플로우 챠트 기반의 모바일 GUI 테스트 도구에 관한 연구

박상필*, 정일재*, 황선명*,윤석진**
*대전대학교 컴퓨터공학과
**한국전자통신연구원 임베디드 S/W 연구단
e-mail:rose2272@naver.com

A Study on Tools for Mobile GUI Testing Based-on Flow Chart

Sang-Pil Park*, Il-Jae Jung*, Sun-Myung Hwang*, Seok-Jin Yoon**
*Dept of Computer Engineering, Daejeon University
**Embedded S/W Research Division, ETRI

요 약

모바일 어플리케이션 소프트웨어의 시장은 여러 소프트웨어 시장 중 가장 많은 어플리케이션 소프트웨어를 출시하고 있으며 모바일 어플리케이션에서 가장 중요한 사용자와의 정보 교환 수단으로는 GUI가 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 플로우 챠트 기반으로 표시함으로써 GUI를 테스트 하는 방법을 제시하고 테스트 커버리지 까지 측정할 수 있는 방법을 제시한다.

1. 서론

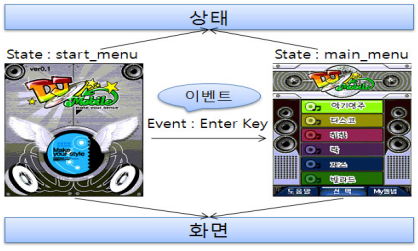
모바일 어플리케이션 소프트웨어의 시장은 여러 소프트웨어 시장 중에서 가장 많은 어플리케이션 소프트웨어를 출시하고 있다. 모바일 소프트웨어는 짧은 개발주기와 짧은 생명주기로 인해 어느 누구보다 먼저 어플리케이션 소프트웨어를 출시해야 소비자를 확보하는 전쟁 아닌 전쟁을 치르고 있다. 모바일 어플리케이션에서 가장 많은 사용자를 확보하고 있는 컨텐츠 분야는 게임, 이미지 등 많은 컨텐츠로 구성 되어져 있다. 이러한 컨텐츠들은 GUI를 기본으로 하여 사용자와의 정보를 교환하는 역할을 한다.

GUI는 Graphics User Interface의 약자로서 사용자가 그래픽을 통해 정보를 교환하고자 하는 대상과 정보를 교환하는 작업환경을 일컫는다. 나날이 변화하는 모바일 어플리케이션 소프트웨어의 GUI는 중요 요소로 자리 잡았고, 모바일 어플리케이션의 중요 테스트 대상으로 보여지고 있다. 하지만 현재 모바일 업체들에서 하는 테스트 방법으로는 테스터가 한단계, 한단계 버튼을 눌러가며 변환되는 화면을 체크하는 방법으로 테스트를 수행하고 있다. 이러한 방법을 탈피하고 GUI 테스트를 자동화 할 수 있는 GUI 테스트 도구를 개발하여야 한다.

따라서 본 논문에서는 플로우 챠트 기반의 모바일 GUI 테스트 기법에 대하여 제시하고자 한다. 2장에서는 모바일 상에서 추출할 수 있는 GUI 모델에 대하여 살펴보고, 3장에서는 플로우 챠트 기반 테스트 도구의 시스템의 구조에 대해야 알아보며, 4장에서는 데이터를 플로우 챠트로 구성하고 테스팅하는 방법에 대하여 제시하고 5장에서 결론을 맺고자 한다.

2. 모바일 상에서 GUI 모델

모바일 상에서 GUI 모델은 화면의 상태(State)와 이벤트(Event) 그리고 화면으로 정의 될 수 있다. 모바일 상에서의 GUI 출력장치는 단일의 LCD로 제한되어 있기 때문에 각각의 정적인 화면을 화면 상태 변수 또는 객체등에 저장하여 이용한다. 이벤트는 일반적으로 사용자가 버튼을 누름으로써 발생되는 이벤트를 뜻한다. 또한 이벤트는 키 이벤트를 통하여 화면의 상태를 변화시킬 수 있기 때문에 기능으로 정의 될 수 있다. 화면은 어떠한 이벤트를 통하여 화면의 상태가 어떠한 상태에 도달하였을 경우에 출력되는 LCD에 출력되는 최종 결과물이다. GUI 모델은 아래 (그림 1)과 같이 표현 할 수 있다.

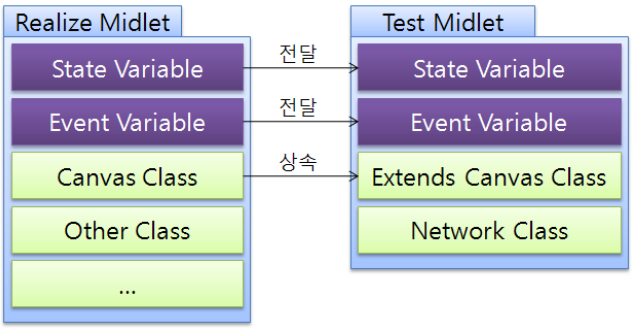


(그림 1) GUI 모델

본 연구는 정보통신부 및 정보통신연구진흥원의 IT신성장동력 핵심기술개발사업의 일환으로 수행하였음. [2007-S032-01, 다중플랫폼지원 모바일 응용 S/W 개발환경 기술 개발]

3. 시스템 구조

3.1 모바일 시스템 구조



(그림 2) 모바일 시스템 구조

모바일 시스템의 구조는 Realize Midlet 과 Test Midlet 의 두 개의 Midlet의 구조로 되어있다. Realize Midlet은 중요 변수인 상태 변수와 이벤트 변수, 화면을 그리는 역할을 하는 Canvas Class 가 있고 그 외에 Sound를 담당하는 클래스나 네트워크 클래스, 로직 클래스등 다양한 클래스들로 구성될 수 있다. Test Midlet의 구조는 상태 변수와 이벤트 변수를 전달받는 부분과 Canvas Class를 상속받는 클래스와 네트워크 클래스로 이루어져 있다. 모바일 시스템의 전체적인 구조는 위 (그림 2)와 같다.

모바일 시스템에서는 개발자가 구현한 코드에 테스트 코드를 삽입 하지 않고 Test Midlet을 따로 두어 실제 Realize Midlet을 상속받아 제작을 하고 이때 paint() 메소드 부분에 아래 (그림 3)와 같이 테스트 수행 코드를 작성한다.

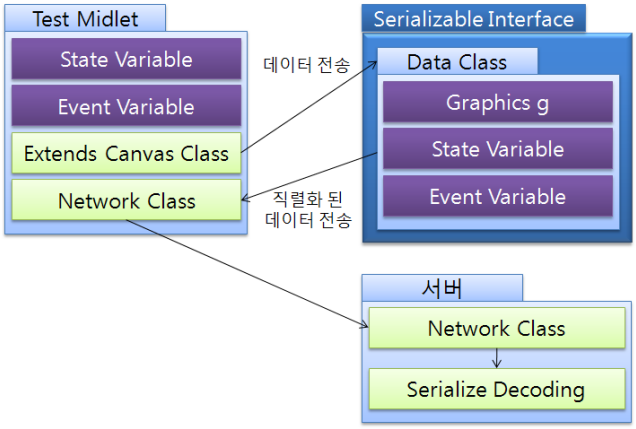
```
public void paint(Graphics g) {
    super.paint(g);

    testStart(g, keyPress, appState);
}
```

(그림 3) Paint() 메소드

이를 통하여 GUI 모델 즉 상태, 이벤트 그리고 화면을 서버로 전송한다.

이때 발생될 수 있는 문제점으로는 상태 변수와 키 이벤트 변수를 담고 있는 변수를 지정해 주여야 한다는 점이다. 프로그래머가 사용하는 키이벤트 저장 변수와 상태 변수의 네이밍 규칙등이 다르기 때문에 Test Midlet 작성시에 해당 변수들의 정보를 입력받아야 한다.

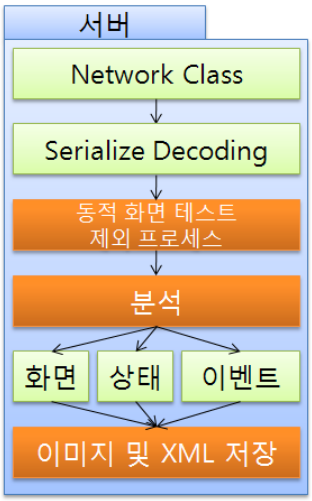
또한 모바일 프로그래밍에서는 객체의 직렬화가 존재하지 않기 때문에 객체 직렬화 클래스를 제작하여야 한다. 모바일 상에서는 Graphics 객체를 이용하는데에 한계가



(그림 4) 직렬화 과정

있기 때문에 모바일 상에서는 분석 및 가공할 수 있는 방법이 없기 때문에 서버로 Graphics 객체를 전송하여 서버 측에서 Graphics 객체를 이용하는 방법을 채택하였다. 객체를 직렬화 하기 위해서는 위 (그림 4)와 같이 직렬화를 하는 인터페이스를 만들고 직렬화 하고자 하는 클래스에 인터페이스를 구현하여 준다. Test Midlet에서 구현한 Extends Canvas Class에서 직렬화를 하고자 하는 데이터를 전송하고 직렬화 클래스에서 직렬화를 한 후에 Network Class에 전달한다. Network Class는 직렬화 된 객체를 서버에 Network를 통하여 전송을 하면 서버측에서는 직렬화된 데이터를 풀어서 원상태의 Graphics g 객체와 상태 변수, 이벤트 변수를 얻는다.

3.2 Server 구조



(그림 5) 서버 구조

Server는 Network Class를 통하여 전송된 직렬화된 데이터를 디코딩하여 원상태의 데이터로 변환한 후 데이터를 동적 화면 테스트 제외 프로세스에 전달한다. 동적 화

면 테스트 제외 프로세스는 테스트를 할 수 없는 부분을 제외하여 분석을 할 수 있도록 한다. 최종 결과화면만을 분석하는 본 논문의 프로그램 특성상 동적으로 변화하는 화면의 경우에는 경우의 수가 기하급수적으로 늘어나기 때문에 정적인 화면만을 테스트 할 수 있도록 한다. 동적 화면 테스트 제외 프로세스에서 제외되고 남은 데이터는 분석을 통하여 상태 비교와 화면 비교를 하여 서버에 해당 데이터를 저장해야 하는지에 대한 판단을 하게 된다. 분석 처리는 아래 (그림 6)과 같다. 분석은 예를 들어 start_menu 상태에서 Enter Key를 눌러야지만 다른 상태 즉 main_menu로 상태 이동을 하게 된다면 다른 키가 눌러졌을 경우 데이터는 동일 데이터가 전송되게 되어서 데이터가 중복되게 된다. 이 경우를 방지하기 위하여 한 단계 전에 들어온 데이터와 전송되는 데이터를 비교하게 된다. 이 경우는 paint() 메소드가 다시 호출되는 상태에서도

(그림 6) 동일 데이터 제거 방법

적용된다.

또한 분석 단계에서 화면비교를 먼저 하는 이유는 상태는 같으나 이벤트에 따라서 화면이 변경될 경우가 있다. 메뉴의 선택이 변경되어 이미지가 다를 경우등의 체크를 위하여 화면 비교를 하고 화면이 같으면 상태를 비교하게 된다. 이벤트의 경우는 전 단계에서 어떠한 이벤트를 통하여 현재의 화면과 상태가 되었는지를 판단하기 위하여 이용된다.

분석 프로세스가 종료되면 데이터를 분할하여 이미지와 XML 데이터로 저장하게 된다. XML 데이터는 아래 <표 1> 과 같다.

<표 1> XML 데이터

```
<data no='1'>
  <image>1.png</image>
  <state>start_menu</state>
  <event>none</event>
</data>
<data no='2'>
  <image>2.png</image>
  <state>main_menu</state>
  <event>Enter Key</event>
</data>
```

또한 플로우 챠트를 작성하기 위해서 분석 단계에서 화면과 상태 정보가 동일하지만 이벤트가 다르게 발생되었을 경우도 체크하게 된다. 데이터 저장 도중 동일 조건의 입력을 방지하기 위하여 수행하는 방법으로 키입력 정보를 상태와 화면의 데이터를 중심으로 하여 아래 <표 2> 와 같이 저장하게 된다. 따라서 다시 해당 상태로 전환 되었을 경우 동일 키를 누르지 않도록 하는 기능을 하게 된다. 단 이전화면으로 되돌아가는 기능으로 많이 사용되는 키인 CLR 키는 마지막에 입력되게 하여 CLR 키 이외의 다른키가 모두 테스트 되었을 경우에 누르도록 한다. 또한 전체 프로그램중 한번이라도 사용되어진 키를 입력하여 테스트 수행시간을 줄일 수 있다.

<표 2> 상태 정보에 따른 키 이벤트 수행 XML

```
<state name='start_menu'>
  <key_enter>yes</key_enter>
  <key_1>no</key_1>
  <key_2>no</key_2>
  <key_3>no</key_3>
  ...
</state>
```

4. 플로우 챠트 테스팅 도구

4.1 플로우 챠트 작성 도구

(그림 7) 플로우 챠트 작성 도구 구조

위 (그림 7)의 플로우 챠트 작성도구에서는 저장된 이

미지 및 XML 문서를 통하여 플로우 챠트의 시작점과 분할 점들을 설정하게 된다. 변화 시점을 체크한 후 상태 변수의 개수를 체크하여 테스트 커버리지를 측정하게 된다. 테스트 커버리지는 아래 <표 3> 과 같다.

<표 3> 테스트 커버리지

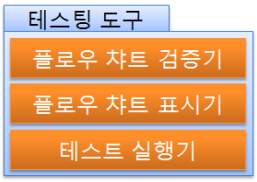
(상태 개수 X 키 개수) / 전체 눌러진 키 개수 * 100

테스트 커버리지 계산을 한 후에 전체적인 플로우 챠트를 작성하여 준다. 플로우 챠트는 아래 (그림 8)과 같다.

(그림 8) 플로우 챠트

4.2 테스팅 도구

테스팅 도구는 플로우 챠트 검증기와 플로우 챠트 표시기 그리고 테스트 수행기로 아래 (그림 9)과 같이 구성되어 있다.



(그림 9) 테스팅 도구 구조

플로우 챠트 검증기는 현재 작성되어진 플로우 챠트에 대하여 어느 부분까지의 테스트가 수행되어졌는지에 대하여 판별한다. 또한 플로우 챠트 표시기는 위 (그림 8)에서의 세로 부분인 상태 변환에 따른 플로우 챠트 표시와 가로부분에 해당하는 화면 변환에 따른 플로우 챠트 표시방법 두가지로 테스터 또는 개발자에게 보여지게 된다. 마지막으로 테스트 수행기는 Test Midlet을 생성하는 기능과 에뮬레이터에서 키 입력을 해주는 기능을 가지고 있다.

5. 결론

본 논문에서는 플로우 챠트를 이용하여 테스트를 수행하고 이를 통하여 테스터 또는 개발자가 플로우 챠트를 확인하여 이벤트에 따른 화면 전환과 상태 전환이 정확하게 일어났는지에 대하여 테스트 할 수 있었다. 그러나 이러한 방법 또한 기존의 방법인 테스터가 눈으로 확인을 하여야 한다는 문제점 존재한다. 또한 프로그램이 커질 수록 테스트를 수행하여야 할 테스트 케이스 수가 (상태 개수 X 키 개수)만큼 늘어나기 때문에 테스트 수행시간 또한 그만큼 늘어나게 된다. 그러나 플로우 챠트 테스트 기법을 도입하면 키입력당의 테스트를 하지 않고 전체적인 수행을 볼 수 있기 때문에 문제가 발생한 부분만을 쉽게 찾을 수 있고 프로그램의 흐름을 쉽게 파악할 수 있다는 이점이 있다. 향후 연구로는 한 방향의 테스트 즉 화면 변환에 대한 테스트만을 테스트 하거나 상태 변환에 대한 테스트만을 테스트하는 방법등의 구체적인 테스트 수행 방법이 필요할 것으로 보인다.

참고 문헌

[1] Pressman, R., Software Engineering: A Practitioner's Approach, McGraw-Hill, 2003.
[2] Patton, R., Software Testing, Sams, 2000.
[3] NIST, "The Economic Impacts of Inadequate Infra-structure for Software Testing", 2002.5
[4] 한국정보통신기술협회, 소프트웨어테스트 전문기술 응용분야, 한국정보통신기술협회, 2005
[5] SQA G. S/W Center, "Manual vs. Automated Test에 대한 사례 연구 소개“, 2003
[6] 권원일. “모바일 소프트웨어 테스팅 현황과 표준적인 테스트 케이스”