한국컴퓨터정보학회 동계학술대회 논문집 제23권 제1호 (2015. 1)

안드로이드 기반의 GUI 플로우 디자이너 구현

조충기*, 윤희용°
*°성균관대학교 정보통신대학
e-mail: cndrldhQk@gmail.com*, youn7147@skku.edu°

Implementation of GUI Flow Designer based on Android Platform

Chung Gi Jo*, Hee Yong Youn°
*°School of Information and Communication Engineering, Sungkyunkwan University

● 요 약 ●

본 논문에서는 안드로이드 애플리케이션 개발을 효과적으로 지원 할 수 있는 GUI Flow Designer for Android(GuiFA) 도구를 구현한다. GuiFA는 안드로이드 프로그램을 구성하는 액티비티들의 흐름을 한 눈에 파악 할 수 있는 기능을 제공할 뿐만 아니라 그것들의 흐름을 구현하는 소스코드를 생성해준다. 이러한 GuiFA의 기능을 사용하여 액티비티들의 흐름을 한 화면에 시각화시켜 효과적인 개발이 가능하고 생성된 코드를 이용하여 효율적으로 애플리케이션을 구현할 할 수 있다.

키워드: GUI, Android Application

I. 서 론

한국 스마트 폰 애플리케이션은 PC 애플리케이션과 비교하여 여러 가지 면에서 차이점을 가지고 있다. 그 중 하나가 GUI (Graphical User Interface)의 구성방식이다. 스마트 폰 애플리케이션은 PC 애플리케이션과 달리 오직 터치 기반의 GUI를 통해 동작한다. 또한 스마트 폰은 PC에 비해서 제한적인 화면 크기를 갖기 때문에 스마트 폰 애플리케이션은 PC 애플리케이션과는 다른 형태의 GUI 구조를 갖는다. 예를 들면 PC 애플리케이션에서는 하나의 화면으로 표현 가능한 사항이 스마트 폰 애플리케이션에서는 화면의 제약으로 인해 불가능할 수도 있다. 이러한 경우 한 화면을 여러 개로 분할하여 애플리케이션의 GUI를 구성해야한다. 이를 위해서는 화면마다 담당하는 기능을 정의하고, 분할된 여러 개의 화면들이 서로 어떻게 연계되어야 하는지도 정의하여야 한다. 따라서 스마트 폰 애플리케이션의 GUI는 한 화면을 구성하는 것뿐만 아니라 전체적으로 연계되는 형상을 파악하는 것도 매우 중요하다.

이를 위해 iPhone을 위한 통합개발환경 XCode에서는 스토리보드(Storyboard)라는 도구를 제공한다[1]. 스토리보드는 애플리케이션을 구성하는 여러 개의 화면들 간의 관계를 표현할 수 있는 기능을 가지고 있다. 이를 통해 GUI를 구성하면 단일 화면의 구조 뿐 아니라 응용을 구성하는 모든 화면들 간의 관계를 한눈에 파악 할 수 있다.

그러나 이와 같은 도구는 안드로이드 환경에선 제공되지 않는다. 현재 가장 널리 사용되는 안드로이드 애플리케이션을 위한 통합개발환경인 이클립스(Eclipse)에서는 오로지 단일 화면의 GUI를 구성하는 기능만을 제공할 뿐 응용 전체를 구성하는 화면들 간의 관계를 한눈에 보여 줄 수 있는 도구를 제공하지 않는다. 이는 GUI의 전체적인 구조를 파악하는데 있어서 어려움을 야기 한다. 본 논문에서는 이와 같은 문제를 해결하기 위해 GuiFA (GUI Flow designer for Android) 도구를 개발한다.

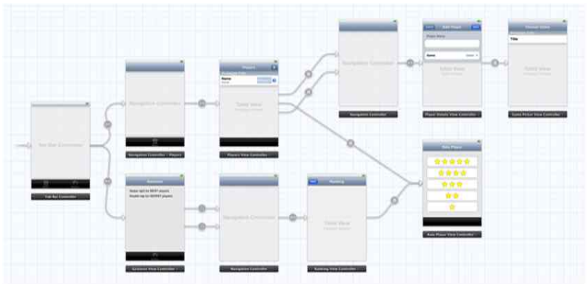

그림 241. XCode의 스토리보드
Fig. 1. Storyboard of XCode

II. 관련 연구

1. 관련연구

1.1. Plug-in

Plug-in 이란 소프트웨어 컴포넌트로 이미 존재하는 소프트웨어에 새로운 기능을 추가하고자 할 때 사용된다[2]. 이클립스에서의 Plug-in은 다양한 측면으로 바라 볼 수 있는데 이클립스 플랫폼 기반 애플리케

이션의 기본 구성 단위이기도하고 시스템에 기능을 추가하는 코드와 데이터의 묶음으로 볼 수도 있다. 이러한 Plug-in을 이용하여 이미 존재하는 개발 환경에 새로운 기능을 추가하거나 새로운 개발 환경을 구축하는 것이 가능하다. 다시 말하면 이클립스가 자체적으로 지원하지 않는 기능이 필요한 경우라면 개발자는 그 기능을 Plug-in이라는 표준화된 구조로 직접 구현하여 그것을 이클립스에 포함시키고 나아가 타 개발자들도 이를 활용할 수 있도록 배포 할 수 있다는 것이다. 이처럼 확장 가능한 표준화된 구조 덕분에 이클립스는 많은 개발자들의 자발적인 기여를 이끌어낼 수 있었고 그 결과 현재 가장 널리 사용되는 개발 플랫폼으로 성장 하였다.

이 논문에서는 안드로이드 개발 환경인 이클립스가 제공하는 기능들은 그대로 유지한 채 Plug-in 형태 로 GuiFA를 구현하여 그것이 제공하는 기능들을 기존의 이클립스에 추가한다.

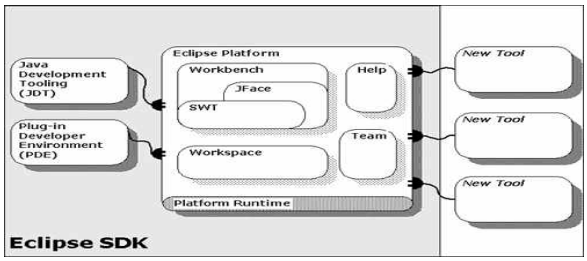

그림 242. 이클립스 플러그인 아키텍처
Fig. 2. Architecture of Eclipse Plug-in

이클립스의 Plug-in을 통한 추가기능 제공은 그림 2와 같은 형태로 이루어진다. 그림 2 에서 볼 수 있듯이 이클립스 자체에 대한 수정 없이 Plug-in을 추가하는 것으로 새로운 기능을 사용할 수 있다. 또한Plug-in 간에도 협업이 가능하기 때문에 기존의 Plug-in을 활용하여 새로운 Plug-in을 구성할 수 있다. 또한 Plug-in의 개발은 이클립스에서 제공하는 Plug-in 개발 환경(PDE)을 사용하면 손쉽게 수행될 수 있다.

1.2 GEF(Graphical Editing Framework)

GEF(Graphical Editing Framework)는 시각 기반의 고품질 에디터를 쉽고 빠르게 만들 수 있게 해 주는 이클립스 프레임워크이다[3]. 트리 다이어그램 또는 플로우 다이어그램같은 다양한 다이어그램들을 위한 시각 에디터를 제작하는 경우에 많이 사용된다. 특히 GEF는 시각 기반 편집기에서 발생 할 수 있는 여러 가지 문제점들을 해결하기 위하여 MVC(Model-View-Controller) 아키텍처[4]에 기초한다.

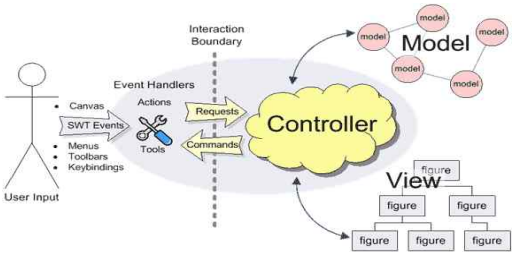

그림 243. MVC 아키텍처
Fig. 3. Architecture of MVC

MVC 아키텍처는 상호작용 시스템을 위한 소프트웨어 아키텍처 중 하나이며 사용자의 인터페이스 변경이 애플리케이션의 핵심 요소에 영향을 주는 문제를 해결하기 위해 제안되었다. MVC 아키텍처에서는 문제 해결을 위해서 핵심 요소와 사용자 인터페이스를 서로 분리하며 이는 모델(Model), 뷰(View), 컨트롤러(Controller)라는 요소로 실현된다. 시각적 요소가 갖는 의미는 모델을 통해 표현되며 시각적 요소에 대한 일반적인 정보(예: 위치, 크기, 색상, 형태 등)는 뷰를 통해서 표현된다. 또한 모델과 뷰 사이의 연산은 컨트롤러를 통해 이루어진다.

따라서 GEF가 제공하는 API와 클래스들을 사용해서 시각 기반 에디터를 구현할 때 이 아키텍처에 맞추어 모델, 뷰, 컨트롤러의 위치 및 역할을 결정해야 한다. 그렇지 않으면 잘못된 애플리케이션이 개발 될 가능성이 커진다.

III. GuiFA의 구현

안드로이드 애플리케이션은 크게 4가지 컴포넌트로 이루어진다. Activity, Service, Content Provider, Broadcast Receiver가 그것들인데 이 중에서 Activity 컴포넌트는 사용자와 애플리케이션 사이의 상호작용을 위해 가시적인 화면을 제공하는 역할을 한다. 예를 들면, 통화 애플리케이션은 시작 시에 전화번호를 입력할 수 있는 화면을 제공하고 번호 입력 후 통화 버튼을 누르면 통화화면으로 전환된다. 그리고 통화가 끝나면 재입력을 선택하거나 종료를 선택할 수 있는 화면으로 돌아간다. 이러한 사용자 화면들은 위에서 언급한 바와 같이 액티비티 컴포넌트로 구성된다. 방금 전의 예처럼 화면이 세 개라면 그 화면들을 위한 액티비티도 각각 하나씩 존재한다. 또한 액티비티들은 특정 작업을 수행하기 위해 다른 액티비티를 호출하여 활성화 시키고 자신은 대기상태로 들어간다. 액티비티들은 이러한 전환을 통해서 사용자에게 적절한 서비스를 제공한다.

GuiFA가 제공하는 가장 중요한 기능이 위에서 언급한 응용을 구성하는 액티비티들의 전환관계를 시각적으로 표현하는 것이다. 이를 위해 GuiFA는 GEF기반의 시각 에디터를 포함한다. 이 에디터를 사용해서 안드로이드 애플리케이션을 구성하는 각각의 화면들과 그들의 흐름을 나타낸다.

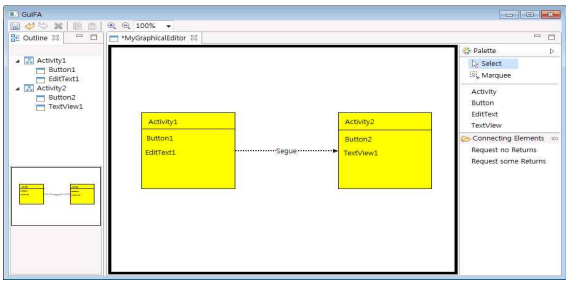

그림 244. GuiFA의 실행 화면
Fig. 4. Execution screen of GuiFA

그림 4는 GuiFA를 사용하여 그린 간단한 안드로이드 애플리케이션의 흐름도이다. 이 애플리케이션은 2개의 액티비티로 이루어져 있고 각각의 액티비티는 버튼과 텍스트 뷰와 같은 시각적인 개체들을

포함한다. 그리고 두 액티비티 사이에 있는 화살표는 그들 사이의 흐름을 나타낸다.

오른쪽에 있는 팔레트에서 그리기 원하는 개체를 선택하여 그것을 에디터 영역에 표현 할 수 있다. 복사, 붙여넣기, 삭제 등과 같은 에디터가 제공해야 하는 필수적인 기능들은 상단의 툴바를 통해 제공된다. 왼쪽에 아웃라인이라는 뷰가 존재하여 에디터에 그려진 개체들을 트리형태로 보여준다. 또한 아웃라인 아래쪽에 작은 뷰는 에디터의 전체적인 시야를 제공해서 사용자에게 편의성을 제공한다.

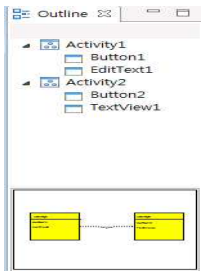

그림 245. 아웃라인
Flg. 5. Outline

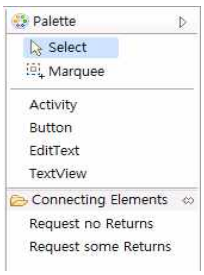

그림 246. 팔레트
Fig. 6. Palette

GuiFA가 제공하는 또 하나의 주요한 기능은 완성된 흐름도로부터 코드를 생성하는 것이다. 사용자는 GuiFA를 이용하여 응용을 구성하는 화면들 간의 관계들을 그린 다음 그에 해당하는 소스코드들을 생성 할 수 있다. 즉, 안드로이드에서 각 화면을 의미하는 액티비티에 대한 코드들과 그 액티비티 사이의 관계를 나타내는 소스코드가 흐름도를 그림으로써 자동으로 만들어 진다. 이렇게 생성된 소스코드들은 XML 레이아웃 파일들과 함께 실제 안드로이드 애플리케이션 개발에서 사용될 수 있다.

그림 7은 그림 4에서 표현된 흐름도로부터 생성된 소스코드 중 Activity1를 위한 부분이다. 그림 4에서 Activity1은 button 1개와 EditText 1개를 가지고 있다. 그림 7의 소스코드를 살펴보면 Activity1을 위한 클래스가 생성되고 button 1개와 EditText 1개를 위한멤버변수가 생성된다. 그리고 클래스의 생성자와 버튼에 의한 이벤트를 처리하는 OnClick함수가 생성된다. 이 함수 내부에 실제 프로그램의 로직에 맞게 버튼에 의한 액티비티 전환을 구현해 주면 된다. 주석으로 제공된 코드는 하나의 예제로 제공한 것이다.

```
public class Activity1 extends Activity implements OnClickListener {
    Button button1;
    EditText edittext1;
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate();
        //setContentView();
        // button1= (button)findViewById(R.id.button1);
        // button1.setOnClickListener(this);
        // edittext1= (EditText)findViewById(R.id.edittext1);
    }

    public OnClick(View v) {
        switch(v.getId()) {
        /* below, one example
         * case R.id.btnedit
         * intent = new Intent(this, ActivityName.class);
         * startActivity(intent);
         * break;
        */
        }
    }
}
```

그림 247. Activity1에 대한 소스코드
Fig. 7. Source code of Activity 1

IV. 결 론

본 논문에서는 안드로이드 애플리케이션 개발을 효과적으로 지원할 수 있는 GuiFA라는 도구를 제작하였다. 기존의 이클립스에서 제공하는 안드로이드 애플리케이션을 위한 GUI관련 툴은 단일화면구성에 관한 기능만을 제공하며 응용을 구성하는 화면들 간의 전환관계를 파악 할 수 있는 방법을 제공하지 않고 있다. 이를 극복하기 위해 GuiFA는 프로그램을 구성하는 액티비티들의 흐름을 한 눈에 파악 할 수 있는 기능을 제공하고 또 그 흐름을 구현하는 소스코드까지 생성해준다. 이러한 GuiFA의 기능을 사용함으로써 액티비티들의 흐름을 하나의 화면에 시각화 시킬 수 있고 이로 인해 보다 더 직관적인 개발이 가능해진다. 또한 자동으로 생성된 코드를 사용하여 효율적으로 애플리케이션을 개발할 수 있다.

ACKNOWLEDGMENT

본 연구는 BK21+사업, 한국연구재단 기초연구사업 (2012R1A1A2040257), (2013R1A1A2060398), 삼성전자(S-2014-0700-000), 미래창조과학부 및 정보통신기술연구진흥센터의 정보통신·방송 연구개발사업 (1391105003)의 일환으로 수행하였음.

참고문헌

[1] iOS Storyboard, *https://developer.apple.com/library/ios/documentation/general/conceptual/Devpedia-CocoaApp/Storyboard.html*

[2] Jeff McAffer, Jean-Michel Lemieux, Chris Aniszczyk, *"Eclipse Rich Client Platform,"* Pearson Education, 2010

[3] GEF, *http://eclipse.or.kr/wiki/Graphical_Editing_Framework*

[4] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal, "*Pattern-Oriented Software Architecture Volume 1: A System of Patterns,*" Willey, 1996.s