지, 제18권, 제2호, pp.163-168, 2018.

[8] 김진영, *웹 환경에서 테스트 케이스 자동 생성을 통한 GUI 테스트 자동화 시스템 설계 및 구현*, 가천대학교 일반대학원, 석사학위논문, 2019.

[9] 정선구, *Sikuli를 이용한 GUI 테스트 케이스 자동 생성 도구 설계 및 구현*, 한양대학교 공학대학원, 석사학위논문, 2018.

[10] 김하연, 노혜민, 유철중, “모바일 애플리케이션 GUI 테스팅을 위한 GUI 이벤트 추출 크롤링 기법,” 한국정보기술학회논문지, 제17권, 제12호, pp.135-145, 2019.

[11] 최민창, *테스트 자동화를 위한 딥러닝 기반 GUI 컴포넌트 탐지 기법*, 성균관대학교 일반대학원, 석사학위논문, 2019.

[12] H. Muccini, A. Di Francesco, and P. Esposito, “Software testing of mobile applications: Challenges and future research directions,” 2012. 7th International Workshop on Automation of Software Test (AST), Zurich, Switzerland, pp. 29-35, 2012.

[13] www.gsmarena.com/makers.php3

[14] wikipedia.org/wiki/Mobile_application_testing

[15] data-flair.training/blog/

[16] developer.android.com/guide/topics/ui/

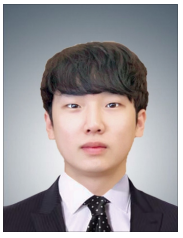
[17] J. Itkonen, “Defect Detection Efficiency: Test Case Based vs. Exploratory Testing,” ESEM, 2007.

[18] developer.android.com/docs/quality-guidelines/core-app-quality

[19] Khoury and Hallé, Lebrun. “Automata-based monitoring for LTL-FO+,” Int J Softw Tools Technol Transfer, 2020.

저 자 소 개

박 세 준(Se-Jun Park) 준회원



■ 2015년 ~ 현재 : 숭실대학교 글로벌미디어학부생

〈관심분야〉 : UX 디자인, 디지털 전략 기획

김 규 정(Kyu-jung Kim) 정회원

■ 2005년 ~ 현재 : 숭실대학교 글로벌미디어학부 교수

〈관심분야〉 : 미디어아트, 영상미디어