Journal of the Korea Academia-Industrial cooperation Society
Vol. 15, No. 7 pp. 4500-4507, 2014

http://dx.doi.org/10.5762/KAIS.2014.15.7.4500
ISSN 1975-4701 / eISSN 2288-4688

모바일 앱 재사용성 측정을 위한 메트릭 개발

조은숙[1], 김철진[2*], 송치양[3]
[1]서일대학교 컴퓨터소프트웨어과, [2]인하공업전문대학교 컴퓨터시스템과
[3]경북대학교 소프트웨어과

Development of Metrics to Measure Reusability of Mobile App.

Eun-Sook Cho[1], Chul-Jin Kim[2*], Chee-Yang Song[3]
[1]Dept. of Computer Software, Seoil University
[2]Dept. of Computer System, Inha Technical College
[3]Dept. of Software, Kyungbuk University

요 약 스마튼 폰 사용자들의 수가 기하급수적으로 증가함에 따라 향후 모바일 시장의 규모는 엄청나게 거대해질 것으로 전망되고 있다. 이러한 성장세에 맞춰 많은 형태의 모바일 앱들이 개발되고 있다. 그런데 문제는 개발된 앱들에 대한 재사용성, 확장성 등과 같은 품질에 대한 검증이 제대로 이루어지고 있지 않다. 특히 유료 앱의 경우, 사용자들로 하여금 많은 불만의 요소들이 생기고 있는 실정이다. 그래서 본 연구에서는 모바일 앱 개발 과정에서 모바일 앱의 품질을 측정할 수 있는 메트릭을 개발하고자 한다. 이를 통해 모바일 앱에 대한 품질 검증이 이루어짐으로써 사용자들로 하여금 모바일 앱에 대한 만족도를 향상시킬 수 있을 것이라 기대한다.

Abstract As the number of smart phone users is increasing exponentially, the scale of the future mobile market is expected to be very large. Many mobile applications are developing according to this growth trend. On the other hand, there has been little verification of the quality, such as the reusability or extensibility of the developed mobile applications. In particular, many users have expressed their dissatisfaction in the case of pay mobile applications. Therefore, developed to measure the quality of mobile applications in developing mobile applications. Overall, the degree of user satisfaction can be improved by realizing the quality verification of mobile applications by applying the proposed metrics.

Key Words : Mobile Application., Reusability, Extensibility, Quality Verification

1. 서론

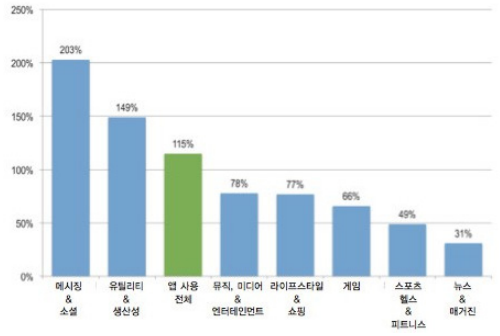
모바일 앱(Mobile App)이란 스마트폰, 태블릿PC 등과 같이 개인이 휴대하거나 이동하면서 인터넷을 비롯한 다양한 데이터나 영상, 음성 정보 등을 송수신 할 수 있는 모바일 기기에 적합하게 디자인되어, 모바일 기기 자체의 기능을 확장 및 향상시키는 소프트웨어를 의미한다. 이러한 모바일 앱은 모바일 디바이스에 애플리케이션을 설치하여 사용하는 프로그램으로 사용자와의 밀착성이 높고, 빠르며, 개인에 특화된 서비스 이용이 가능한 특성을 지니고 있다.

Fig. 1에 나타난 2013년 Flurry가 모바일 앱 이용 상황의 분석 결과 발표에 따르면 모바일 이용 규모는 전년도에 비해 2.1배(115%) 증가했으며, 카테고리에서는 메시지와 소셜이 203% 증가로 전체를 견인하고 있다[1].

이와 더불어 안드로이드 마켓에는 매주 수 천 개의 새로운 앱이 출시되고 있다. 이로 인해 앱의 혼잡현상을 헤쳐 나갈 사전 대책이 점차 중요해지고 있다. 이러한 대책

본 논문은 2013년 서일대학교 교내연구과제로 수행되었음.
*Corresponding Author : Chul-Jin Kim(Inha Technical College)
Tel: +82-32-870-2338 email: cjkim@inhatc.ac.kr
Received April 14, 2014 Revised June 11, 2014 Accepted July 10, 2014

방법의 한가지 방법은 적절한 타깃팅의 모바일 광고와 앱 연동 프로모션을 활용하는 것이다.



[Fig. 1] Status of App. Usage by Category

현재 많은 앱들이 이 방법을 채택하여 앱 스토어에 출시하고 있다. 그러나 '임프레션-설치-랭킹'이라는 앱의 소비 사이클을 가속화하기 위한 증명된 또 다른 방법이 하나 있다. 그것은 바로 앱의 품질을 높이는 것이다. 좋은 앱일수록 앱의 수명 기간이 길다. 즉, 품질이 높은 앱일수록 일반적으로 사용자 평가, 랭킹, 다운로드, 설치기간에서 높은 점수와 연결된다. 고품질의 앱은 안드로이드 마켓이나 소셜 미디어에 소개되는 것과 같은 예기치 않은 주목을 받을 가능성도 그만큼 높다. 본 논문에서는 이처럼 모바일 앱 개발에 있어서 모바일 앱의 품질을 측정하기 위한 메트릭들을 제안한다. 특히 모바일 앱의 여러 가지 품질 요소 가운데 재사용성 측정을 위한 메트릭을 개발한다.

본 논문의 구성으로 2장에서는 관련연구로 안드로이드 플랫폼과 특성 및 재사용성 측정과 관련된 기존 연구에 대해 설명하며, 3장에서는 본 논문에서 제안하는 모바일 앱의 재사용성 측정을 위한 메트릭을 제안한다. 4장에서는 3장에서 제안한 기법을 실제 사례에 적용한 결과를 제시한다. 마지막으로 5장에서 결론 및 향후 연구 과제를 제시한다.

2. 관련 연구

2.1 안드로이드 플랫폼

그동안 여러 스마트 폰 운영체제들이 개발되어 오고 있지만 현재 대부분의 스마트 폰에 내장되어 있는 플랫폼으로는 애플의 iOS와 구글의 안드로이드 플랫폼이 주류를 이루고 있는 상황이다[2,3]. 안드로이드는 구글에서 만든 모바일 기기를 위한 플랫폼으로서, 운영체제, 미들웨어 및 주요 응용 프로그램을 포함하는 모바일 기기용 소프트웨어 모음을 의미한다[2].

안드로이드는 다음과 같은 기술적 특징들을 지니고 있다.

■ 개방성

안드로이드는 개방형 리눅스 커널을 기반으로 하고 있다. 나아가, 그것은 메모리와 하드웨어 리소스를 모바일 환경에 최적화시킨 맞춤형 버추얼(가상) 기기를 사용하고 있다.

■ 다양한 애플리케이션의 동등성

안드로이드는 전화기의 기본 애플리케이션과 부가 애플리케이션을 구별하지 않는다. 다양한 종류의 애플리케이션과 서비스를 사용자에게 제공하는 전화기의 성능에 동일한 접속이 가능하도록 설계되었다.

■ 상호 호환 및 연동성

안드로이드는 새롭고 혁신적인 애플리케이션에 대한 경계를 없앴다. 예를 들면, 좀 더 사용자 중심적인 사용감을 제공하기 위해서 개발자는 웹으로부터 정보를 가져와 사용자의 연락처들, 캘린더, 또는 지리적 위치 정보 같은 개개인의 모바일 폰에 있는 데이터와 병합할 수 있다.

■ 애플리케이션 개발의 용이성

안드로이드는 풍부한 애플리케이션 설계를 위한 다양하고 유용한 라이브러리와 툴을 제공한다. 한 예로, 안드로이드는 개발자가 기기의 위치를 알게 하고, 기기 간에 통신할 수 있게 하여 뛰어난 피어-투-피어(Peer-to-Peer) 소셜 애플리케이션(Social Application)을 구현한다. 덧붙여, 안드로이드는 개발자들에게 높은 생산성과 그들의 기기에 대한 깊은 통찰력을 주는 플랫폼을 기반으로 개발된 통합 툴세트를 포함하고 있다.

2.2 재사용성 측정

재사용성을 측정을 위한 방법들에 대한 관련 연구로 경험적 방법에 의한 방법과 정성적 방법에 의한 방법에 대해 살펴봄으로서, 기존 연구에서의 특징과 한계점들을

제시한다.

2.2.1 경험적 방법에 의한 재사용성 측정

소프트웨어의 재사용성을 측정하기 위한 방법으로 크게 경험적 방법과 정성적 방법으로 구분한다[4,5]. Prieto-Diaz와 Freeman[6]은 재사용성의 측정에 사용되는 다섯 가지 요소들을 나열하였다. 프로그램의 사이즈, 프로그램의 구조, 프로그램의 문서화, 프로그래밍 언어, 그리고 재사용 경험이 소프트웨어의 재사용성의 측정에 사용된다고 보았다. Caldiera와 Basili[7]는 소프트웨어의 재사용성은 소프트웨어의 정확성(Correctness), 가독성(Readability), 테스트가능성(Testability), 수정용이성(Ease of Modification), 그리고 성능(Performance)에 의하여 좌우가 된다고 보았다. 그러나, 이러한 요소들은 직접적인 측정이 어렵기 때문에 그들은 다음과 같은 방법으로 측정을 하였다. Halstead의 프로그램의 크기(Program Volume)를 이용하는 방법, McCabe의 Cyclomatic 복잡도를 이용하는 방법, 정규도(Regularity), 그리고 재사용 빈도수에 의하여 소프트웨어의 재사용성을 측정 하였다[8]. 그러나 이러한 방법은 소프트웨어 재사용성에는 쉽게 적용되지만, 단위 컴포넌트의 재사용성이나 프레임워크의 재사용성을 평가하는데 있어서는 컴포넌트와 프레임워크의 특징인 가변성, 인터페이스, 메소드, 속성 등과 같은 요소들이 반영되어 있지 않아서 그대로 적용하는데 한계성이 있다.

2.2 정성적 방법에 의한 재사용성 측정

재사용성은 소프트웨어의 품질 측면에서 상당히 중요한 부분을 차지한다. 따라서, 소프트웨어의 품질을 향상시킬 수 있는 요소들이 역시 재사용성을 향상시킬 수 있는 요소가 될 수 있다[8,9]. 정성적인 측정 방법은 개인적인 기준에 의해서 측정하는 방법을 말한다. 따라서 측정결과가 다소 주관적인 결과가 나올 수 있다.

ISO 9126[10]은 소프트웨어의 품질 측정에 관한 모델을 제시하고 있다. ISO 9126은 소프트웨어의 품질을 6개의 특징(Characteristics)으로 나누어서 측정하며, 그 각각의 특징에는 하위 특징(Sub Characteristics)들이 존재한다[11]. 이러한 하위 특징들은 내부 측정기준(Internal Metrics)과 외부 측정기준(External Metrics)으로 측정을 한다. ISO 9126은 소프트웨어의 일반적인 품질 측정에 관한 모델은 제시하고 있지만 컴포넌트와 관련된 특징들에 대해서는 반영하지 못하는 부분들이 존재하기 때문에 그대로 적용하기가 어렵다.

In Guen[12]의 연구에서 제시한 컴포넌트 재사용성 측정 방법은 직접적 측정과 간접적 측정 방법을 제시하였다. 직접적 측정은 컴포넌트를 구성하는 클래스들과 컴포넌트의 인터페이스들을 기반으로해서 얻는 방법으로서, 컴포넌트의 크기, 복잡도, 결합도, 응집도 등을 측정하였다. 간접적 측정은 직접적 측정기준을 토대로 측정이 되는 방법으로서 컴포넌트의 이해도, 적용가능성, 수정가능성, 모듈화 가능성 등을 측정하였다. 이러한 간접적 측정이 궁극적으로 재사용성 측정을 하는데 사용하였다. 그러나 이 방법에서도 프레임워크의 특징인 가변성을 고려한 확장성이나 이식성에 대한 측정은 제시하지 못했다.

3. 모바일 앱의 재사용성 측정 방법

이 장에서는 안드로이드 기반의 모바일 앱 개발에 있어서 앱의 품질을 향상시키기 위한 척도로서 재사용성을 측정하는 메트릭을 제시하고자 한다.

본 논문에서는 모바일 앱의 재사용성을 3가지 관점에서 측정하기 위한 메트릭을 제안하고자 한다. 그 이유는 보다 정확하고 객관적인 측정 결과를 도출하기 위함이다. 본 논문에서 측정하고자 하는 3가지 측면의 측정 요소는 변경성(Changeability), 교체성(Replaceability), 확장성(Extensibility)이다.

3.1 모바일 앱의 변경성 측정 메트릭

모바일 앱에서의 변경성이라 함은 모바일 앱 내의 액티비티(Activity)들 가운데 변경 가능한 액티비티들의 비율을 의미한다[13-16].

[정의 1] 변경성(RAC: Rate of App. Changeability)

$$RAC(m) = \begin{cases} \frac{A_c(m)}{A(m)} & (A(m) > 0) \\ 0 & (otherwise) \end{cases}$$

where,
Ac(m) : number of changeable activities of Mobile App.
A(m) : number of activities of Mobile App.
Confidence Interval : [0.17, 0.34]

3.2 모바일 앱의 교체성 측정 메트릭

모바일 앱에서의 교체성이라 함은 모바일 앱 내의 액티비티들 가운데 교체 가능한 액티비티들의 비율을 의미한다.

[정의 2] 교체성(RAR: Rate of App. Replaceability)

RAR(a)는 모바일앱 내의 액티비티들 가운데 교체 가능한 액티비티들의 비율을 의미한다.

$$RAR(m) = \begin{cases} \frac{A_r(m)}{A(a)} & (A(m) > 0) \\ 0 & (otherwise) \end{cases}$$

where,
Ar(m) : number of replaceable activities of Mobile App.
A(m) : number of activities of Mobile App.
Confidence Interval : [0.17, 0.34]

3.3 모바일 앱의 확장성 측정 메트릭

모바일 앱에서의 확장성이라 함은 모바일 앱 내의 액티비티들 가운데 확장 가능한 액티비티들의 비율을 의미한다.

[정의 3] 확장성(RAE: Rate of Extensibility)

RAE(m)는 모바일 앱 내의 액티비티들 가운데 확장 가능한 액티비티들의 비율을 의미한다.

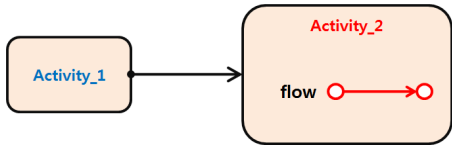
$$RAE(m) = \begin{cases} \frac{A_e(m)}{A(m)} & (A(m) > 0) \\ 0 & (otherwise) \end{cases}$$

where,
Ae(m) : number of extensible activities of Mobile App.
A(m) : number of activities of Mobile App.
Confidence Interval : [0.17, 0.34]

4. 평가

이 장에서는 본 논문에서 제시하는 모바일 앱 재사용성을 측정하기 위해 변경성, 교체성, 확장성을 측정하는 메트릭을 적용한 사례 연구를 제시하고, 이를 측정하여 재사용성을 측정하고자 한다.

Fig. 2는 변경 불가능한 액티비티 관계와 변경 가능한 액티비티 관계를 보여준다. Fig. 2와 같이 설정 되면 해당 Activity_1에서 Activity_2를 호출하게 되는 경우 Activity_2 내의 흐름이 한가지로만 고정된 것이다. 따라서 Activity_2 내에서 다른 어떤 흐름 설정도 변경이 불가능하게 된다.



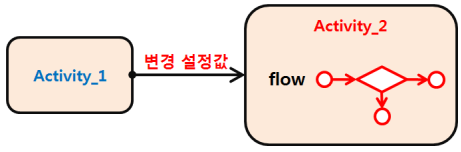
[Fig. 2] Non-Changeable Activity Relations

이에 대한 안드로이드 코드는 Fig. 3과 같이 이루어진다.

```
public class Activity_1 extends Activity {
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.caller_layout_001);
    Button myButton = (Button) findViewById(R.id.button);
    myButton.setOnClickListener(new View.OnClickListener() {
      public void onClick(View v) {
        Intent i = new Intent(IntentAndroid.this, Activity_2.class);
        startActivity(i);
      }
    });
  }
}
```

[Fig. 3] Non-Changeable Activity Relation's Code

Fig. 4는 Activity_1에서 Activity_2로 호출할 경우에 Activity_2 내에서 흐름 설정이 변경 가능한 경우이다.



[Fig. 4] Changeable Activity Relations

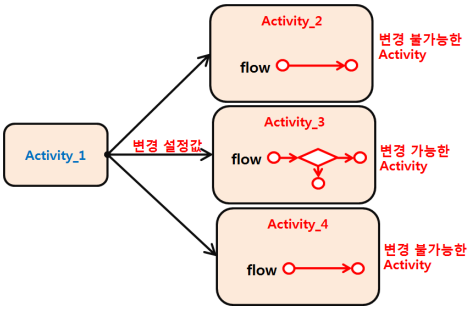
Fig. 4와 같이 변경 가능한 관계에 있게 되면 코드는 Fig. 5와 Fig. 6과 같이 이루어진다.

```
public class Activity_1 extends Activity {
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.caller_layout_001);
    Button myButton = (Button) findViewById(R.id.button);
    myButton.setOnClickListener(new View.OnClickListener() {
      public void onClick(View v) {
        Intent i = new Intent(IntentAndroid.this, Activity_2.class);
        i.putExtra("flow_key", "flow_01");   // 변경 설정값
        startActivity(i);
      }
    });
  }
}
```

[Fig. 5] Changeable Activity's Code

```
public class Activity_2 extends Activity {
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.caller_layout_001);
    Button myButton = (Button) findViewById(R.id.button);
    myButton.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            String key = getIntent().getStringExtra("flow_key");
            if(key.equals("flow_01")) {
              // flow 01
            } else {
              // flow 02
            }
        }
    });
  }
}
```

[Fig. 6] Changeable Activity's Code

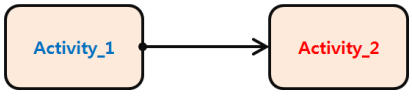


[Fig. 7] Case of Non-Changeable and Changeable

Fig. 7과 같은 사례의 경우, 총 액티비티 수는 4개이고, 변경 가능한 액티비티 수는 Activity_1과 Activity_3 2개이므로 이 모바일 앱의 변경성 RAC(m)=2/4=0.5가 된다.

4.2 교체성에 대한 사례 연구

Fig.8과 같은 사례는 교체 불가능한 액티비티 관계와 교체 가능한 액티비티 간의 관계를 보여준다.



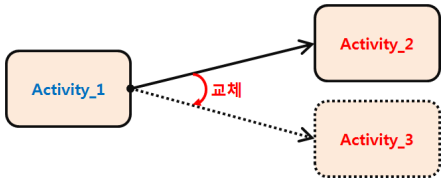
[Fig. 8] Non-Replaceable Activity Relation

Fig. 9는 교체 불가능한 액티비티 관계를 구현한 코드이다.

```
public class Activity_1 extends Activity {
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.caller_layout_001);
    Button myButton = (Button) findViewById(R.id.button);
    myButton.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            Intent i = new Intent(IntentAndroid.this, Activity_2.class);
            startActivity(i);
        }
    });
  }
}
```

[Fig. 9] Non-Replaceable Activity Relation's Code

Fig. 10은 교체 가능한 액티비티 관계를 표현한 것이고, Fig.11과 Fig.12는 이를 구현한 코드이다.



[Fig. 10] Replaceable Activity Relation

```
public class Activity_1 extends Activity {
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.caller_layout_001);
    Button myButton = (Button) findViewById(R.id.button);
    myButton.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            Intent i = new Intent();
            i.setAction(Intent.ACTION_MAIN);

            i.addCategory("_VARIABILITY_NAME");
            startActivity(i);
        }
    });
  }
}
```

_VARIABILITY_NAME은 가변성 식별자 (Activity_2 를 명시하지 않고 가변성 식별자만 명시 하여 동적으로 Activity 변경가능함.)

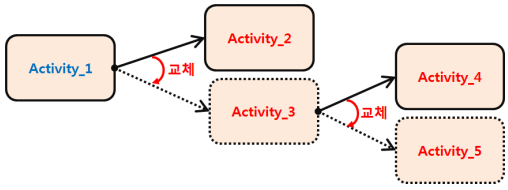
[Fig. 11] Replaceable Activity Relation's Code

```
<application android:icon="@drawable/icon" android:label="@string/activity_name1">
        <activity android:name="Activity_1"
                  android:label="@string/activity_name1">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

<activity android:name="Activity_2" android:label="@string/activity_name2">
            <intent-filter>
                        <action android:name="android.intent.action.MAIN" />
                        <category android:name="android.intent.category.DEFAULT" />
                        <category android:name="_VARIABILITY_NAME" />
            </intent-filter>
</activity>
```

Activity_2 를 Activity_3로 교체 가능함.

[Fig. 12] Replaceable Configuration's Code

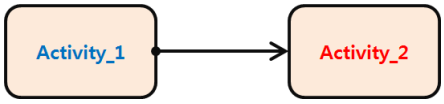


[Fig. 13] Replaceability Case

Fig. 13과 같이 교체성이 이루어지도록 설계될 경우 이에 대한 교체성 측정 결과는 총 액티비티 수 5개 가운데 교체 가능한 액티비티 수가 3개가 되어 3/5=0.6이 된다.

4.3 확장성에 대한 사례 연구

Fig. 14는 확장이 불가능한 액티비티들 간의 관계를 보여준다.



[Fig. 14] Non-Extensible Activity Relation

Fig. 14처럼 확장 불가능한 액티비티들 간의 관계는 Fig. 15와 Fig. 16과 같은 형태의 코드가 개발된다.

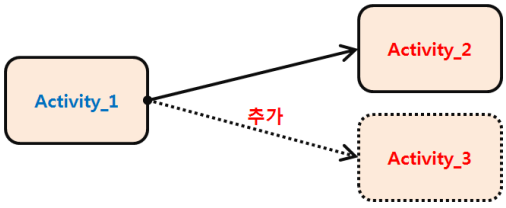
```
public class Activity_1 extends Activity {
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.caller_layout_001);
    Button myButton = (Button) findViewById(R.id.button);
    myButton.setOnClickListener(new View.OnClickListener() {
      public void onClick(View v) {
        Intent i = new Intent(IntentAndroid.this, Activity_2.class);
        startActivity(i);
      }
    });
  }
}
```

[Fig. 15] Non-Extensible Activity Relation's Code

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="inha.android"
    android:versionCode="1"
    android:versionName="1.0">
  <application android:icon="@drawable/icon" android:label="@string/activity_name1">
    <activity android:name="Activity_1"
          android:label="@string/activity_name1">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>

    <activity android:name="Activity_2" > /activity>
```

[Fig. 16] Non-Extensible Configuration's Code



[Fig. 17] Extensible Activity Relation

Fig. 17은 확장이 가능한 액티비티들 간의 관계를 보여준다. Fig. 17처럼 확장 가능한 액티비티들 간의 관계는 Fig. 18처럼 구현되고, Fig. 19와 같은 형태의 설정 코드가 개발된다.

```
public class Activity_1 extends Activity {
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.caller_layout_001);
    Button myButton = (Button) findViewById(R.id.button);
    myButton.setOnClickListener(new View.OnClickListener() {
      public void onClick(View v) {
        Intent i = new Intent();
        i.setAction(Intent.ACTION_MAIN);

        i.addCategory("_VARIABILITY_NAME");
        startActivity(i);
      }
    });
  }
}
```

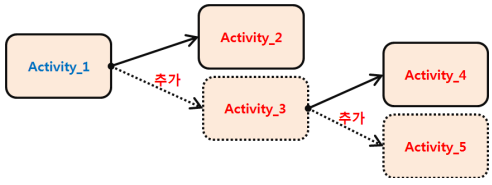
_VARIABILITY_NAME은 가변성 식별자
(Activity_2 를 명시하지 않고 가변성 식별자만 명시하여 동적으로 Activity 확장 가능함.)

[Fig. 18] Extensible Activity Relation's Code

```
<application android:icon="@drawable/icon" android:label="@string/activity_name1">
      <activity android:name="Activity_1"
            android:label="@string/activity_name1">
        <intent-filter>
          <action android:name="android.intent.action.MAIN" />
          <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
      </activity>

<activity android:name="Activity_2" android:label="@string/activity_name2">
        <intent-filter>
          <action android:name="android.intent.action.MAIN" />
          <category android:name="android.intent.category.DEFAULT" />
          <category android:name="_VARIABILITY_NAME" />
        </intent-filter>
</activity>
<activity android:name="Activity_3" android:label="@string/activity_name3">
        <intent-filter>
          <action android:name="android.intent.action.MAIN" />
          <category android:name="android.intent.category.DEFAULT" />
          <category android:name="_VARIABILITY_NAME" />
        </intent-filter>
</activity>
```

[Fig. 19] Extensible Activity Configuration



[Fig. 20] Extensibility Case

따라서 이를 반영한 사례로 Fig. 20의 경우에는 총 액티비티 수가 5개이고, 확장 가능하도록 설정된 액티비티 수가 3개 이므로 3/5=0.6이 된다.

이와 같이 사례 연구에서처럼 변경성, 교체성, 확장성을 합하여 평균을 내면 해당 모바일 앱의 재사용성 측정 값이 산출된다. 위 사례에서는 변경성이 0.5, 교체성이 0.6, 확장성이 0.6이므로 이에 대한 재사용성은 0.56이 된다. 이 값은 가장 높을 경우가 1에 해당하게 된다.

5. 결론 및 향후 연구과제

본 연구를 통해 얻을 수 있는 기대효과는 다음과 같다.

첫째, 모바일 앱 개발시 향후 재사용성이 높은 앱을 개발할 수 있다. 둘째, 개발된 모바일 앱의 품질을 측정할 수 있는 도구로 사용할 수 있다. 셋째, 모바일 앱의 기능 및 속성의 확장성을 높일 수 있도록 개발할 수 있다. 본 연구를 통해 도출된 연구 결과는 안드로이드 기반의 모바일 애플리케이션 개발에 적용함으로써, 모바일 앱 개발의 생산성 향상을 가져올 수 있을 뿐만 아니라 안드로이드 기반의 모바일 앱 개발에 있어서 재사용성의 향상을 도모할 수 있다. 또한 이는 결국 안드로이드 기반의 모바일 앱의 품질 향상을 기대할 수 있다.

향후 연구로는 제안된 재사용성 측정 메트릭을 적용하여 이를 자동으로 측정할 수 있는 자동화 도구를 개발하는 것이다.

References

[1] Flurry, Flurry Analytics Reort, http://www.flurry.com

[2] Google Android [Online]. http://www.android.com

[3] Apple ios-7, http://www.apple.com

[4] Jeffrey S. Poulin, "Measuring Software Reusability," Proceeding of the Third International Conference on Software Reuse, Rio de Janeiro, Brazil, 1-4 November 1994, pp.1-5.

[5] Jeffrey S. P., "Measuring Software Reusability", IEEE Software, 1994.
DOI: http://dx.doi.org/10.1109/ICSR.1994.365803

[6] Prieto-Diaz, Ruben and Peter Freeman, "Classifying Software for Reusability," IEEE Software, Vol. 4, No. 1, January 1987, pp.6-16.
DOI: http://dx.doi.org/10.1109/MS.1987.229789

[7] Caldiera, Gianluigi and Victor R. Basili, "Indentifying and Qualifying Reusable Software Components," IEEE Software, Vol. 24, No. 2, Febuary 1991, pp.61-70.
DOI: http://dx.doi.org/10.1109/2.67210

[8] Pressman, R.S., Software Engineering: A Practitioner's Approach, McGraw-Hill, 2002.

[9] STARS, "Repository Guidelines for the Software Technology for Adaptable, Reliable Systems(STARS) Program," CDRL Sequence Number 0460,15 March 1989.

[10] ISO/IEC, FCD 9126-1.2 Information Technology Software product quality-Part 1:Quality model, 1998.

[11] ISO/IEC JTC1/SC7 N2419 "DTR 9126-2: Software Engineering - Product Quality Part 2 - External Metrics", 2001.

[12] In Guen Park, Soo Dong Kim, "Software Component Reusability Metrics", Vol.31, No.6, pp.760-772, June, 2004.

[13] Sang Hyung Kim, "Android Programming Complete Guide", Hanbit Media, April, 2013.

[14] B.-H. Oh, K.-S.k Hong, "Multi-Modal User Distance Estimation System based on Mobile Device", The Journal of The Institute of Internet, Broadcasting and Communication, Vol. 14, No. 2, pp.65~71, 2014.

[15] J.-S. Seo, S.-C. Park, "Design and Implementation of Support System for Personalized Medical Service Based on Mobile", The Journal of The Institute of Internet, Broadcasting and Communication, Vol. 13, No. 6, pp. 37~45, 2013.

[16] H.-T. Seok, "Hybrid Web App Development for Eye movement at Mobile Devices", The Journal of The Institute of Internet, Broadcasting and Communication, Vol. 13, No. 6, pp. 249~254, 2013.

조 은 숙(Eun-Sook Cho) **[정회원]**

- 1993년 2월 : 동의대학교 전산통계학과 졸업(이학사)
- 1996년 2월 : 숭실대학교 대학원 컴퓨터학과(공학석사)
- 2000년 2월 : 숭실대학교 대학원 컴퓨터학과(공학박사)
- 2000년 9월 ~ 2005년 2월 : 동덕여자대학교 정보학부 강의전임교수
- 2005년 3월 ~ 현재 : 서일대학교 컴퓨터 소프트웨어과 부교수

<관심분야>
Software Engineering, Embedded Software, Mobile Computing

김 철 진(Chul-Jin Kim) **[정회원]**

- 1996년 2월 : 경기대학교 전자계산학과(학사)
- 1998년 2월 : 숭실대학교 컴퓨터공학부(공학석사)
- 2004년 2월 : 숭실대학교 컴퓨터공학부(공학박사)
- 2004년 3월 ~ 2009년 2월 : 삼성전자 책임연구원
- 2009년 3월 ~ 현재 : 인하공전 컴퓨터시스템과 부교수

<관심분야>
Software Engineering, Embedded Software, Mobile Computing

송 치 양(Chee-Yang Song) **[정회원]**

- 1985년 2월 : 한남대학교 전자계산학과 (공학사)
- 1987년 2월 : 중앙대학교 전자계산학과 (공학석사)
- 2003년 8월 : 고려대학교 컴퓨터학과 (이학박사)
- 1990년 5월 ~ 2005년 9월 : KT 중앙연구소 책임연구원
- 2005년 10월 ~ 2008년 2월 : 상주대학교 소프트웨어공학과 조교수
- 2008년 3월 ~ 현재 : 경북대학교 컴퓨터정보학부 부교수

<관심분야>
CBD, MDA, Service-oriented modeling, Formal specification