Journal of the Korea Academia-Industrial cooperation Society
Vol. 13, No. 5 pp. 2338-2346, 2012

http://dx.doi.org/10.5762/KAIS.2012.13.5.2338

안드로이드 기반 모바일 가변성 설계 및 구현

김철진[1], 조은숙[2*]
[1]인하공업전문대학 컴퓨터시스템과, [2]서일대학 컴퓨터 소프트웨어과

A Design and Implementation of Mobile Variability based on Android

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

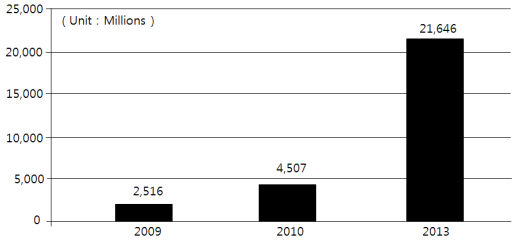
요 약 향후 모바일 어플리케이션 규모는 커질 것으로 예상되며 이에 따라 다른 모바일 어플리케이션과 또는 서버와의 결합도가 커질 것이다. 모바일 어플리케이션 규모의 증가는 가변성을 위한 예측 설계가 수반되어야 함을 의미한다. 현재 모바일 어플리케이션 변경이 발생할 경우 어플리케이션 전체를 재설치 해야 한다. 그러나 이러한 재설치는 결합도가 큰 어플리케이션인 경우 부작용(Side-Effect)이 발생할 가능성이 높다. 따라서 본 논문에서는 안드로이드 플랫폼 기반에서 어플리케이션 가변성에 대해 설계할 수 있는 기법을 제안한다. 이러한 모바일 가변성 기법은 선택 기법과 플러그인 기법으로 구분한다.

Abstract According to the size of mobile applications has been expanded, the coupling of among mobile applications or servers also will be growing. The growth of mobile application's size means that predicting design for variability should be involved. If mobile application's change is occurred, application should be reinstalled totally. However this reinstallation can raise side-effects in case of high-coupling applications. Therefore, this paper proposes a technique of designing variability for mobile applications in android platform. Proposed technique is separated into selection technique and plug-in technique.

Key Words : Mobile Application, Variability, Variability Design, Android

1. 서론

그림 1에 나타난 것처럼 어플리케이션의 시장 규모가 2009년 다운로드 회수가 2,516 million에서 2013년 21,646 million을 크게 증가할 것으로 예상한다[1,2]. 이렇게 폭발적으로 증가하는 모바일 어플리케이션들은 그림 2의 (1)과 같이 대부분 독자적으로 운영되고 있는 스탠드얼론(Stand Alone) 어플리케이션 이지만, 그림 2의 (2)와 같이 앞으로 어플리케이션들 간의 또는 서버와의 관계를 통해 결합도가 높아 질 것으로 예상할 수 있다. 이러한 결합도의 증가는 어플리케이션의 변경에 따른 오류를 발생시킬 수 있는 부작용을 안고 있다. 따라서 어플리케이션을 효과적으로 변경하기 위한 체계 및 기법이 필요하다[3,4].

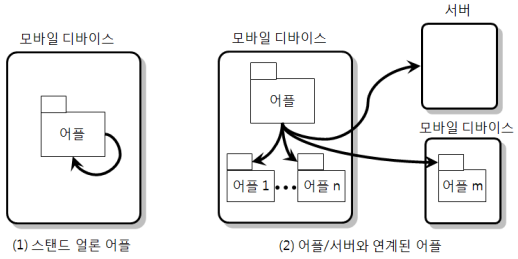


[그림 1] 모바일 앱 다운로드 (참조:Gartner)
[Fig. 1] Downloads of Mobile App. (Ref. Gartner)

*Corresponding Author : Eun-Sook Cho
Tel: +82-2-490-7562 email: escho@seoil.ac.kr
접수일 12년 04월 23일 수정일 12년 05월 03일 게재확정일 12년 05월 10일

본 논문에서는 어플리케이션을 정적 또는 동적으로 효과적으로 변경하기 위한 모바일 어플리케이션 가변성 기법을 제안한다. 본 논문에서 제안하는 가변성 기법으로는 정적/동적 선택 기법과 플러그인 기법으로 구분하여 제안한다.

본 논문의 구성으로 2장에서는 관련연구로 컴포넌트 가변성과 재사용 프레임워크에 대해 설명하며, 3장에서는 본 논문의 핵심인 모바일 커스터마이제이션 기법을 선택 기법과 플러그인 기법으로 구분하여 제안한다. 4장에서는 3장에서 제안한 기법을 실 사례에 적용하여 적합성을 검증한다.



[그림 2] 모바일 어플 구조
[Fig. 2] Mobile App.'s Structure

2. 관련 연구

2.1 컴포넌트 가변성

가변성(Variability)[5]은 공통성(Commonality)과 함께 제품군(Product Family)[6]의 요소로서 제품군 멤버들 사이의 차이를 말한다. TV 제품군에서 가변성은 화면 사이즈가 되며, 공통성은 화면 분사 방식이나 방송파 수신 방식 등이다. 이와 같이 가변성은 동일 기능을 하는 제품군의 멤버들 간의 차이로서 컴포넌트에서 가변성은 소프트웨어 구성 요소와 일치하는 속성, 오퍼레이션, 그리고 메시지의 흐름이 된다.

행위 가변성은 동일 컴포넌트 멤버들 간에 동일 기능을 하는 오퍼레이션이 서로 다른 알고리즘으로 기능을 제공하는 경우에 행위 가변성이라고 한다. 예를 들면, 은행 도메인의 계좌 컴포넌트에서 이자 계산 기능은 은행마다 존재하지만 서로 다른 알고리즘으로 기능을 제공하기 때문에 은행 마다 이자 계산의 기능은 각각 다르다. 따라서 이자 계산은 행위 가변성에 해당한다.

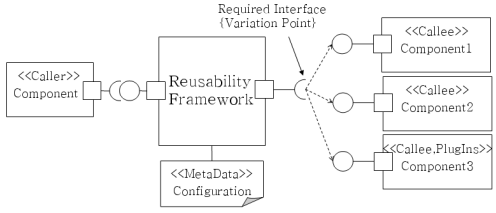
워크플로우 가변성[7]은 동일 컴포넌트 멤버들 간에 동일 기능이 서로 다른 메시지 흐름을 제공하는 경우에 워크플로우 가변성이라고 한다. 예를 들면, 회원 등록하는 회원 컴포넌트에서 회원 입력, 회원 인증, 데이터 베이스 저장의 프로세스를 요구할 수 도 있지만 특정 도메인에서는 데이터 베이스 저장이 아니라 레거시 시스템으로 연동되는 프로세스로 요구할 수 있기 때문에 동일한 회원 등록에 대해 다른 워크플로우를 요구하는 경우에 워크플로우 가변성이라고 한다.

속성 가변성은 동일 컴포넌트 멤버들 간에 동일 역할의 속성이 서로 다른 타입이나 값을 가지는 경우 속성 가변성이라고 한다[5]. 예를 들면 동일 컴포넌트에서 날짜를 나타내는 속성이 도메인의 요구 사항에 따라 스트링 타입이나 데이트 타입으로 요구할 수 있는데 이러한 경우를 속성 가변성이라고 할 수 있다.

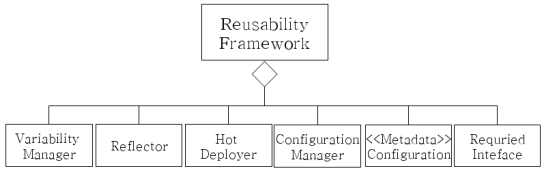
2.2 컴포넌트 재사용 프레임워크[8,9]

컴포넌트 재사용 프레임워크는 컴포넌트들 간의 통합을 유연하게 제공하여 다양한 서비스를 제공할 수 있다[10]. 그림 3은 컴포넌트 재사용 프레임워크를 통해 컴포넌트의 가변부를 설계할 수 있음을 나타낸다. 이러한 가변부에 대한 변경은 물리적으로 컴포넌트 코드의 변경이 아닌 설정정보(메타정보)를 변경함으로 가능하다.

컴포넌트 재사용 프레임워크를 구성하는 주요 핵심 요소들은 그림 4와 같이 가변성을 담당하는 가변성 관리자(Variability Manager), 동적 전개기(Hot Deployer), 설정 관리자(Configuration Manager), 리플렉터(Reflector) 등으로 구성된다.



[그림 3] 컴포넌트 재사용 프레임워크
[Fig. 3] Component Reusability Framework



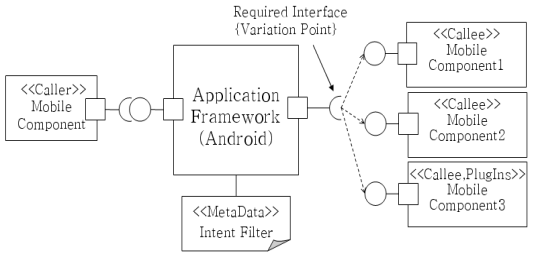
[그림 4] 컴포넌트 재사용 프레임워크 핵심요소
[Fig. 4] Core Elements of Component Reusability Framework

가변성 관리자는 가변부를 사용할 수 있도록 중계하는 역할을 하며 재사용 프레임워크의 인터페이스 역할을 한

다. 가변부를 사용하는 영역에 의해 특정 가변부를 호출하게 되면 가변성 관리자는 재사용 프레임워크 내의 리플렉터, 설정 관리자, 동적 전개기, 자원 할당기, 그리고 가변성 어댑터들과 상호 작용을 통해 가변부의 특정 기능을 동적으로 호출한다.

동적 전개기는 홈 네트워크 시스템 내부에서 가변성을 제공할 수 없는 경우 시스템 외부에서 요구하는 기능 클래스를 시스템 내부로 플러그-인 시키기 위한 도구이다. 플러그의 개념은 시스템 패키지 내에 요구하는 클래스를 포함하는 것이 아니라 시스템 운영 환경에 맞게 객체가 생성되는 것을 의미 한다.

설정 관리자는 가변부 사용 영역에 의해 사용될 가변부의 메타정보를 관리한다. 설정 관리자는 가변부의 가변부 식별자를 통해 가변부 메타정보를 호출하며, 이러한 가변부 식별자는 가변부 사용 영역에서 정의하여 가변성 관리기를 통해 설정 관리자에 전달한다. 가변부 사용 영역에서 전달된 가변부 식별자를 기반으로 설정 관리자는 가변부의 상세한 가변부 메타정보를 얻는다.

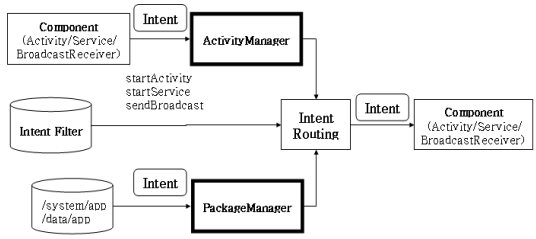


[그림 5] 안드로이드 어플리케이션 프레임워크
[Fig. 5] Android Application Framework

리플렉터는 가변부의 메타정보를 통해 물리적인 가변성 어댑터나 가변성 클래스를 호출하기 위한 도구로서 동적으로 클래스를 호출할 수 있도록 하기 위해 리플렉션(Reflection) 기능을 기반으로 한다. 리플렉션은 메타 형태의 클래스 명(String 타입)과 행위 명(String 타입), 그리고 입력 파라메터(Object Array 타입)를 제공하여 물리적인 클래스의 기능을 호출할 수 있는 메커니즘이다. 이러한 리플렉션 메커니즘은 표준 개발 플랫폼(J2EE, .NET) 에서 제공되고 있다. 재사용 프레임워크의 리플렉터는 이러한 메커니즘을 변경하여 가변부의 메타정보를 처리할 수 있는 기능을 제공한다. 이와 같이 재사용 프레임워크의 리플렉터는 가변부 클래스의 다양한 행위뿐만 아니라 다양한 인터페이스의 클래스를 동적으로 변경할 수 있도록 한다.

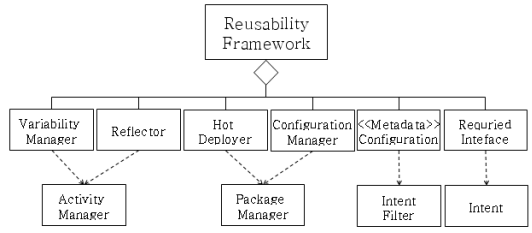
2.3 안드포이드 어플리케이션 프레임워크

안드로이드의 어플리케이션 프레임워크는 그림 4의 컴포넌트 재사용 프레임워크처럼 안드로이드 플랫폼 상에서 모바일 컴포넌트(어플리케이션 또는 액티비티)를 유연하게 조합하여 다양한 서비스를 제공하기 위한 프레임워크이다. 그림 5에서와 같이 가변부에 대해 설정하기 위해 인텐트(또는 인텐트 필터) 라는 메타정보를 이용한다.



[그림 6] 안드로이드 어플리케이션 프레임워크 구조
[Fig. 6] Android Application Framework's Architecture

안드로이드 플랫폼 상에서 재사용 프레임워크 기능을 제공하기 위해 어플리케이션 프레임워크 내의 액티비티 관리자(Activity Manager)와 패키지 관리자(Package Manager)가 핵심적인 역할을 한다.



[그림 7] 컴포넌트 재사용 프레임워크과 안드로이드 재사용 프레임워크 간의 비교
[Fig. 7] Component Reusability Framework vs Android Reusability Framework

그림 6과 같이 액티비티 관리자는 컴포넌트로부터 가변적으로 호출되는 데이터 정보를 인텐트를 통해 전달하며, 패키지 관리자는 인텐트 필터에 정의된 가변적인 호출 컴포넌트 정보를 필터링하여 해당 컴포넌트(어플리케이션)를 호출한다[5]. 인텐트 필터나 인텐트를 변경하면 액티비티 관리자와 패키지 관리자가 변경된 메타정보에 따라 가변적으로 컴포넌트를 변경한다.

컴포넌트 재사용 프레임워크 핵심 요소([그림 4])와 안드로이드 어플리케이션 프레임워크 내의 재사용 구성요소 간에 관계는 그림 7과 같다.

가변성 관리자와 리플렉터는 동적으로 컴포넌트를 연결해 주므로 안드로이드 어플리케이션 프레임워크의 액티비티 관리자와 같은 역할이다. 동적 전개기와 설정 관리자는 메타정보를 관리하기 위한 역할을 하므로 패키지 관리자와 동일하다고 할 수 있다.

설정정보나 요구 인터페이스는 인텐트 필터와 인텐트에 해당한다고 할 수 있다.

3. 모바일 커스터마이제이션 기법

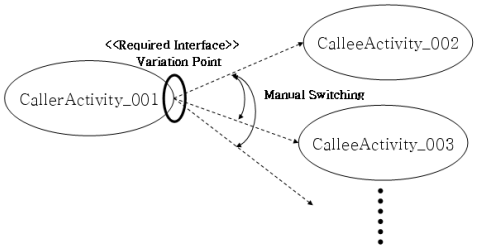
안드로이드 플랫폼의 어플리케이션 프레임워크에 기반하여 서비스를 커스터마이제이션 하기 위한 기법으로 선택 기법과 플러그인 기법을 제안한다[9].

3.1 선택 기법(Selection Technique)

선택기법은 모바일 어플리케이션 내의 여러 서비스 컴포넌트 중에 하나의 컴포넌트를 선택하여 요구하는 서비스를 제공하는 기법이다. 선택기법은 정적 선택 기법과 동적 선택 기법을 구분할 수 있다.

3.1.1 정적 선택 기법(Static Selection Technique)

정적 선택 기법은 어플리케이션 내의 서비스를 다양하게 제공할 수 있도록 서비스 실행 중에 컴포넌트를 변경하기 위한 기법이다.

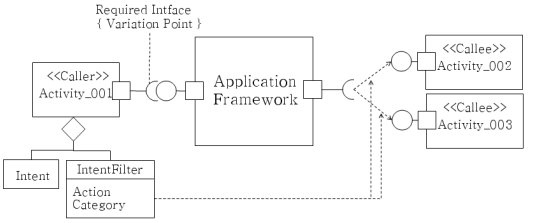


[그림 8] 정적 선택 기법
[Fig. 8] Static Selection Technique

그림 8과 같이 “CallerActivity_001” 컴포넌트에서 선택적으로 “CalleeActivity_002”나 “CalleeActivity_003” 컴포넌트를 선택할 수 있다. 이러한 서비스의 선택은 서비스 사용자에 의해 선택되며 모바일 서비스 실행 중에 변경 가능하다.

안드로이드 플랫폼 기반의 정적 선택 기법을 위한 클래스 다이어그램은 그림 9와 같다.

“Application Framework”는 안드로이드 플랫폼 상에서 어플리케이션의 라이프사이클을 관리하거나 컴포넌트를 변경하기 위한 매커니즘을 제공한다. 동적인 서비스 구성을 위해 호출 컴포넌트는 인텐트와 인텐트 필터를 포함해야 하며 어플리케이션 프레임워크는 해당 인텐트 필터를 통해 서비스를 선택할 수 있도록 필터링 해 준다.



[그림 9] 정적 선택기법 클래스 다이어그램
[Fig. 9] Class Diagram of Static Selection Technique

정적 선택 기법을 위해 선택 영역을 가변성 영역으로 정의해야 하며, 안드로이드 플랫폼에서 인텐트 필터를 활용하여 정의할 수 있다.

표 1에서와 같이 “Action”과 “Category” 필터를 정의할 수 있으며, 선택적으로 호출하기 위한 컴포넌트들은 “Category” 필터에 가변성 식별자(“com.app.VARIABILITY_NAME”)를 정의해야 한다. 또한 호출되는 컴포넌트들도 해당 가변성 식별자를 정의해야 한다.

[표 1] 정적선택의가변성을 위한 인텐트 필터
[Table 1] Intent Filter for Variability of Static Selection

Filter Type	Filter Value	Etc
Action	android.intent.action.MAIN	
Category	android.intent.category.DEFAULT	Filter for Implict Intent
	com.app.VARIABILITY_NAME	Variability Name

그림 10은 표 1에 대한 가변성 정보를 안드로이드 플랫폼 상에서 설정한 설정 코드이다.

그림 11과 같이 인텐트 필터를 적용하여 가변성 부분을 동적으로 선택할 수 있도록 구현할 수 있다. 사용자는 서비스 실행 중에 요구하는 컴포넌트를 선택하여 서비스를 변경할 수 있다.

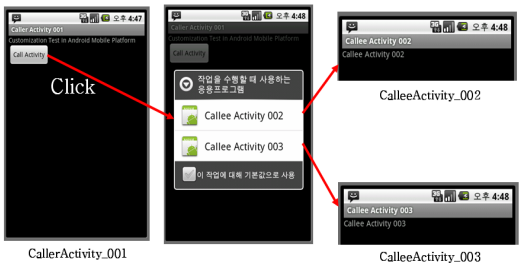
3.1.2 동적 선택 기법(Dynamic Selection Technique)

동적 선택 기법은 정적 선택 기법과 동일하게 어플리케이션 내의 서비스를 다양하게 제공할 수 있도록 서비스 실행 중에 컴포넌트를 변경하기 위한 기법이다. 그러나 그림 12와 같이 “CallerActivity_001” 컴포넌트에서 “CalleeActivity_002" 컴포넌트나 “CalleeActivity_003” 컴

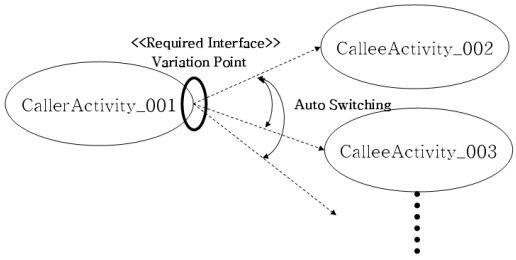
포넌트로 자동으로 변경되 어 선택된다.

```
...
<activity android:name="CalleeActivity_002" >
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="com.app._VARIABILITY_NAME" />
  </intent-filter>
</activity>
<activity android:name="CalleeActivity_003" >
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="com.app._VARIABILITY_NAME" />
  </intent-filter>
</activity>
</application>
...
```

[그림 10] 정적 선택의 가변성을 위한 설정
[Fig. 10] Configuration of Static Selection's Variability

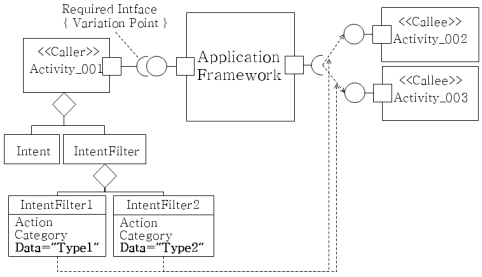


[그림 11] 정적 선택 기법 사례
[Fig. 11] A Case of Static Selection



[그림 12] 동적 선택 기법
[Fig. 12] Dynamic Selection Technique

안드로이드 플랫폼 기반의 동적 선택 기법을 위한 클래스 다이어그램은 그림 13과 같다. 정적 선택 기법의 구조와 다르게 동적 선택 기법은 동적으로 컴포넌트를 선택할 수 있도록 "Data" 타입의 인텐트 필터를 추가한다. 인텐트 필터에서 정의한 데이터 타입을 요구하는 컴포넌트를 자동으로 호출할 수 있도록 한다. 그림 13의 구조에서와 같이 "Activity_001"에서 전달하는 데이터 타입이 "Type1"인 경우에는 자동으로 "Activity_003" 컴포넌트를 호출하며, 데이터 타입이 "Type2"인 경우에는 자동으로 "Activity_002"를 호출한다.



[그림 13] 동적 선택 기법 클래스 다이어그램
[Fig. 13] Class Diagram of Dynamic Selection

```
...
<activity android:name="CalleeActivity_002" >
  ① <intent-filter>
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.DEFAULT" />
      <category android:name="com.app._VARIABILITY_NAME" />
    </intent-filter>
</activity>
<activity android:name="CalleeActivity_003" >
  ② <intent-filter>
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.DEFAULT" />
      <category android:name="com.app._VARIABILITY_NAME" />
        <data android:scheme="http" />
    </intent-filter>
</activity>
</application>
...
```

[그림 14] 동적 선택 기법의 가변성을 위한 설정
[Fig. 14] Configuration of Dynamic Selection's Variability

[표 2] 동적 선택 기법의 가변성을 위한 인텐트 필터
[Table 2] Intent Filter for Dynamic Selection's Variability

- Send Data by Intent

Name	ID	Type
Data_1	Data_1	String
Data_n	Data_n	String

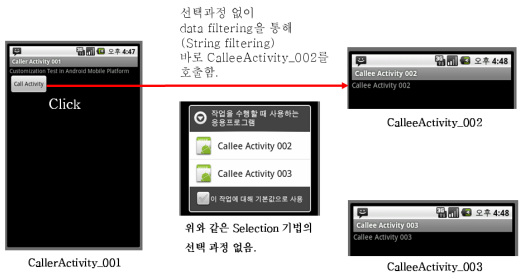
- Intent Filter

Filter Type	Filter Value	Etc
Action	android.intent.action.MAIN	
Category	android.intent.category.DEFAULT	Filter for Implict Intent
	com.app.VARIABILITY_NAME	Variability Name
Data	String / URL / MIME	Data Type

동적 선택 기법의 가변성 정의는 표 2와 같이 인텐트 필터를 통해 정의할 수 있다. 정적 선택 기법에 추가적으로 "Data" 타입을 추가하여 동적인 변경을 가능하게 할 수 있다. 데이터 타입 중에 "String", "URL", "MIME" 타입을 지정할 수 있다.

안드로이드 플랫폼 상에서 동적 선택 기법을 위한 Manifest 코드는 그림 14와 같다. ①번 인텐트 필터는 데이터 타입을 설정하지 않았다. 이럴 경우 전달되는 데이터가 문자열(String) 인 경우 해당 컴포넌트를 호출하도록

한다. ②번 인텐트 필터인 경우 “http” 타입을 지정하였으므로 URI 데이터 타입을 전달할 경우 해당 컴포넌트를 호출한다.

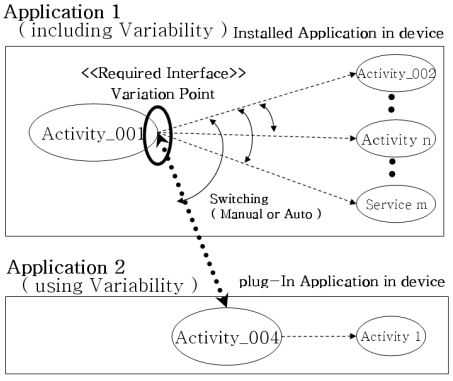


[그림 15] 동적 선택 기법 사례
[Fig. 15] A Case of Dynamic Selection

그림 15는 그림 14의 동적인 선택을 위한 Manifest 설정 정보를 통해 구현된 동적 선택 기법의 사례이다. 전달되는 데이터 타입이 문자열인 경우 “CalleeActivity_002” 컴포넌트가 호출되며, 전달되는 데이터 타입이 URI 인 경우 “CalleeActivity_003” 컴포넌트가 호출될 수 있다. 이와 같이 동적 선택 기법은 정적 선택 기법과 다르게 서비스 실행 중에 사용자의 선택 과정 없이 동적으로 변경된다.

3.2 플러그인 기법 (Plug-In Technique)

플러그인 기법은 기존 어플리케이션에 새로운 컴포넌트를 플러그인 하여 새로운 서비스를 제공하는 방법이다. 선택 기법이 어플리케이션 내에 존재하는 컴포넌트들을 변경하는 것에 반해 플러그인 기법은 어플리케이션 내에 존재하지 않는 새로운 컴포넌트를 플러그인 하여 새로운 서비스를 제공한다.

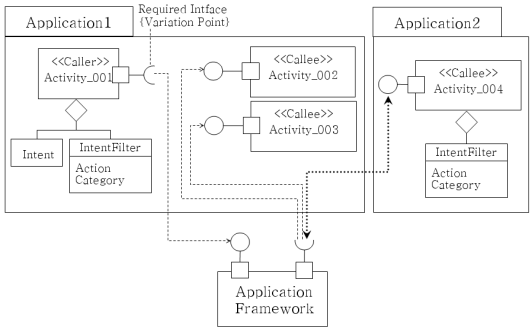


[그림 16] 플러그인 기법
[Fig. 16] Plug-In Technique

그림 16에서와 같이 기존 어플리케이션 “Application 1”은 새로운 컴포넌트인 “Activity_004”를 플러그인 하여 새로운 서비스 제공할 수 있다.

플러그인 되는 컴포넌트는 어플리케이션 “Application 2” 내에 포함되어 디바이스에 설치된다. “Application 1”과 “Application 2”는 독립된 어플리케이션이므로 독립적으로 설치될 수 있다. “Activity_001”은 가변부를 통해 “Activity_002”, … “Activity n”, … “Service m”, 그리고 “Activity_004” 컴포넌트로 변경이 가능하다.

안드로이드 플랫폼 기반의 플러그인 기법을 위한 클래스 다이어그램은 그림 17과 같다.



[그림 17] 플러그인 기법 클래스 다이어그램
[Fig. 17] Class Diagram of Plug-In Technique

“Application 1”의 “Activity_001” 컴포넌트는 안드로이드의 “Application Framework”으로 인텐트 필터를 전달하여 가변부에 해당하는 컴포넌트를 선택할 수 있다. 이때 “Application 1” 내부뿐 만 아니라, 플러그인 된 “Application 2”의 “Activity_004” 컴포넌트도 선택 가능하다. 플러그인 기법의 가변성 정의는 표 3과 같이 인텐트 필터를 통해 정의할 수 있다.

플러그인 기법의 가변성 인텐트 필터는 정적 선택 기법의 가변성 인텐트 필터에 추가적으로 “android.intent.category.ALTERNATIVE”를 추가해야 한다. “.ALTERNATIVE”는 외부 어플리케이션과의 연동을 위한 인텐트 값이다.

[표 3] 플러그인 기법의 가변성을 위한 인텐트 필터
[Table 3] Intent Filter for Plug-In’s Variability

Filter Type	Filter Value	Etc
Action	android.intent.action.MAIN	
Category	android.intent.category.DEFAULT	Filter for Implict Intent
	android.intent.category.ALTERNATIVE	Filter for PlugIn
	com.app.VARIABILITY_NAME	Variability Name

그림 18에서와 같이 “category” 타입의 인텐트 값으로 “android.intent.category.ALTERNATIVE”를 설정하면 “Application 2”의 “Activity_004” 컴포넌트는 외부 어플리케이션에서 호출될 수 있다.

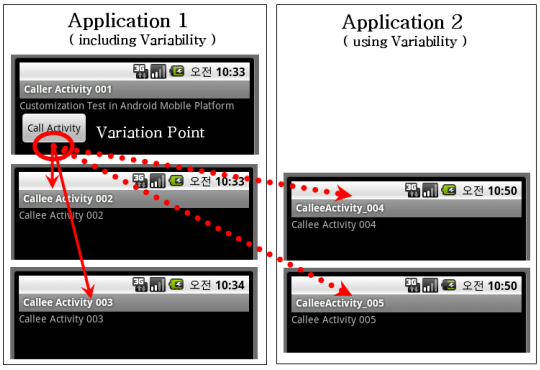
① Application 1's Intent Filter

```
...
<activity android:name="CalleeActivity_002" >
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="com.app._VARIABILITY_NAME" />
  </intent-filter>
</activity>
...
```

② Application 2's Intent Filter

```
...
<activity android:name="CalleeActivity_004" >
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.DEFAULT" />
    <cateogry android:name="android.intent.cateogry.ALTERNATIVE" />
    <category android:name="com.app._VARIABILITY_NAME" />
  </intent-filter>
</activity>
</application>
...
```

[그림 18] 플러그인 기법의 가변성을 위한 설정
[Fig. 18] Configuration for Plug-In Technique's Variability

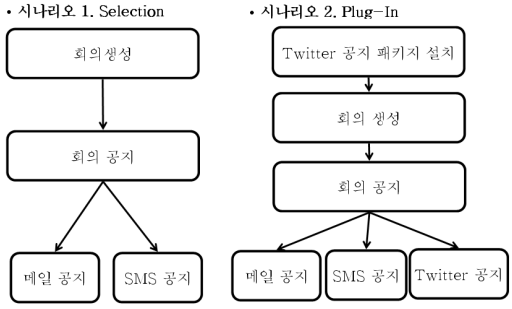


[그림 19] 플러그인 기법 사례
[Fig. 19] A Case of Plug-In Technique

그림 19는 그림 18의 Manifest 설정 정보를 통해 구현된 플러그인 기법의 사례이다. “Activity_001”에서 내부 컴포넌트인 “Activity_002”나 “Activity_003”을 호출할 수 있으며, 플러그인 된 외부 컴포넌트인 “Activity_004”나 “Activity_005”를 호출할 수 있다. 이때는 정적 선택 기법이나 동적 선택 기법을 이용할 수 있다.

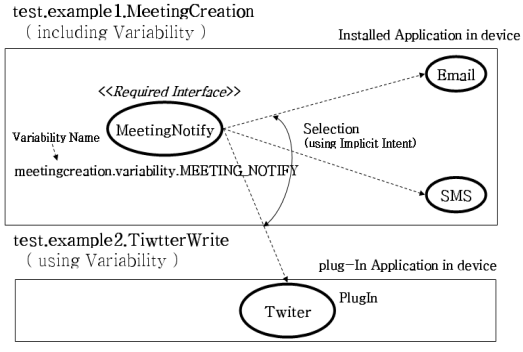
4. 실험 및 평가

본 논문에서 제안한 안드로이드 기반의 가변성 설계 및 구현 기법을 회의 생성/공지 사례에 적용하여 적합성을 검증한다. 그림 20은 회의 생성 시 회의 정보를 공지하기 위한 시나리오이다.



[그림 20] 회의 생성 및 공지 시나리오
[Fig. 20] Scenario of Meeting Creation and Notification

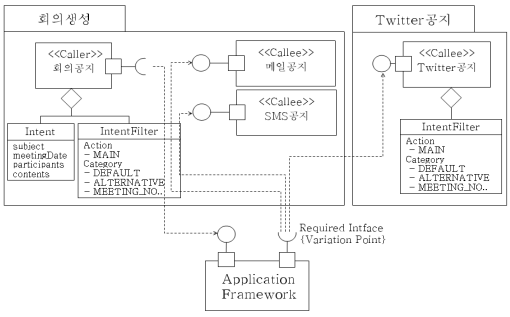
시나리오 1은 회의 공지를 선택 기법에 의해 메일 및 SMS를 선택하여 공지할 수 있도록 하기 위한 사례이다. 시나리오 2는 새로운 공지 서비스를 제공하기 위해 Twitter[11] 공지 컴포넌트를 플러그인하여 Twitter 공지 서비스를 선택할 수 있도록 하기 위한 사례이다.



[그림 21] 회의 공지를 위한 가변적 서비스 구조
[Fig. 21] Architecture of Variant Service for Meeting Notification

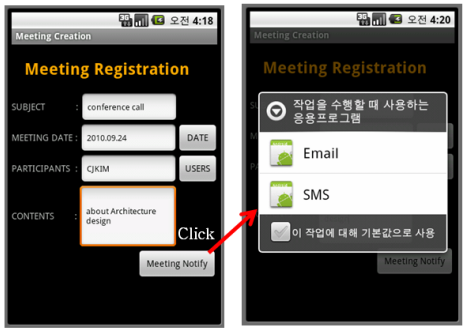
회의 공지를 가변적으로 처리하기 위한 구조는 그림 21과 같다. “MeetingNotify” 컴포넌트는 어플리케이션 내부의 “Email” 이나 “SMS” 컴포넌트를 호출할 수 있거나, 플러그인 된 외부 컴포넌트인 “Twitter”를 호출하여 회의 공지를 가변적으로 변경할 수 있다.

회의 공지에 대한 클래스 다이어그램은 그림 22와 같다. 회의 공지를 위한 어플리케이션 내부 및 외부 컴포넌트를 호출 할 수 있도록 인텐트 필터를 포함하며, 안드로이드 플랫폼의 “Application Framework”을 통해 내부 및 외부 컴포넌트를 호출 할 수 있다.



[그림 22] 회의 공지 클래스 다이어그램
[Fig. 22] Class Diagram of Meeting Notification

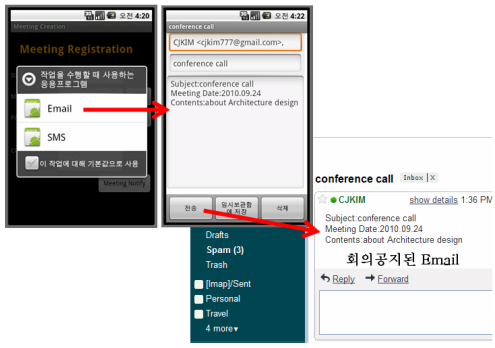
인텐트 필터 값 "meetingcreation.variability.MEETING_NOTIFY"은 회의 공지 서비스의 가변부를 처리하기 위한 정보이며, "android.intent.category.ALTE RNATIVE"는 플러그인 된 외부 컴포넌트와의 가변부 처리를 위한 정보이다.



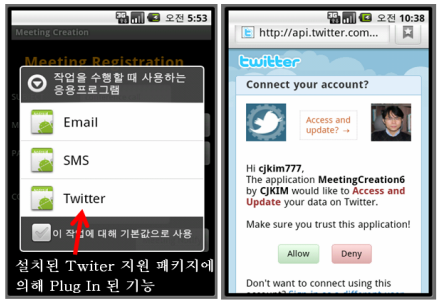
[그림 23] 회의 공지 사례에서 선택 기법 적용
[Fig. 23] Applying Selection Technique in Meeting Notification Case

회의 생성 및 회의 공지를 위한 가변성 설계를 기반으로 그림 23에서 그림 25와 같이 구현했다. 그림 23은 회의 공지를 위한 정적 선택 기법을 적용하였으며, "Email" 이나 "SMS" 컴포넌트는 가변적 선택을 위한 표 4의 인텐트 필터 정보를 정의했다.

그림 24는 선택된 "Email" 컴포넌트로 회의 생성 시 생성된 "회의 주제", "회의 날짜", "회의 참여자", "회의 내용" 정보가 전달되었으며 메일로 공지된 결과이다. 그림 25는 새로운 회의공지 컴포넌트인 "Twitter"를 플러그인 한 경우이다. 정적 선택 기법을 위한 선택 메뉴가 자동적으로 추가되었다. 그림 24의 "Email" 컴포넌트로 회의 정보를 전송하는 것과 마찬가지로 "Twitter" 컴포넌트로 회의 정보가 전송되며, 트위터와 연동을 위해 초기 인증 과정을 수행한다.



[그림 24] Email을 통한 공지 사례
[Fig. 24] A Case of Notification by Email



[그림 25] 플러그인 된 Twitter 컴포넌트를 통한 회의 공지 사례
[Fig. 25] A Case of Meeting Notification by Plug-In's Twitter Component

위의 사례를 통해 안드로이드 모바일 플랫폼에서의 가변성 적용이 가능함을 검증하였다. 제안된 모바일 선택 기법과 플러그인 기법을 통해 유연한 가변성 처리가 가능함을 증명하였다.

5. 결론 및 향후 연구과제

모바일 어플리케이션은 변경이 발생할 경우 모바일 어플리케이션 전체를 재설치 해야 한다. 이러한 재 설치는 모바일 어플리케이션의 규모가 커지거나 다른 어플리케이션과의 결합도가 커지면 부작용이 발생할 가능성이 있다. 따라서 본 논문에서는 안드로이드 플랫폼 기반에서 어플리케이션의 변경에 대해 기존 어플리케이션의 변경 없이 수정이 가능한 설계 기법을 제시하였으며 구현을 통해 가능성을 검증하였다. 또한 향후 본 모바일 커스터마이제이션 설계 기법을 활용하여 자동화된 업그레이드를 위한 스마트 업그레이드에 대한 연구를 진행할 예정이다.

References

[1] Konig-Ries, B. and Jena, F., "Challenges in Mobile Application Development," it-Information Technology, Vol.52, No.2, pp.69-71, 2009

[2] Android Developers, http://developer.android.com/index.html (accessed January 31, 2011).

[3] Caldiera, Gianluigi and Victor R. Basili, "Indentifying and Qualifying Reusable Software Components," IEEE Software, Vol. 24, No. 2, Febuary 1991, pp.61-70.

[4] Salmre, I., Writing Mobile Code: Essential Software Engineering for Building Mobile Applications, Addison-Wesley Professional, 2005.

[5] Heineman, G. T. and Councill, W. T., “Component - Based Software Engineering”, Addison-Wesley, 2001.

[6] Atkinson, C., Bayer, J., Bunse, C., Kamsties, E., Laitenberger, O., Laqua, R., Muthig, D., Paech, B., Wust, J., Zettel, J., Component-based Product Line Engineering with UML, Pearson Education Ltd, 2002.

[7] Kim C. J. and Kim S. D., “A Component Workflow Customization Technique”, Vol.27,No.5, Korea Information Science Society, 2000.

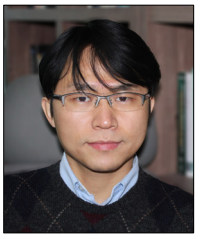
[8] Kim C. J., Lee S. H. and Cho E. S., “A Framework for Improving Reusability of Home Network System”, Vol.1, No.2, ITIRC, Sep 2008.

[9] Chul Jin Kim, Eun Sook Cho, Chee Yang Song, "A Design Technique of Configurable Framework for Home Network Systems", Journal of the Korea Academia-Industrial Cooperation Society, Vol. 12, No.4, pp.1844-866, April 2011.

[10] Eun Sook Cho, Chul-Jin Kim, Sook Hee Lee, "A Study on Reusability Metric of Framework for Embedded Software”, Journal of the Korea Academia-Industrial Cooperation Society, Vol 12, No.11, pp.5252-5259, December 2011.

[11] http://dev.twitter.com/apps/new

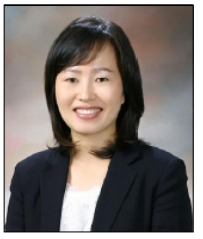
김 철 진(Chul-Jin Kim) **[정회원]**



- 2004년 2월 : 숭실대학교 대학원 컴퓨터학과 (공학박사)
- 2004년 4월 : 가톨릭대학교 컴퓨터 정보공학부 강의전담교수
- 2004년 3월 ~ 2009년 2월 :삼성전자 책임연구원
- 2009년 3월 ~ 현재 : 인하공전 컴퓨터시스템과 조교수

<관심분야>
컴포넌트 기반 개발 방법론, 컴포넌트 커스터마이제이션, 모바일 서비스, 클라우드 컴퓨팅

조 은 숙(Eun-Sook Cho) **[정회원]**



- 2000년 2월 : 숭실대학교 대학원 컴퓨터학과 (공학박사)
- 2000년 9월 ~ 2005년 2월: 동덕여자대학교 강의전임교수
- 2005년 3월 ~ 현재 : 서일대학교 컴퓨터소프트웨어과 부교수

<관심분야>
컴포넌트 기반 개발 방법론, 서비스지향 아키텍처(SOA), 프레임워크 설계 및 개발, 클라우드 컴퓨팅