DOI: 10.3745/KIPSTD.2008.15-D.3.347

이미지 플로우 기반의 모바일 GUI 테스트 도구에 관한 연구

황 선 명[†] · 윤 석 진[††]

요 약

모바일 어플리케이션 소프트웨어의 시장은 여러 소프트웨어 시장 중 가장 많은 어플리케이션 소프트웨어를 출시하고 있으며 모바일 어플리케이션에서 가장 중요한 사용자와의 정보 교환 수단으로는 GUI 가 있다. 현재까지의 모바일 어플리케이션에서의 GUI 테스트 방법으로는 테스터가 한단계, 한단계 버튼을 눌러가며 화면을 체크하는 원시적인 방법의 테스트가 이루어지고 있다. 이에 본 논문에서는 모바일 상에서 이루어지는 정적 화면 전환의 경우 테스트 수행 결과를 이미지 플로우 기반으로 표시함으로써 GUI를 테스트 하는 방법을 제시하고 테스트 커버리지까지 측정할 수 있는 방법을 제시한다.

키워드 : 이미지 플로우, GUI테스팅, 테스트커버리지

Mobile GUI Testing Tool Based-on Image Flow

Sun-Myung Hwang[†] · Seok-Jin Yoon[††]

ABSTRACT

In order to enhance the productivity of mobile and develop reliable software, the test of mobile application software should be required absolutely. The most important way to communicate to users has been used to GUI. But GUI test method in mobile has no the test automation system except manual test by checklist. In this paper we present a test method and tool by image flow to reduce the time required and finds out errors to GUI, by carrying out the study on automatic GUI testing tool based on image flow to GUI test of mobile application.

Key Words : Image Flow, GUI Testing, Test Coverage

1. 서 론

모바일 어플리케이션 소프트웨어의 시장은 여러 소프트웨어 시장 중에서 가장 많은 어플리케이션 소프트웨어를 출시하고 있다. 또한 모바일 소프트웨어는 짧은 개발주기와 짧은 생명주기로 인해 빠른 출시와 시장선점이 가장 큰 이슈이다.[4]

GUI는 Graphics User Interface의 약자로서 사용자가 그래픽을 통해 정보를 교환하고자 하는 대상과 정보를 교환하는 작업환경을 일컫는다. 나날이 변화하는 모바일 어플리케이션 소프트웨어의 GUI는 점차 중요한 요소로 자리 잡고 있으며, 모바일 어플리케이션의 중요 테스트 대상으로 관심이 증가되고 있다. 하지만 일반적으로, 이동 통신 단말기의 소프트웨어뿐만 아니라, 대부분의 소프트웨어의 GUI의 테스트는 사람에 의존할 수밖에 없는 것이 지금까지의 현실이었다.

그러나 이렇게 사람에 의한 사용자 인터페이스 테스트의 경우에는 사람의 노동력이 오랜 시간 동안 지속적으로 투입되어야 한다는 단점이 있다. 그래서 최근에 일부 통신 사업자의 경우에는 사용자 인터페이스를 자동적으로 테스트하는 방법을 개발하고 있는데, 이러한 방법 자체도 역시 그리 효율적이지 못 하고 있다.

본 논문에서는 모바일의 GUI 테스트를 하는데 기존의 테스터가 단계적으로 모든 화면에 대한 테스트를 하던 문제점을 극복하고자 "MoGuT Image Flow(Mobile GUI Testing for Image Flow)"를 구현하고, 이미지 플로우를 제공하여 개발자 또는 테스터가 오류부분을 명확하게 판별할 수 있는 기준을 제안 하고자 한다[11].

본 논문의 2장에서는 관련 연구로 CLDC와 미들렛에 대하여 살펴보고, 3장에서는 이미지 플로우 기반 테스트 도구의 시스템의 구조에 대해야 알아보며, 4장에서는 데이터를

※ 본 연구는 정보통신부 및 정보통신연구진흥원의 IT신성장동력핵심기술개발사업의 일환으로 수행하였음. [2007-S032-01, 다중 플랫폼 지원 모바일 응용 S/W 개발환경 기술 개발]
† 종신회원 : 대진대학교 컴퓨터공학과 교수
†† 정 회 원 : 한국전자통신연구원 선임연구원
논문접수 : 2008년 12월 17일
접 수 일 : 2008년 2월 14일
심사완료 : 2008년 2월 16일

이미지 플로우로 구성하고 테스팅하는 방법에 대하여 제시하고, 5장에서는 Image Flow의 적용평가에 대하여 기술하고, 마지막으로 6장에서 결론을 맺는다.

2. 관련 연구

2.1 CLDC

CLDC(connected limited device configuration)는 성능이 제한된 CPU나 메모리가 한정적인 시스템을 대상으로 하는 spec이다. CLDC 관련 library는 시스템에 독립적인 고수준의 API 그리고 네트워크관련 API로 되어 있다. 크게 두 가지로 나눌 수 있는데 J2SE에 포함되어 있는 부분과 CLDC에만 있는 부분이다.[13]

J2SE에 포함된 부분	CLDC에만 관련된 부분
- java.lang	- javax.microedition.io.*
- java.util	
- java.io	

CLDC에서 주목해야할 몇 가지 특징들은 다음과 같다.

- 부동 소수점을 지원하지 않는다.
- 마무리(finalization)을 지원하지 않는다.
- 에러 처리가 제한적이다.
- JNI(Java Native Interface)를 지원하지 않는다.
- 리플렉션을 지원하지 않는다.
- 쓰레드 그룹과 데몬 쓰레드를 지원하지 않는다.
- 사용자 정의 클래스 로더를 생성할 수 없다.
- 약한 참조(weak reference)를 지원하지 않는다.
- 클래스 검증과정이 오프디바이스와 온디바이스로 - 2단계로 나뉘어 졌다.
- 클래스 파일 포맷이 다르고, 클래스 룩업, 클래스 로딩과 링킹 방법이 다르다.
- 모래상자 보안 모델을 사용한다.
- 전혀 다른 네트워킹 및 입출력 모델을 가지고 있다.
- 새로운 애플리케이션 모델을 가정하고 있다.

2.2 미들렛

미들렛은 Mobile Information Device Profile(MIDP) 어플리케이션이다. 애플릿과 마찬가지로, 미들렛은 managed 어플리케이션이다. 그러나 애플릿의 경우처럼 웹 브라우저에 의해 manage되는 것이 아니고, 양방향 페이져나 핸드폰에 탑재되는 어플리케이션을 제어하기 위해 특별히 만들어진 소프트웨어인 AMS에 의해 manage된다. 어플리케이션이 실행되는 도중 전화가 온다면, 어플리케이션으로 인해 전화 수신이 방해받아서는 안 될 것이다. 그러므로 이런 경우, 무선기기에서 실행되는 어플리케이션의 경우 이러한 외부적인 management가 적절하다.

미들렛 모델은 PDAlet을 지원하기 위한 Personal Digital Assistant Profil(PDAP)에서도 지원된다.

2.3 J2ME 플랫폼

J2ME 비록 휴대폰이나 임베디드 기기를 위한 플랫폼이지만, 기본적으로 자바가 가지고 있는 특징인, 객체지향방식의 프로그래밍, 코드의 높은 이식성, 안전한 네트워크 보안 지원 및 J2SE와 J2EE와의 상위 호환성은 그대로 유지하고 있다. 여기에 더하여 J2ME는 다음과 같은 특징을 지니고 있다. 여기에 더하여, J2ME는 다음과 같은 특징을 지니고 있다.[14]

2.3.1 다중플랫폼호환성(cross-platform compatibility)

100% 순수한 J2ME API를 사용하는 애플리케이션은 쉽게 다른 업체가 만든 모델과 광범위하게 호환될 수 있다. 단말기가 다르다고 해서 프로그램 개발자 측에서 같은 프로그램을 다시 개발하는 일은 상당히 번거로운 일이다. 하지만 J2ME 플랫폼은 어느 단말기에서든지 시간과 장소에 관계없는 강력한 호환성을 가진다.

2.3.2 보안성

무선 인터넷 환경에서의 보안은 아직도 해결되지 않는 문제점들이 많다. J2ME는 기존 자바의 보안 모델을 무선통신에 적용시킴으로써, 이를 상당부분 해결하고 있다. J2ME로 작성된 애플리케이션은 기본적으로 디바이스의 하드웨어나 다른 리소스에 직접 접근할 수 없기 때문에 바이러스나 다른 악성 프로그램을 만들어낼 수 없다.

2.3.3 동적 애플리케이션 다운로드

무선 서비스를 통한 자바 애플리케이션들은 실시간적으로 동적으로 다운로드 되므로 사용자들은 A/S 센터를 방문하고나, 업그레이드를 쉽게 받을 수 있다.

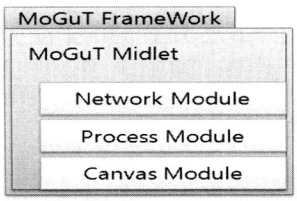
3. MoGuT System 설계

본 장에서는 기존의 눈으로만 테스트하던 매뉴얼 방식의 모바일 GUI 테스트 방법을 탈피하여 소프트웨어적으로 테스트하고 기존의 문제점을 개선하며 이미지 플로우를 이용한 모바일 GUI 테스트 방법을 제시하고자 한다.

MoGuT System은 에뮬레이터 또는 폰에 구현코드와 같이 탑재되어 구동되는 'MoGuT Framework'과 모바일에서 전송하는 데이터를 받아서 테스트를 수행하는 서버역할을 하는 'MoGut Image Flow'로 구성되어 있으며 모바일 모듈에서는 모바일 상에 표시된 화면 이미지와 현재의 화면 코드, 그리고 키 이벤트 코드를 서버에 전송하는 기능을 가지고 있다. 서버 모듈은 모바일 모듈에서 전송한 데이터들을 분류하고 처리 및 테스트하는 기능을 가지고 있다.

3.1 MoGuT Framework 설계

MoGuT Framework은 아래 (그림 1)과 같이 MoGuT Midlet을 기본으로 MoGuT Image Flow와의 통신을 담당하

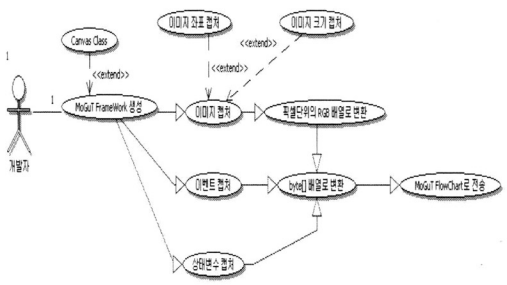


(그림 1) MoGuT Framework 아키텍처

는 Network 모듈과, 화면을 캡쳐하고 자르고 데이터를 생성하는 Process 모듈 그리고 위의 작업들을 할 수 있도록 실질적인 기능을 담당하는 Canvas 모듈이 있다.

MoGuT Framework은 이미지와 필요한 데이터들을 서버에 전송하는 역할만 하는 부분으로 모바일 상에 탑재되어 구동되어야 하기 때문에 최소한의 기능과 코드로 이루어져 있다. 또한 모바일용 API인 J2ME의 경우 모바일의 특성상 지원하는 기능이 극히 제한되어 있다. 그렇기에 객체를 직렬화하는 기능도 존재하지 않기에 해당 데이터들을 byte형의 형태로 변환하여 서버로 전송해야 한다. 이에 해당하는 유즈케이스 다이어그램은 아래 (그림 2)와 같다.

MoGuT Framework는 총 4개의 클래스를 가진다. 통신방법에 따라 네트워크 클래스는 늘어날 수 있다. TCP/IP, Socket 등등 휴대폰으로 통신을 할 수 있는 방법은 여러 가지가 있기 때문에 필요에 따라서 통신방법을 달리해 사용할 수 있다.



(그림 2) MoGuT FrameWork 유즈케이스 다이어그램

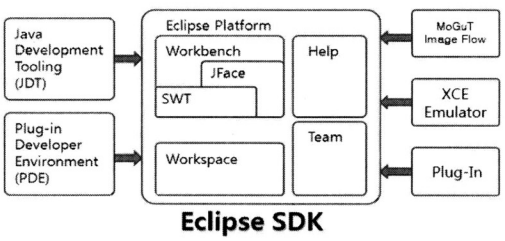
아래 (그림 3)에서 보듯이 클래스들은 상속이 아닌 다른 클래스들의 존재를 알고 있는 형식으로 표현된다. 그 이유는 각 클래스들은 상속을 받아야 하는 클래스들이 있고 다중 상속을 막기 위해서이다. 개발자가 작성한 코드에 코드를 삽입 하는 방식이 아니고 구현코드를 얻어와 테스트만 하고 삭제되어야 하기 때문이다.

3.2 MoGuT Image Flow 설계

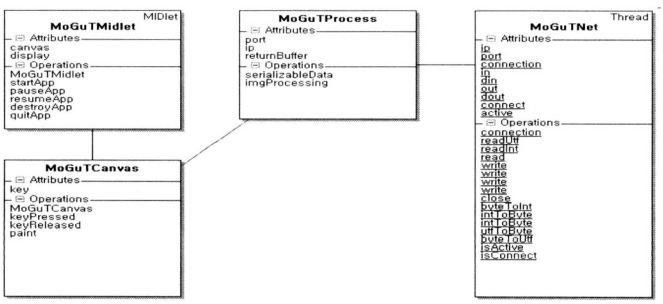
MoGuT Image Flow는 이클립스 기반위에 플러그인 형식으로 이클립스와 맞물려 돌아가는 프로그램이다. 플러그인으로 개발을 하게되면 이클립스의 기능들을 이용할 수 있고 또한 코드 에디터를 따로 제작하지 않아도 되는 장점이 있다. 이클립스의 아키텍처와 MoGuT Image Flow의 관계는 아래 (그림 4)과 같다. Eclipse SDK에 MoGuT Image Flow 뿐만 아니라 XCE Emulator 까지 Plug-In으로 포함시켜 에뮬레이터를 실행하는데 필요한 기능까지 구성하였다.

MoGuT Image Flow의 경우 5가지의 플러그인으로 구성되어 있는데 처리 흐름은 아래 (그림 5) MoGuT Image Flow 유즈케이스 다이어그램을 통해 알 수 있다.

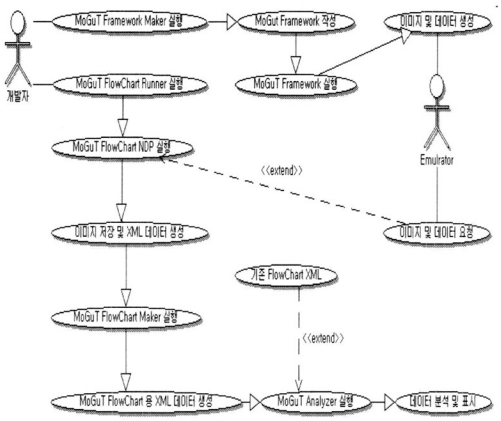
개발자가 Framework Maker 실행을 통해서 Framework를 작성하고 이미지와 데이터를 생성하여 Emulrator에 저장하여 놓는다. 그럼 네트워크를 통하여 이미지 및 데이터를 MoGuT Image Flow가 요청하면 전달하고 데이터들이 처리되는 과정을 유즈케이스 다이어그램에서 보여주고 있다.



(그림 4) 이클립스와 MoGuT Image Flow의 관계도



(그림 3) MoGuT Framework 클래스 다이어그램



(그림 5) MoGuT Image Flow 유즈케이스 다이어그램

4. MoGuT System의 구현

본 장에서는 MoGuT System을 구현한 방법과 데이터의 처리방법 그리고 각각의 요소 기술등에 대하여 살펴본다.

4.1 데이터 직렬화 기술(Encoding)

모바일 플랫폼 즉 J2ME에서는 성능이 제한된 CPU나 메모리가 한정적인 시스템이어서 시스템에 독립적인 고수준의 API 그리고 네트웍 관련 API만 한정적으로 지원하고 있다. 모바일의 스펙은 아래 <표 1>과 같다.

이러한 제한 때문에 리플렉션이 지원되지 않는데 리플렉션이란 런타임시에 자바 프로그램이 가상머신 내부의 클래스, 인터페이스, 객체 인스턴스들을 조사할 수 있게 하는 자바 가상머신의 특징이다. CLDC(Connected, Limited Device Configuration) 에서는 이러한 리플렉션을 지원하지 않기 때문에 리플렉션에 의존적인 객체 직렬화가 지원되지 않는다.

GUI 의 데이터를 네트워크로 전송하고 처리를 하려면 이미지 객체를 직렬화하여 처리를 해야 한다. 그러나 리플렉션이 지원되지 않기 때문에 MoGuT Framework에 Encoder를 제작하고 MoGuT Image Flow에 Decoder를 제작하여 이를 해결하였다.

〈표 1〉 모바일의 CLDC 스펙

구 분	CLDC
Processor	16bit 또는 32bit
네트웍	때로는 끊어지기도 하는 저속 통신망을 사용하고 종종 TCP/IP를 사용하지 않는다.
메모리	32KB - 512KB
가상머신	K Virtual Machine
목표시장	개인적이고 이동중에 사용하며 상호 연결된 정보기기들이다.
적용 예	휴대폰, 무선호출기, 스마트폰, POS 단말기

화면 이미지 객체는 이미지 객체에 있는 모든 픽셀값들을 0xAARRGGBB 형태의 값으로 변환하여 int[]형 배열에다가 담았다가 네트워크로 전송할 수 있는 데이터의 형태인 byte[]형 배열로 재 변환을 시켜 다시 담는 과정을 함으로써 Encoder 역할을 하게 된다.

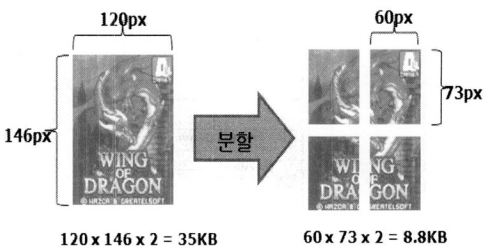
4.2 데이터 변환 기능

모바일에서는 메모리의 크기가 작기 때문에 화면 하나를 그리는데 드는 메모리 비용이 크다. 일반적으로 메모리의 크기가 32KB - 512KB 인데 기본적으로 드는 메모리가 240 x 320 픽셀 사이즈의 경우 화면 해상도(Resolution)와 색심도(Color Depth)의 곱으로 계산하여 152Kb의 메모리가 들게 된다. 테스트를 위하여 한 장의 화면에 해당하는 메모리가 필요하기 때문에 아무런 처리 없이 화면만 그릴 경우에 기본 화면과 테스트에 필요한 화면 두장을 그리는데 필요한 메모리가 304KB 라는 용량이 나오게 된다. 또한 화면에 그리는 메소드, 키 이벤트를 처리하는 메소드, 이미지를 불러오는 역할 등등을 따지면 메모리에서 오버헤드가 발생하여 프로그램이 멈출 수 있다. 이러한 현상을 막기 위하여 이미지를 분할하여 전송을 하는 기술을 적용하였다.

이미지를 분할하는 기준은 이미지를 그리는데 필요한 메모리 크기를 10KB 로 제한하여 자르게 되는데 width/2 와 height/2 의 기준으로 이미지를 나누게 된다. 이에 대한 설명은 아래 (그림 6)에 표현되어 있다.

이미지를 분할할 때 width/2 와 height/2의 계산 공식을 사용한 이유는 핸드폰의 경우 LCD의 크기가 폰마다 다르고, 또한 폰에 따른 이미지의 크기들도 다르기 때문이다.

이렇게 분할된 이미지를 byte[] 배열에 한번에 한 장의 분할 이미지를 넣게 되고 그와 함께 이미지의 좌표인 x축 시작좌표, y축 시작좌표를 함께 전송하고 이미지를 다시 int[]형 배열로 변환 할 수 있도록 이미지의 픽셀 사이즈 크기를 알려주어야 하기 때문에 분할 이미지의 넓이와 높이도 함께 전송이 된다. 또한, 이미지와 함께 현재 화면의 화면 상태 변수의 값과 어떠한 이벤트를 통하여 화면이 변하였는지에 대한 데이터를 얻기 위하여 키 이벤트를 발생시키게 되는 keyPressed() 메소드의 값을 함께 전송하게 된다. MoGuT Image Flow에서는 이들 데이터를 순서와 형식에 맞게 Decoding 하여 사용하게 된다.



(그림 6) 이미지 분할

4.3 데이터 복원 및 키이벤트 입력 기법

MoGuT Framework에 의해서 Encoding된 데이터를 이미지와 각 데이터로 복원하는 기술로 이미지 객체를 int[]형 배열로 변환하기 전에 먼저 이미지의 사이즈에 대한 정보를 먼저 불러옴으로써 int[]형의 배열의 크기를 설정하고 그 배열에 픽셀 데이터들을 담는 역할을 하게 된다. 그 이후에 나머지 값들인 키 이벤트 값과 화면 상태 값, 이미지의 x 시작 좌표, 이미지의 y 시작 좌표들을 변수에 담게 된다. 이러한 모든 작업들은 스레드를 통하여 이루어진다.

또한 이미지 플로우를 작성하기 위해서는 화면 상태 전환에 대한 모든 키 이벤트에 대한 데이터가 필요하다. 그러나 화면의 수가 늘어날수록 입력해야 하는 키 이벤트의 수가 (화면의 수 x 키 버튼의 수) 만큼의 테스트가 필요하므로 테스트의 속도가 느려지게 된다. 이에 대한 처리 속도를 높이기 위하여 해당 프로그램에 사용되는 키 버튼을 알아내는 방법이 있다.

또한 이외의 문제점으로 어떠한 특정화면 상태에서 어떠한 키를 눌렀었는지에 대한 정보도 필요하게 된다. 그렇기에 이에 대한 정보를 저장하고 있는 배열이 필요하게 된다.

해당 배열은 2차원 배열을 가지게 되며 배열의 정보는 다음과 같다.

KeyEvent[상태 번호][키 이벤트 번호];

이를 통하여 전체 화면 대비 테스트 커버리지를 구할 수 있다. 커버리지를 구하는 공식은 다음과 같다.

(화면당 테스트한 키이벤트의 수) ÷
(전체 화면의 수 × 전체 키 이벤트의 수) * 100

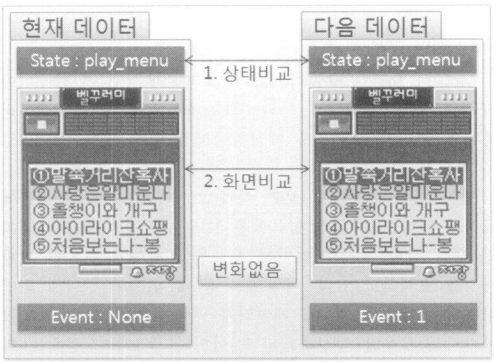
4.4 이미지 플로우 작성 기법

MoGuT Framework에서 전송된 데이터를 기초로 하여 작성된 XML 데이터를 이미지 플로우 작성용 XML 데이터로 변환하는 것은 MoGuT Analyzer에서 처리하게 된다. 기본적으로 MoGuT Framework에서 작성된 XML의 형식은 분할된 이미지를 기초로 하여 작성되어 있는데 MoGuT Analyzer에서 분할된 이미지를 합쳐서 한 장의 이미지로 다시 재변환을 시키게 된다. 또한 그림과 동시에 키 이벤트와 상태 변수를 체크하여 다음 화면에 대한 정보도 찾게 된다.

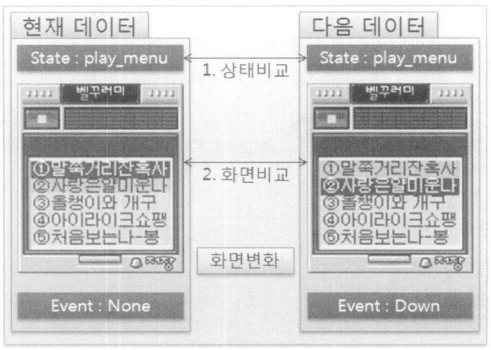
변환된 화면을 찾는 방법으로는 먼저 현재 데이터의 화면 상태와 다음 데이터의 화면 상태를 비교하여 상태별 변환 정보에 저장을 하고, 다음으로 화면의 픽셀데이터들을 비교하여 화면별 변환 정보에 저장을 하게 된다.

다음의 (그림 7)에서 보면 상태 비교 시 상태가 같고 다음의 화면 비교 시 화면의 픽셀 데이터도 동일하다면 해당 정보는 폐기하게 된다. 폐기 이유는 Event가 발생하였지만 해당 이벤트로는 GUI의 변화를 찾지 못하였기에 폐기를 하게 되는 것이다.

아래의 (그림 8)의 경우 상태는 같고 화면비교 시 픽셀의 변화가 생기게 되었다. 이 경우 화면별 변환 정보에 데이터를 넣고 키 이벤트 또한 작성하게 된다. 해당화면은 현재 데이터에서 다음데이터로 넘어갈 시에 화면변화를 일으키며 키 이벤트 Down 이벤트에 동작한다는 것을 의미하게 된다.



(그림 7) 데이터 비교의 예 (변화 없음)



(그림 8) 데이터 비교의 예 (화면 변화)

(그림 9) 데이터 비교의 예 (상태 변화)

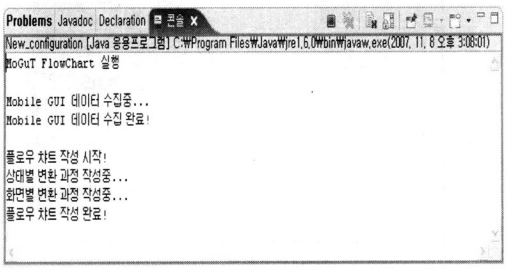
위의 (그림 9)에서 보면 상태 비교 시 상태 값이 변경되었다는 것을 알 수 있다. 이를 통하여 상태별 변환 정보에 데이터를 넣는다. 또한 상태변화에서 비교를 마친다. 화면의 픽셀 비교를 하지 않는 이유는 상태가 변하면 화면 역시 변하기 때문에 이중 처리를 막기 위한 조치가 된다. 저장되는 데이터로 Event가 저장이 되는데 현재 데이터에서 다음 데이터로 변화는 Enter 이벤트를 통하여 변환된다고 할 수 있다.

4.5 화면 구성

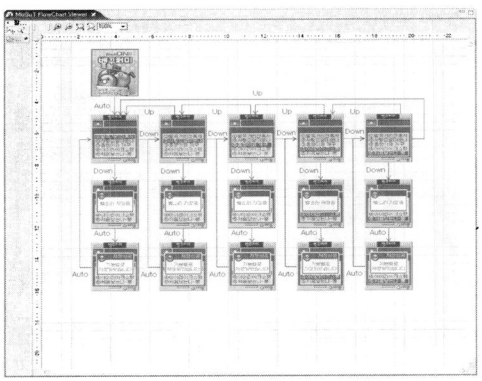
MoGuT Image Flow Maker는 모바일에서 넘어오는 데이터들을 수집, 처리하는 등의 처리 과정을 보여주는 데이터 뷰어와 이미지 플로우를 제작하여서 보여주는 이미지 플로우 뷰어가 존재한다.

먼저 데이터 뷰어를 보면 아래 (그림 10)과 같이 콘솔에 표시가 된다. 표시되는 항목들을 보면 Mobile GUI들의 데이터 수집 상황 표시와 이미지 플로우를 작성에 필요한 변환 과정을 표시하고 있다. 이렇게 수집 및 변환이 끝나게 되면 이미지 플로우용 XML 데이터가 작성되는데 이를 가지고 이미지 플로우를 작성하게된다. 이미지 플로우 작성에는 미리 작업하였던 자료들과의 비교 과정을 거친 후 화면에 표시하게 된다.

Image Flow Viewer는 아래 (그림 11)에서 보는 것과 같다. 가로방향으로는 화면 변화에 대한 이미지 플로우를 나열한 것이고 세로방향은 화면의 상태 변화에 따른 이미지



(그림 10) MoGuT Image Flow의 데이터 수집과 변환

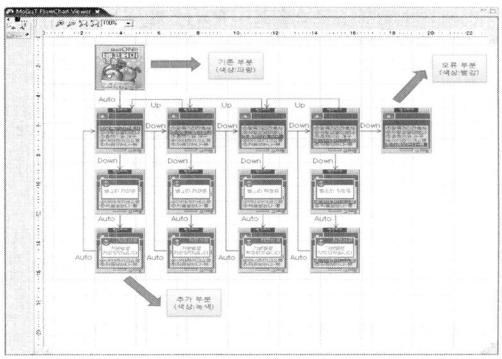


(그림 11) MoGuT Image Flow Viewer

플로우를 나열한 것이다. 이미지 플로우는 가로방향별 보기 및 세로방향별 보기가 가능하고 링크를 통하여 모바일 상에서 해당 화면이 출력되도록 할 수 있다.

이미지 플로우의 비교 기능을 통하여 이전에 제작하였던 이미지 플로우와 비교를 하여 어느 부분이 변경이 되었는지 확인을 할 수 있기 때문에 개발자가 테스트하는 GUI의 오류 판단을 쉽게 해줄 수 있다.

데이터와 이미지의 비교는 데이터는 같은 데이터인지의 간단한 비교를 통하여 체크를 하게 되고 이미지의 경우는 이미지를 픽셀데이터로 변환하여 픽셀의 0xAARRGGBB 값들을 전부 비교하게 된다. 또한 기존 이미지 플로우 데이터와의 비교는 색상에 의해 표시되는데 화면에 대한 것은 아래 (그림 12)과 같다.



(그림 12) 데이터 비교에 의한 색상 표시

기존 부분 (파랑색) : 데이터를 비교하여 기존에 있던 데이터와 같을 경우에는 파랑색으로 표시하여 변경된 것이 없음을 표현한다.

추가된 부분 (녹색) : 데이터의 비교를 마친 후에 기존 데이터에는 없는데 새로 생성된 부분을 녹색으로 표시하여 추가된 것이 있음을 표현한다.

오류 부분 (빨강색) : 데이터를 비교하여 기존 데이터에는 있는데 없어졌거나 픽셀의 변화가 생긴 부분을 체크하여 표시한다.

5. 성능평가

두 가지의 테스트 방법을 사용하여 응용프로그램을 테스트 했을 경우 수행 결과는 다음 <표 2>와 같다.

이 실험의 결과에 따르면 본 논문에서 구현한 MoGuT Image Flow를 이용하면 기존의 테스트 방식인 매뉴얼 방식으로 테스트를 수행하는 것보다 평균 1,024%의 시간을 절약할 수 있다는 것을 알 수 있다.

또 단위화면 별 필요한 테스트 비용에 대한 테스트 수행 결과표는 아래 <표 3>과 같고 표에서 보듯이 단위화면에

대한 비용 또한 9배 정도 감소하는 것을 볼 수 있다.

〈표 2〉 테스트 결과 비교(수행 시간)

(단위 : 평균/second)

프로그램명	메뉴얼 방식	MoGuT Image Flow 방식	효율
BellPack	244	41	595%
GoStop	3,650	264	1,383%
PhotoAlbum	423	45	940%
MusicAlbum	533	52	1,025%
PhoneEmoticon	1,200	102	1,176%
평균			1,024%

〈표 3〉 단위화면 별 테스트 비용

(단위 : second)

프로그램명	화면수	매뉴얼 방식	Image Flow 방식
BellPack	16장	15.25	2.56
GoStop	234장	15.59	1.12
PhotoAlbum	31장	13.64	1.45
MusicAlbum	39장	13.66	1.33
PhoneEmoticon	112장	10.71	0.91
평균		13.77	1.474

6. 결 론

모바일 응용 프로그램은 데스크 탑 응용 프로그램보다 개발 기간과 시장 진입 시간이 단기간이다. 그리고 모바일 응용프로그램에서 가장 많이 이용되고 사용자와 커뮤니케이션이 가장 활발한 부분이 화면이다. 그러나 모바일에서의 GUI 테스트 방법이 체크리스트에 의한 테스트로 테스트 자동화 시스템이 전무한 실정이었다. 그렇기에 본 논문에서는 GUI 테스트에 대한 자동화 도구에 대하여 연구를 수행함으로써 모바일 응용 프로그램의 GUI 테스트에 대하여 소요되는 시간을 감소시키고 GUI에 대한 오류를 쉽게 찾아내고자 하였다. 모바일 GUI 테스트는 개발자 입장에서 오류가 없는 프로그램을 제작하였는지가 핵심이다. 그러나 "오류가 없는 프로그램은 없다" 라는 말처럼 오류에 대하여 빠르게 찾아내는 방법이 필요하고 개발 기간과 시장 진입 시간이 단기간인 모바일 응용 프로그램의 특성상 GUI 테스트의 자동화 도구는 필수적이라고 할 수 있다. 또한 본 프로그램에서는 SK-VM의 환경에서만 테스트가 가능하기 때문에 WIPI환경과 GVM 환경 등등의 환경에서도 테스트를 할 수 있는 MoGuT Framework의 개발이 이루어져야 할 것이다.

참 고 문 헌

[1] Myers, G, Sandler, C., Badgett, T., and Thomas, T., The Art of software Testing, Second Edition, John Wiley & sons, 2004.

[2] IEEE Standard Glossary of Software Engineering Terminology, IEEE std 610.12-1990, The Institute of Electrical and Electronics Engineers, 1990.

[3] Pressman, R., Software Engineering: A Practitioner's Approach, McGraw-Hill, 2003.

[4] 홍준성, "모바일 플랫폼의 기술현황 및 발전방향," 정보과학회지 제22권 21호 통권 제176호, pp.8-14, 2004, 01.

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

[6] Mark Fewster & Dorothy Graham, Software Test automation, Addision-Wesley, 1999.

[7] McGregor, J.D., and T.D. Korson, "Integrated Object-Oriented Testing and Development Process," CACM, Vol.37, No.9, September, 1994.

[8] Patton, R., Software Testing, Sams, 2000.

[9] 한국정보통신기술협회, 소프트웨어테스트 전문기술 기초분야, 한국정보통신기술협회, 2005.

[10] 한국정보통신기술협회, 소프트웨어테스트 전문기술 응용분야, 한국정보통신기술협회, 2005

[11] 권원일. "모바일 소프트웨어 테스팅 현황과 표준적인 테스트 케이스."

[12] NIST, "The Economic Impacts of Inadequate Infrastructure for Software Testing," 2002.5.

[13] 이상윤, 김선자, 김홍남, "한국 무선 인터넷 표준 플랫폼(WIPI)의 표준화 현황 및 발전 방향," 정보과학회지 제22권 21호 통권 제176호, pp.16-23, 2004. 01.

[14] J2ME Documentation, Available at URL : http://java.sun.com/j2me/docs/index.html.

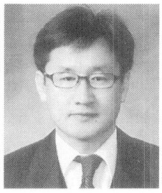
[15] Mark Fewster & Dorothy Graham, Software Test automation, Addision-Wesley, 1999.

[16] "QtUnit(Software development testing framework) Introduction Manual."

[17] 퍼슨넷, "제3자 소프트웨어 테스팅 사례(Case on 3'rd Party S/W testing)," 스텐컨퍼런스 발표.

[18] ISO/IEC, Information Technology-software product quality -Part1: Quality In Use Metrics, ISO/IEC9126, 1998.

[19] Kent Beck, Erich Gamma, "JUnit Cookbook."



황 선 명

e-mail : sunhwang@dju.ac.kr

1982년 중앙대학교 전자계산학과 (이학사)

1984년 중앙대학교 소프트웨어공학전공 (이학석사)

1987년 중앙대학교 소프트웨어공학전공 (이학박사)

2000년~현 재 한국S/W프로세스심사인협회(KASPA) 이사

2000년~현 재 한국정보처리학회 논문지 편집위원

1997년~현 재 ISO/IEC JTC7/WG10 한국운영위원

1998년~현 재 한국정보통신기술협회TTA 특별위원

1989년~현 재 대전대학교 컴퓨터공학과 교수

관심분야 : 소프트웨어 프로세스 모델, 품질 메트릭스, 소프트웨어공학 표준화, 컴포넌트 품질측정, 테스팅 방법론 등

윤 석 진

e-mail : sjyoon@etri.re.kr

1992년 중앙대학교 컴퓨터공학과(학사)

1994년 중앙대학교 대학원 컴퓨터공학과 (석사)

1994년~현 재 한국전자통신연구원 선임 연구원

관심분야 : 객체지향 방법론, Agile 방법론, CASE, 컴포넌트, 재사용