중소기업융합학회 논문지 제3권 제2호 pp. 9-17, 2013

안드로이드 어플리케이션의 재사용을 위한 소프트웨어 아키텍처 생성

박진수, 권장진, 홍장의[1*], 최민[2]
[1]충북대학교 컴퓨터과학과, 2충북대학교 정보통신학과

Software Architecture Recovery for Android Application Reuse

Jin-Soo Park, Jang-Jin Kwon, Jang-Eui Hong[1*] and Min Choi[2]
[1]Department of Computer Science, Chungbuk National University
[2]Department of Information and Communication Engineering, Chungbuk National University

요 약 스마트폰의 대중화로 안드로이드 어플리케이션 시장이 급증하였다. 안드로이드 어플리케이션 시장에서 경쟁력을 갖추기 위해서는 높은 생산성, 비용 절감 및 유지 보수가 잘 이루어져야 한다. 또한 어플리케이션의 수요가 높아지면서 짧은 개발 주기가 요구되며 단기간 내에 개발을 진행해야 하므로 개발자는 요구사항 분석 및 체계적인 설계 과정을 생략하는 경우가 많다. 하지만 어플리케이션의 생산성과 비용 절감을 위해 기존 어플리케이션의 전체 또는 일부분을 재사용 할 경우 설계 및 문서화의 생략으로 인해 많은 어려움을 수반되며, 개발 이후 어플리케이션의 유지보수가 제대로 이루어지지 않을 수 있다. 따라서 본 연구에서는 설계 및 문서화가 생략된 안드로이드 어플리케이션의 소스코드를 대상으로 하여 해당 어플리케이션의 아키텍처를 추출하는 리버스 엔지니어링 방법과, 재사용 가능한 기능들을 식별하는 방법을 제안한다. 제안하는 방법을 통해 안드로이드 어플리케이션의 생산성 증가 및 비용감소와 원활한 유지보수가 이루어지는 것을 기대할 수 있다.

Abstract Android applications market has increased rapidly due to the popularity of smart phones. In order to high competitiveness in the application market should be high productivity, reduce cost. And short development cycle is required because of increased the android applications demand. Owing to develop applications in short time, the requirements analysis, design process are able to omitted. But in the case of reuse application at development phase, involved many problems because omit document or design. so target of this paper is android application source code that omit document or design. we propose architecture recovery techniques from android application source code by reverse engineering with identify functions are reused. We expect that increase productivity and reduce development cost, smooth maintain by proposed technique.

Key Words : Android, Application, Reverse Engineering, Reuse, Architecture

1. 서론

스마트 폰은 휴대폰과 개인휴대단말기(Personal Digital Assistant; PDA)의 장점을 결합한 것으로, 휴대폰의 기능에 데이터 통신기능을 통합시킨 것이다. 스마트 폰의 가장 큰 특징은 기존의 모바일 환경의 플랫폼에서 출시되어 주어진 기능만 사용하던 것과는 다르게 다양한 어플리케이션을 사용자가 원하는 대로 설치하고 추가 또는 삭제할 수 있다는 점이다. 사용자가 원하는 어플리케이션을 직접 제작할 수 있고, 같은 운영체제를 가진

이 논문은 정부(교육과학기술부)의 재원으로 한국연구재단-차세대정보컴퓨팅기술개발사업(No. 2011- 0020523)과 기초연구사업(2011-0010396)의 지원을 받아 수행된 것임.
접수일 : 2013년 8월 30일 수정일 : 2013년 9월 22일 게재확정일 : 2013년 10월 10일
*교신저자 : 홍장의(jehong@chungbuk.ac.kr)

스마트 폰 간에 어플리케이션을 공유 할 수 있는 점 등도 기존 휴대폰이 갖지 못한 장점으로 꼽힌다. 이를 위한 운영체제로는 iOS, 안드로이드, 윈도우즈 모바일, 블랙베리 OS 등이 있다. 이러한 특성으로 인해 스마트 폰 어플리케이션에 대한 관심과 관련 산업이 증가하고 있다. 다양한 스마트 폰 중 안드로이드가 가장 많은 관심을 받고 있는데, 그 까닭은 가장 많은 장치와 사용자를 확보하고 있기 때문이다[1].

안드로이드 기반의 어플리케이션은 제약 조건이 없이 누구나 쉽게 개발자로 등록하여 구글에서 제시하는 절차에 따라 쉽게 개발/배포가 가능하다. 또한 안드로이드 어플리케이션은 개발 주기가 짧은 시간에 이루어진다[2]. 따라서 단기간 내에 어플리케이션을 개발하기 때문에 어플리케이션에 대한 요구사항을 분석하고 이를 설계에 옮기는 시간이 생략될 수 있다. 즉, 체계적으로 기술하여 문서화하는 것을 생략될 수 있음을 의미한다.

안드로이드 어플리케이션 시장은 빠르게 급증한다. 2012년 구글 플레이에 등록된 어플리케이션의 수는 1년 사이 56% 증가하여 70만여 어플리케이션이며, 커져가는 어플리케이션 시장에서 경쟁력을 갖추기 위해서는 빠른 개발이 요구된다. 기존의 개발된 어플리케이션의 내용 중 일부분을 새로운 어플리케이션 개발에 재사용될 수 있으며, 재사용성이 높아질수록 어플리케이션에 대한 생산성은 높아진다. 이와 더불어 개발 비용 절감의 효과도 얻을 수 있다.

리버스 엔지니어링(Reverse Engineering) 또는 역공학은 장치 또는 시스템의 기술적인 원리를 그 구조분석을 통해 발견하는 과정이다. 리버스 엔지니어링의 사용 목적은 사라진 문서나 기존 상품의 분석, 악성코드 분석이나 소프트웨어 보안성 테스트 등의 이유가 된다. 또한 유지보수를 위해, 또는 같은 기술 새로운 시스템에 기존의 시스템 일부를 이용하여 개발하기 위해 작동을 분석하는 것 역시 리버스 엔지니어링의 사용 이유가 된다.

안드로이드 어플리케이션의 생산성을 높이고, 개발 비용을 줄이기 위해서 기존 어플리케이션의 재사용을 적용할 수 있다. 안드로이드 어플리케이션 개발은 빠른 개발의 요구에 따라 기술 문서화가 생략될 가능성이 높다. 이때 재사용을 적용하기 위해 기존 어플리케이션에 대한 분석을 수행할 때 소스 코드만을 분석하는 것은 매우 많은 노력이 요구된다. 따라서 우리는 안드로이드 기반의 어플리케이션의 빠른 개발 기간을 위해 기존의 어플리케이션에 대한 재사용성을 높일 수 있는 리버스 엔지니어링을 통한 아키텍처 추출을 제안한다. 제안하는 기법은 기존의 안드로이드 어플리케이션의 소스 코드를 대상으로 분석하여, 내부 구조에 대한 리버스 아키텍처를 생성한다.

2. 관련연구

리버스 엔지니어링은 기존의 프로그램의 이해와 프로그램의 검증, 아키텍처 복구 등을 위하여 연구되고 있다.

Nija Shi[3]의 연구진은 프로그램의 이해도를 위한 도구를 개발하였다. 이 연구의 동기는 기존 프로그램의 이해를 위해서는 UML다이어그램과 같은 설계 산출물로부터 분석이 필요하지만, 만약 이러한 산출물이 없이, 소스 코드를 기반한 프로그램 분석은 대단히 많은 노력이 필요하기 때문에, 해당 코드들을 분석하기 위한 리버스 엔지니어링을 접목한 자동화 분석 도구를 연구하였다. 해당 연구에서는 GoF의 패턴을 재분류하여, 소스 코드를 구조적 관점과 동작 행위 관점에서 프로그램의 의도와 동작의 정적/동적 분석을 제공한다.

mobile application에서 application lifecycle들은 suspended, runing, ready와 같은 절차 관련 상태들과 이들 간의 상태 변화들로 구성되어 있다. 따라서 mobile application lifecycle의 잘못된 구현은 데이터 손실과 같은 많은 문제를 초래할 수 있다. Franke[4]의 연구에서는 application lifecyle을 reverse engineering하는 방법을 제안하고, 주어진 다양한 mobile platform들에 대하여 불완전, 부정확한 부분을 찾는 연구를 진행하였다.

Lungu[5]의 연구에서는 Architecture recovery를 위한 도구인 Softwarenaut를 제안하였다. 제안하는 도구을 통해 소프트웨어 시스템을 계층적으로 분해하고, 이를 이용하여 architectural view의 recovery를 가능하게 하였다.

Sylvain[6]의 연구에서는 대부분의 architecture recovery를 위한 연구들이 자동화 레벨에서 인간의 전문지식을 통합하는데 실패하였다고 언급하였다. 이러한 문제를 해결하기 위해 해당 연구에서는 intentional architecture 개념을 사용하여 기존의 ROMANTIC 접근 방법을 확장하였다. 이를 통해 소프트웨어 아키텍처와 아키텍트의 의도 사이의 적합함을 높이고자 하였다.

소프트웨어 아키텍처의 구성적 재사용은 소프트웨어 분석 도메인에서 어려운 문제 중 하나이다. Xiaojian[7]의 연구에서는 이러한 문제를 해결하기 위하여 reflective 방법과 pi-Calculus를 사용한 아키텍처 재사용 방법을 제안하였다.

3. 재사용 측면의 안드로이드 어플리케이션 요소 식별

안드로이드 어플리케이션에서의 재사용이란 생산성을 높이고, 개발 비용을 줄이기 위해 제공되는 안드로이드 API나 기존 어플리케이션의 전체 또는 일부분을 다시 사용하는 것이다. 따라서 본 연구에서는 재사용 측면에서 안드로이드 어플리케이션의 구성요소를 크게 2가지로 구분하였다. 각 구성요소는 개발된 안드로이드 어플리케이션에서 재사용 가능한 부분을 식별하고, 기능별로 분류하기 위해 사용된다.

3.1 안드로이드 API

안드로이드 API는 개발자가 어플리케이션을 쉽게 개발할 수 있도록 도와주는 클래스 라이브러리의 집합을 의미한다. 개발자는 파일 입출력, 네트워크 등과 같이 복잡하지만 필요한 클래스들이 미리 정의된 API를 응용하여 특정 기능을 개발할 수 있다. 안드로이드 개발 시 API를 사용하기 위해서는 해당 API를 현재 프로젝트에 import시켜야 한다. 따라서 각 클래스에 import된 API들을 식별하고, 동일 API가 import된 클래스들을 묶어 하나의 재사용 가능한 기능으로 정의한다. [표 1]은 안드로이드 API의 몇 가지 예를 보여주고 있다.

3.2 이벤트 리스너

안드로이드에서는 특정 이벤트들을 처리하기 위한 여러 이벤트 리스너(Event Listener)들이 인터페이스로 정의되어 있다. 이벤트란 말 그대로 하나의 사건을 의미하므로 이벤트 리스너를 통해 하나의 기능이 트리거(Trigger)된다고 볼 수 있다. 이벤트 리스너의 종류로는 [표 2]와 같이 예를 들 수 있다.

Table 1. Example of Android API

API	Description
android.location	Google 위치 서비스를 제공해주는 API로써 특정 지역의 좌표 값 등을 반환
android.security	안드로이드 보안 서브시스템의 몇 가지 기능에 대한 접근 허가를 제공
android.widget	응용 프로그램 화면에서 사용할 수 있는 UI요소들이 포함된 위젯 패키지 제공
android.sql	SQL 기반의 database에 접근하기 위한 호환성 인터페이스 제공

4. 리버스에 의한 아키텍처 생성

안드로이드 어플리케이션의 리버스 아키텍처를 생성하기 위해서는 소스 코드 내 클래스와 인터페이스를 식별하여, 해당 클래드들과 인터페이스의 관계를 분석한다. 이때 클래스와 인터페이스는 리버스 아키텍처의 단위 컴포넌트로써 표현된다. 또한 각 클래스와 인터페이스 내부의 멤버 함수와 멤버 변수들을 식별하여, 단위 컴포넌트의 내부 구조를 표현한다. 마지막으로 재사용을 위하여 3절에서 식별한 안드로이드 요소들을 리버스 아키텍처에 접목하여 기능별로 그룹을 표현하고, 각 기능의 시작점을 파악한다.

Table 2. Example of Event Listener

API	Method	Event
OnAttachStateChangeListener	onViewAttachedToWindow, onViewDetachedToWindow	뷰(View)가 윈도우에 탈착 될 때
OnClickListener	onClick	뷰를 클릭할 때
OnLongClickListener	onLongClick	뷰를 오래 클릭할 때
OnCreateContextMenuListener	onCreateContextMenu	컨텍스트 메뉴 생성시
OnDragListener	onDrag	뷰를 드래그할 때
OnFocusChangeListener	onFocusChange	포커스가 한 뷰에서 다른 뷰로 전환될 때
OnKeyListener	onKey	키를 누르거나 뗄 때
OnLayoutChangeListener	onLayoutChange	뷰의 레이아웃 경계가 바뀔 때
OnTouchListener	onTouch	사용자가 화면에 뷰를 터치할 때

4.1 아키텍처 리버스 절차

리버스 아키텍처 생성을 위해서 단위 클래스와 인터페이스를 식별하고 각 클래스들과 인터페이스들 사이의 관계를 분석하는 것이 가장 우선적으로 이루어져야 한다. 단위 클래스들은 리버스 아키텍처의 단위 컴포넌트로 표현이 되며, 리버스 아키텍처의 골격으로써 표현된다. 그 이후, 단위 클래스 내부의 멤버 함수들과 멤버 변수들에 대한 관계를 분석한다. 멤버 함수들과 멤버 변수들은 각 단위 클래스의 내부 구조를 표현하게 된다. [그림 1]은 해당 절차들을 나타낸다.

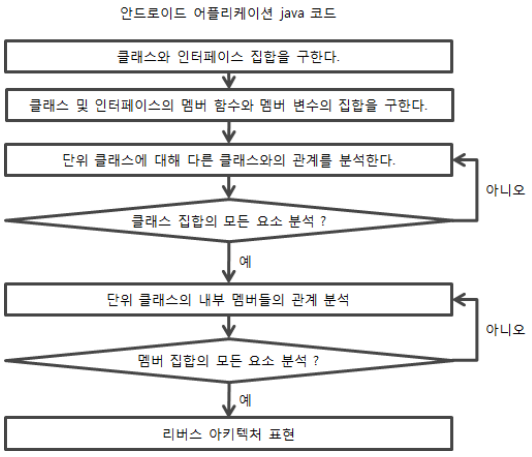


Fig.1. Reverse architecture generation process

[그림 1]의 절차를 보다 자세하게 설명하면 다음과 같다.

(1) 소스 코드에 등장하는 모든 클래스들을 집합 C의 원소로 구한다. 또한 모든 인터페이스들을 집합 I의 원소로 구한다.
(2) 집합 C의 원소 ci의 모든 멤버 함수를 집합 CMi의 원소로, ci의 모든 멤버 변수를 집합 CVi의 원소로 구한다. 또한 집합 I의 원소 ij의 모든 멤버 함수를 집합 IMj의 원소, ij의 모든 멤버 변수를 집합 IVj의 원소로 구한다.
(3) 집합 C의 원소들과 집합 I의 원소들간의 관계를 분석한다.
(4) 집합 CMi의 모든 원소, mij와 집합 CVi의 모든 원소, vij 간의 관계를 분석한다. IMi와 IVi역시 동일한 내용을 수행한다.
(5) 분석된 내용들을 리버스 아키텍처로 표현한다.

4.1.1 단위 클래스 간의 관계

본 절에서는 안드로이드 어플리케이션에서 클래스와 인터페이스들 간의 관계의 특성을 분류하고, 특성을 이용하여 클래스들의 관계를 표현한다. 클래스의 관계는 클래스 간의 관련성이 있는 것을 의미한다. 클래스들의 관계는 포함관계, 상속관계, 연관관계가 존재한다. 클래스들의 관계가 포함관계라면, 예를 들어 A 클래스가 B클래스를 포함한다면, A 클래스의 변경은 B 클래스에 영향을 미치게 된다. 따라서 A 클래스의 객체가 생성, 삭제, 수정, 참조 등이 된다면 B 클래스의 객체에서도 똑같은 연산이 미치게 된다. 따라서 포함관계의 클래스들은 수직적인 관계를 가진다고 정의할 수 있다. 이는 상속관계에서도 동일한 근거로 정의될 수 있다. 만약 A 클래스와 B 클래스가 연관관계를 이룬다면, A 클래스와 B 클래스의 사이에 메시지 호출이 이루어지며, 이것은 수평적 관계를 가지게 된다. 다시 말해 A 클래스나 B 클래스가 상대 클래스의 객체를 생성하고 삭제, 수정 그리고 참조 등을 포함한 메시지 호출에 의해서 이루어진다. 따라서 연관관계의 클래스들은 수평적인 관계를 가진다고 정의할 수 있다. 이를 [그림 1]의 절차 (1)에서 식별된 클래스와 인터페이스 집합의 원소들의 관계를 모두 적용하여 서로의 관계성을 분석한다.

```
public class FileIOActivity extends Activity {
/** Called when the activity is first created. */
  public void onCreate(Bundle savedInstanceState) {
  protected void onDestroy() {
  private boolean FileSave(int Num,String Date,String
Data) {
  private File FileOpen(int Num,String Date) {

  Handler m_finish = new Handler() {

  @Override
  public void handleMessage(Message msg) {
  // TODO Auto-generated method stub
  switch(msg.what) {
    case 0:
      context.finish();
      break;
    case 1:
      Intent intent = new
      Intent(context,TrailSavedTapActivity.class);
      intent.putExtra("IsOnlySearch",true);
      startActivity(intent);
      context.finish();
      break;
    }
  }
};
```

Fig. 2. Example for source code of FileIOActivity

[그림 2]의 액티비티 클래스 FileIOActivity의 예제는 내부에 Handler 클래스를 포함하고 있다. 따라서

FileIOActivity 클래스와 Handler 클래스는 포함관계를 이루며, handleMessage 함수가 TrailSavedTabActivity를 생성한다. 이를 표현하면 [그림 3]과 같다.

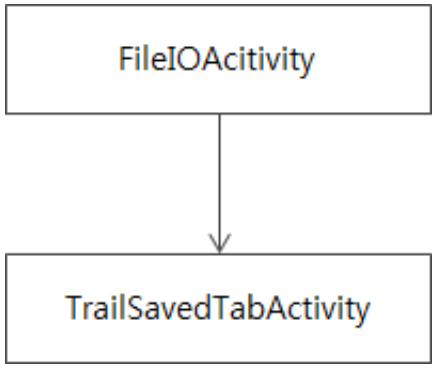


Fig. 3. Relationship between unit classes

[그림 3]에서 FileIOActivity는 TrailSavedTabActivity를 호출하므로 수평적인 관계를 가지게 된다. [그림 2]를 살펴보면 Handler 클래스의 경우, FileIOActivity에 포함된 포함관계를 가지며, 두 클래스는 수직적인 관계를 형성한다. [그림 3]의 추상화 레벨을 낮추어서 표현하면 [그림 4]와 같이 표현된다.

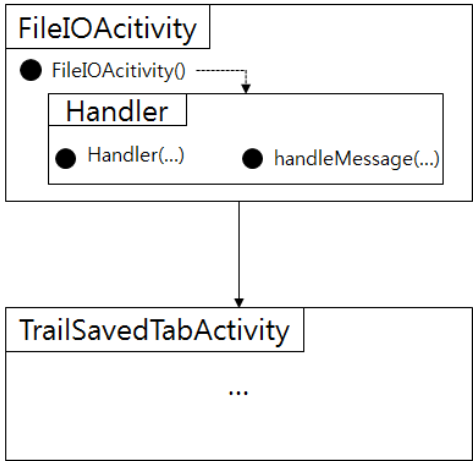


Fig. 4. Vertical relationship of FileIOActivity class

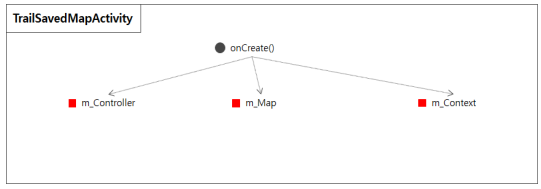
4.1.2 멤버 함수와 멤버 변수의 관계

멤버 함수와 멤버 변수들의 식별이 이루어 졌다면, 멤버 변수의 집합의 원소에 접근하는 모든 멤버 함수의 집합의 원소들을 파악한다. 멤버 함수의 멤버 변수에 대한 접근은 해당 멤버 변수의 생성, 수정, 참조로 파악할 수 있다.

```
public class TrailSavedMapActivity extends
MapActivity{
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
setContentView(R.layout.saved_trailallroute_map_layo
ut);
    m_Context = this
    m_Map =
    (MapView)findViewById(R.id.saved_mapview);
    m_Controller = m_Map.getController();
    ...                    ...
```

[Fig.5] onCreate member function of TrailSavedMapActivity

[그림 5]와 같은 액티비티 클래스의 멤버 함수 onCreate를 가정할 때, m_Context, m_Map, m_Controller 멤버 변수의 수정이 onCreate함수 내부에서 이루어지므로 onCreate함수는 세 멤버 변수에 대해 접근한다고 정의할 수 있다. 이때 멤버 함수와 멤버 변수의 표현은 [그림 6]와 같다.



[Fig. 6] Relation represantation of TrailSavedMapActivity s Member variable and function

[그림 6]와 같이 식별된 멤버 변수는 붉은 색의 작은 네모와 변수명을, 멤버 함수는 검은 색의 원과 함수명으로 표현한다.

4.2 재사용을 위한 기능별 클래스 분류

4.1 절의 절차에 따라 리버스 아키텍처가 생성되었다면, 3절에서 식별한 요소들을 기준으로 각 단위 클래스들을 동일한 기능을 수행하기 위한 그룹으로 분류한다. 3절에서 식별한 API들은 소스코드에서 import 되며, 클래스를 개발하기 위해 import된 API들이 같은 클래스들이 해당 API를 통한 동일한 기능의 목적을 가진다고 할 수 있다. [그림 7]은 각각의 소스 파일에 import된 API들을 나타낸다. [그림 7]의 왼쪽의 소스 코드의 import 리스트에서 maps의 API가 [그림 7]의 오른쪽 소스 코드들의

import 리스트에 모두 존재함을 확인할 수 있고, 이를 통해 세 소스 코드는 동일한 기능의 목적을 가지는 것으로써 두 소스 코드에 해당하는 클래스들을 같은 그룹으로 분류할 수 있다.

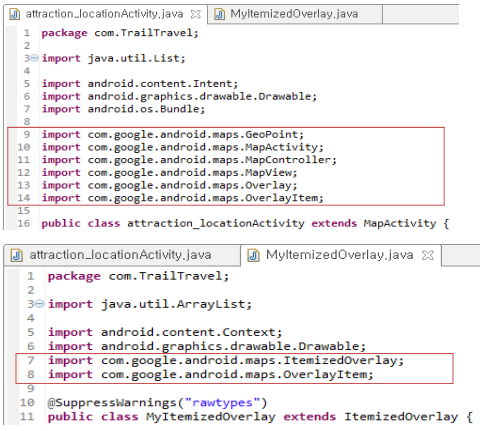


Fig. 7. Class group using same API

어플리케이션의 소스코드에 존재하는 이벤트 리스너들은 사용자의 조작이나 H/W 센스의 값의 변화 등의 외부 이벤트를 대기하고 해당 이벤트가 발생되었을 때 지정한 액션을 취하게 된다. 해당 액션은 함수 및 클래스의 객체 생성일 가능성이 높다. 따라서 소스 코드의 이벤트 리스너들을 분석하면, 클래스 그룹에 해당하는 기능의 트리거 역할을 하는 리스너를 찾을 수 있다. 즉, [그림 7]의 경우 구글의 map에 해당하는 기능을 발동시키는 리스너부터 클래스 그룹을 통해 map에 관련된 소스 코드의 묶음을 파악하는 것이 용이하다.

5. 적용 및 분석

본 절에서는 4절에서 제안한 리버스 아키텍처 생성 방법을 예제 시스템에 적용하고 적용된 내용을 분석하는 내용을 담고 있다.

5.1 예제 어플리케이션 정의

본 절에서는 4장에서 제시한 리버스 아키텍처 생성 과정에 대하여 블랙박스 안드로이드 어플리케이션 예제 시스템[8]에 적용하고, 생성된 리버스 아키텍처를 통해 어플리케이션의 구조를 확인 할 수 있었다.

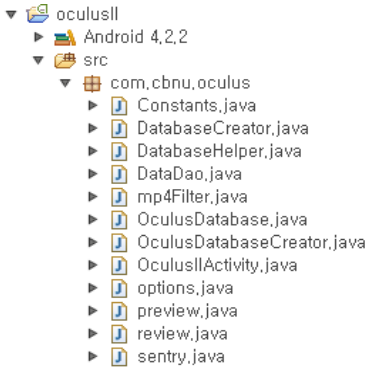


Fig. 8. Composition of source codes of example system

Table 3. Relation Table of Unit classes

단위 클래스	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
OculusIIActivity(1)		H	H	H												
sentry (2)					H	H					V	V	V			
options(3)						H										
review(4)							H									
preview(5)																
DataDao(6)								H							V	
mp4Filter(7)																
DatabaseHelper(8)									H	H						
OculusDatabaseCreator(9)										V						
DatabaseCreator(10)																
Handler(11)																
BroadcaseReceiver(12)																
CountDownTimer(13)																
DataTable(14)														V		
DataTo(15)																
OculusDatabase(16)																

[그림 8]은 예제 어플리케이션의 소스 코드의 구성을 나타낸 것이다. 구체적인 어플리케이션의 요구사항은 다음과 같다.

(1) 어플리케이션은 사용자의 전방을 촬영하고 영상을 저장한다.
(2) 어플리케이션은 사고 감지를 지속적으로 수행한다.
(3) 사고가 감지되었을 때 사전에 기록된 응급 전화번호로 사고 전파를 수행한다.
(4) 사고 전파는 SMS 발송과 Emergency Call 기능을 수행한다.

[그림 8]과 같이 소스 코드는 모두 12개의 파일로 구성되어 있으며, 15개의 클래스와 1개의 인터페이스를 확인할 수 있다. 또한 모든 단위 클래스들에서 103개의 멤버 함수들과 76개의 멤버 변수들이 확인되었다. 리버스 아키텍처 생성 절차에 따라 식별된 클래스, 인터페이스, 멤버 함수, 멤버 변수들간의 관계를 분석을 하였고, 해당 결과는 각각 [표 3]과 [표 4]에 해당한다.

5.2 예제 어플리케이션의 단위 클래스 분석

[표 3]을 보면 H와 V로 각 단위 클래스들의 관계를 나타내었다. H는 수평적 관계로 함수의 호출이나 객체의 생성 등을 예로 들 수 있으며, V는 수직적 관계로 포함관계나 상속관계를 예로 들 수 있다. [표 3]의 내용을 통해 [그림 9]와 같이 아키텍처의 골격을 생성할 수 있다.

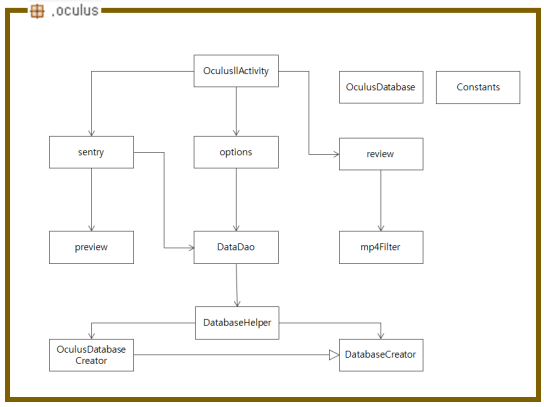


Fig. 9. Skeleton of reverse architecture

Table 4. Analysis result relationship between member function and variable for each class

class : OculusIIActivity					
	optionBtn	driveBtn	exitBtn	replayBtn	
onCreate()	√	√	√	√	
OculusIIActivity()					
...					
class : DataDao					
	db	classname	id	DataText	TypeText
getSensor()	√				
getPhoneNum()	√				
close()	√				
update(...)	√	√			
delete(...)	√	√			
insert(...)	√	√			
get()	√	√			
getId()			√		
setId(...)			√		
toString()			√	√	√
getDataText()				√	
setDataText(...)				√	
getTypeText()					√
setTypeText(...)					√
class : options					
	sBar	cnt	saveBtn	nText	savenum
onCreate()	√		√	√	√
onClick()				√	√
OculusIIActivity()					
onPause()					
onResume()					
OnSeekBarChangeListener()		√			
...					

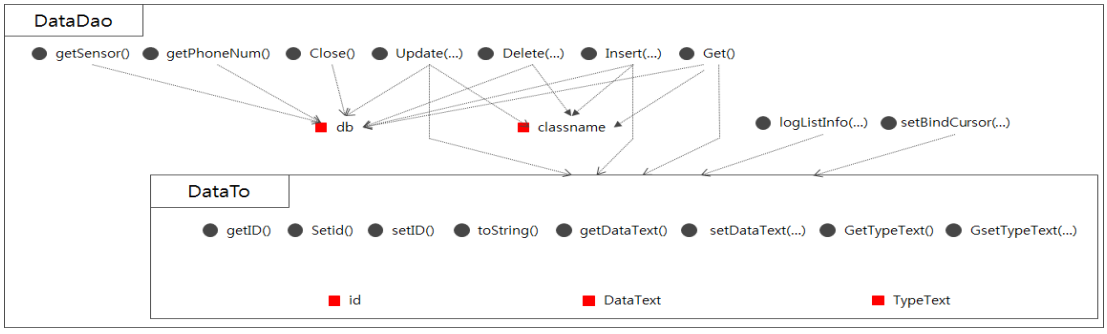


Fig. 10. Internal Architecture of DataDao class

5.3 예제 어플리케이션의 단위 클래스 분석

[표 4]는 각 단위 클래스 별 멤버 함수의 멤버 변수에 대한 접근을 나타내었다. 표시된 변수는 해당 함수로부터 접근된다. [그림 10]는 [표 4]를 기반으로 단위 클래스 DataDao에 대한 세부 리버스 아키텍처를 표현한다.

5.4 동일 기능 클래스 그룹화

[그림 9-10] 과 같이 리버스 아키텍처가 도출되고 나면, 소스 코드의 API정보들을 이용하여 각 단위 클래스들의 그룹을 만들 수 있다. OculusDatabase와 DatabaseHelper, DataDao는 모두 database API를 import하여 개발되었다. 따라서 세 클래스는 DB의 접근이라는 기능의 그룹을 만들 수 있다.

OculusDatabaseCreator는 database API를 직접적으로 호출하진 않지만, DatabaseCreator 클래스를 import하고, DatabaseCreator를 상속받는 클래스이므로 해당 그룹에 포함될 수 있다. [그림 11]은 이러한 방법으로 예제 어플리케이션의 아키텍처에 같은 기능을 가진 클래스 그룹을 표현하였다.

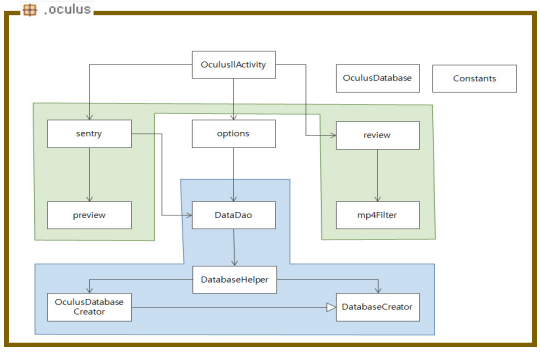


Fig. 11. Unit class groups having similar functionality

[그림 11]에서 파란 그룹은 DB 기능을 수행하는 클래스 그룹이며, 녹색 그룹은 영상을 수행하는 클래스 그룹임을 밝힌다.

6. 결론

기존 안드로이드 어플리케이션 개발 과정에서 설계 및 문서화의 생략이 안드로이드 어플리케이션 재사용 및 유지보수 과정에서 어려움을 초래할 수 있다. 본 연구에서는 이러한 문제점을 해결하기 위해 2가지 방법을 제안하였다. 첫 번째는 리버스 엔지니어링을 통해 안드로이드 어플리케이션의 아키텍처를 추출하는 방법이었으며, 두 번째는 추출된 아키텍처에서 재사용 가능한 기능들을 식별 및 분류하는 방법이었다. 커져가는 안드로이드 시장에서 경쟁력을 갖추기 위해서는 높은 생산성 및 비용 절감이 필요하다. 따라서 제안하는 방법을 통해 아키텍처를 추출하고 이를 통해 개발된 어플리케이션의 이해도를 높일 수 있으며, 재사용 가능한 기능들을 분류함으로써 개발 생산성 및 비용 절감이 향상하는 것을 기대할 수 있다. 제안한 방법을 효율적으로 사용하기 위해서는 자동화가 요구될 것이다. 따라서 향후 연구에서는 제안한 방법들을 활용하여 자동화 도구를 개발하는 연구를 진행하고자 한다.

참 고 문 헌

[1] Gartner, “Gartner Says Worldwide Smartphone Sales Soared in Fourth Quarter of 2011 With 47 Percent Growth,” http://www.gartner.com/it/page.jsp?id=1924314, February 2012.

[2] S. M. Hwang, J. J. Kim, “A GUI Testing Method base on Scenario for Mobile Application Software,” Journal of the Korea Academia-Industrial cooperation Society, Vol.9, No.3, pp.681-689, 2008.

[3] SHI Nija, OLSSON, Ronald A., “Reverse engineering of design patterns from java source code,” In: Automated Software Engineering, 2006. 21st IEEE/ACM International Conference on. IEEE, pp. 123-134, 2006.

[4] Franke, Dominik, et al. "Reverse engineering of mobile application lifecycles." Reverse Engineering (WCRE), 2011 18th Working Conference on. IEEE, pp. 283-292, 2011.

[5] Lungu, Mircea, Michele Lanza, and Oscar Nierstrasz. "Evolutionary and collaborative software architecture recovery with Softwarenaut." Science of Computer Programming, page to appear, 2012.

[6] Chardigny, Sylvain, and Abdelhak Seriai. "Software architecture recovery process based on object-oriented source code and documentation." Software Architecture. Springer Berlin Heidelberg, vol. 6285, pp. 409-416, 2010.

[7] Xiaojian, Li, and Zheng Ying. "Towards Compositional Reuse for Software Architecture." Affective Computing and Intelligent Interaction. Springer Berlin Heidelberg, vol. 137, pp. 651-659, 2012.

[8] Jin-Soo Park, Doo-Hwan Kim, Jang-Eui Hong, "Vehicle Blackbox application based on Android Platform," Journal of IT Convergence Society for SMB, Vol.1, No.1, pp.71-74, 2011.

[9] Frakes, William B., and Kyo Kang. "Software reuse research: Status and future." Software Engineering, IEEE Transactions on, vol. 31, Issue 7, pp. 529-536, 2005.

[10] Sang-Hwan Kung, "Architecture Based Design Methodology for Smart Phone Applications," Journal of IT Convergence Society for SMB, Vol.1, No.1, pp.111-114, 2011.

저 자 소 개

박 진 수(Jin-Soo Park) [정회원]

- 2012년 2월: 충북대학교 컴퓨터공학부 학사
- 2012년 3월 ~ 현재 : 충북대학교 컴퓨터과학과 석사

<관심분야> : 소프트웨어 공학, 소프트웨어 아키텍처, 소프트웨어 테스팅 등

권 장 진(Jang-Jin Kwon) [정회원]

- 2012년 2월: 충북대학교 컴퓨터공학부 학사
- 2012년 3월 ~ 현재 : 충북대학교 컴퓨터과학과 석사

<관심분야> : 소프트웨어 공학, 소프트웨어 아키텍처, Safety Critical System 등

홍 장 의(Jang-Eui Hong) [정회원]

- 2001년 2월: KAIST 전자학과 공학박사
- 2001년 2월 ~ 2002년 9월: 국방과학연구소 선임연구원
- 2002년 10월 ~ 2004년 8월: (주) 솔루션링크 기술연구소장
- 2004년 9월 ~ 현재 : 충북대학교 소프트웨어학과 교수

<관심분야> : 소프트웨어 품질공학, 모델기반 검증 분석, 소프트웨어 아키텍처, 소프트웨어 프로세스 개선 등

최 민(Min Choi) [정회원]

- 2008년 2월: KAIST 전자전산학과 전산학전공 박사
- 2008년 3월 ~ 2010년 2월 : 목삼성전자 반도체총괄 메모리사업부 책임연구원
- 2010년 3월 ~ 2011년 8월: 원광대학교 공과대학 컴퓨터공학과 조교수
- 2011년 9월 ~ 현재 : 충북대학교 전자정보대학 정보통신공학부 조교수

<관심분야> : 컴퓨터구조, 임베디드 시스템