http://dx.doi.org/10.3745/KIPSTD.2011.18D.4.245

서비스 기반 안드로이드 어플리케이션의 설계 및 구현 프로세스

이 호 중[†] · 라 현 정[††] · 금 창 섭[†††] · 김 수 동[††††]

요 약

인터넷의 빠른 보급과 함께 발전한 모바일 디바이스는 유연한 이동성과 함께 언제 어디서나 네트워크에 연결되는 특성을 가지고 있고, 어플리케이션들을 실행할 수 있다. 그러나, 모바일 디바이스의 특성인 자원 제약성에 영향을 받기 때문에 복잡한 기능을 하는 어플리케이션을 실행시킬 수가 없다. 그러므로, 일부 기능을 외부에 위치하거나 외부 서비스를 이용하는 접근 방법에 대해 연구가 진행되고 있다. 안드로이드는 대표적인 모바일 플랫폼 중 하나로, 제한된 자원을 가지는 모바일 디바이스 상에서 여러 모바일 어플리케이션들을 실행시킬 수 있도록 설계되었다. 서비스 개념과 안드로이드 플랫폼을 이용한 서비스 기반 안드로이드 어플리케이션은 어플리케이션 기능이 분산되었다는 점과 안드로이드에 특화된 컴포넌트를 포함하고 있다는 점에서 전형적인 소프트웨어와 차이점을 가지고 있다. 그러므로, 일반적으로 널리 적용된 객체지향 분석 및 설계 기법은 서비스 기반의 안드로이드 모바일 어플리케이션 개발에는 그대로 적용되기가 어렵다. 본 논문은 기존 개발 프로세스를 확장하여 서비스 기반 안드로이드 어플리케이션의 개발 프로세스를 제안한다. 우선 서비스 기반 안드로이드 어플리케이션을 개발함에 있어서 발생되는 설계 이슈를 도출한다. 그리고, 각 도출된 이슈를 해결하기 위하여 기존의 객체지향 개발 프로세스를 확장한다. 확장된 프로세스는 이슈를 구체적으로 해결하기 위한 상세 지침과 설계 결과 양식을 포함한다. 마지막으로, 사례연구를 통해 확장된 프로세스의 적용 결과를 보여준다. 본 논문에서 제안된 설계 프로세스는 서비스 기반 안드로이드 어플리케이션 개발을 보다 체계적이고 효과적으로 수행하는데 기본적인 방법론 지침으로 활용될 수 있다.

키워드 : 안드로이드, 서비스 기반 어플리케이션, 객체지향 개발

A Process to Design and Implement Service-based Android Applications

Ho Joong Lee[†] · Hyun Jung La[††] · Chang Sup Keum[†††] · Soo Dong Kim[††††]

ABSTRACT

Mobile Devices, which are developed with the fast growing of the Internet, have flexible internet accessibility and can access the network anywere so that they can execute software applications. However, it is very challenging to deploy highly complex applications on mobile devices since they have limited resources. To overcome the limitation, researches on applying a concept of services to mobile applications have been proposed. Android is one of the popular mobile platforms and is designed to effectively execute mobile applications on the mobile devices having limited resources. Since service-based Android applications, which adopt a concept of services and Android platform, invoke remote services and are built with Android-specific components, they are much different from traditional software applications. Consequently, it is not straightforward to apply object-oriented (OO) analysis and design methods to developing service-based Android applications, although they have been frequently applied to developing traditional applications. In this paper, we present a process to develop service-based Android mobile applications, which extends a traditional OO development process. First, we raise design issues to be considered in developing service-based Android applications. Then, to solve the issues, we present detailed guidelines for essential phases of OO-based development process that are customized to service-based Android applications. Finally, to show applicability of the process, we perform a case study. The proposed design process is effectively utilized as a set of guidelines to develop service-based Android applications more systematically and effectively.

Keywords : Android, Service-based Application, Object oriented Development

※ 이 논문은 정보통신산업진흥원의 SW공학 요소기술 연구개발사업에 의해 지원되었음.
※ 본 연구는 방송통신위원회의 차세대통신네트워크원천기술개발사업의 연구결과로 수행되었음(KCA-2011-(09913-05001)).
† 준 회 원 : 숭실대학교 컴퓨터학과 석사과정
†† 정 회 원 : 숭실대학교 모바일서비스 소프트웨어공학센터 연구교수(교신저자)
††† 정 회 원 : 한국전자통신연구원 서비스융합연구팀 책임연구원
†††† 종신회원 : 숭실대학교 컴퓨터학부 교수
논문접수 : 2011년 2월 14일
수 정 일 : 1차 2011년 4월 19일, 2차 2011년 5월 6일
심사완료 : 2011년 5월 11일

1. 서 론

모바일 디바이스는 무선 인터넷 접속 기능을 제공하는 휴대 가능한 모든 종류의 장비들을 총칭하는 용어로서, 유연한 무선 인터넷 접속성과 높은 이동성을 제공한다는 장점을 가지고 있다. 이런 모바일 디바이스는 PC와 같이 CPU, 메모리 등으로 구성되고, 운영체제를 탑재하고 있으므로, PC를 대체할 장비로 널리 사용될 것으로 예상된다. 그리고, 모바일 디바이스가 널리 보급됨에 따라 복잡한 기능을 수행하는 모바일 어플리케이션의 수요도 많아지고 있다[1].

구글이 중심이 되어 개발한 개방용 모바일 플랫폼인 안드로이드는 제한된 자원을 가지는 모바일 디바이스 상에서 여러 어플리케이션 개발자들이 제공하는 어플리케이션 또는 서비스를 실행시킬 수 있도록 설계되었다[2]. 그러므로, 모바일 어플리케이션 개발을 위한 플랫폼을 널리 사용되고 있다[3][4].

일반적으로 소프트웨어 어플리케이션을 개발하기 위하여 UML 기반의 객체지향 분석 및 설계 기법 (OOAD)을 사용하였지만, 이 기법은 두 가지 관점에서 서비스 기반 안드로이드 모바일 어플리케이션 개발에 적용되기 어렵다.

첫째, OOAD 기법은 안드로이드 기반의 모바일 어플리케이션 특징을 수용하지 못한다.

- 모바일 디바이스는 제한된 하드웨어 자원을 가지고 있으며, 풍부한 인터넷 접근 기능을 제공한다는 기존의 PC와는 구별된 특징을 가진다.
- 안드로이드 플랫폼을 기반으로 개발한 모바일 어플리케이션은 기존의 소프트웨어 어플리케이션과 차별되는 컴포넌트를 기반으로 구성된다.

둘째, 자원된 제약성을 가지는 모바일 디바이스에서 복잡한 기능의 어플리케이션을 수행하기 위하여 서비스 개념을 이용하는 것은 보편적인 방법이다[5]. 이는 어플리케이션의 복잡한 기능은 클라우드 서비스 및 서버 기능에서 실행하도록 하고, 모바일 디바이스에서는 이를 호출하도록 하는 것이다. 이런 어플리케이션은 서비스 기반 모바일 어플리케이션이라고 부르며, 여기서 서비스는 한 모바일 어플리케이션만을 위해 제공되는 서버의 기능뿐 아니라 구글의 Map 서비스와 같은 클라우드 서비스를 모두 포함한다. 그 결과, 모바일 디바이스의 자원 한계성을 극복하면서 복잡도가 높은 소프트웨어의 실행도 가능해진다. 그러나, 전통적인 OOAD 방법론들은 서비스를 설계 및 구현하고, 이 기능을 호출하여 사용하게 하는 지침을 제공하지 않는다.

그러므로, 본 논문에서는 서비스 기반 안드로이드 어플리케이션을 설계 및 개발하기 위해 두 가지의 한계점을 보완할 수 있는 프로세스를 개발한다.

- 차별화된 어플리케이션 컴포넌트로 구성된 안드로이드 어플리케이션을 구현하기 위한 모바일 어플리케이션의 설계 및 구현 지침을 제공한다.
- 모바일 디바이스의 제한된 자원을 이용하여 비교적 복잡한 기능을 가지는 모바일 어플리케이션을 실행시킬 수 있는 서비스 기반의 모바일 어플리케이션 설계 지침을 제공한다.

이를 위해, 3장에서는 서비스 기반의 안드로이드 어플리케이션 설계에 대한 기술적 이슈를 도출한다. 4장에서는 OOAD 프로세스를 확장하여, 서비스 기반 안드로이드 어플리케이션을 구현하기 위한 산출물 및 상세 설계 지침을 정의한다. 5장에서는 제안된 프로세스의 적용 가능성과 3장에서 제기한 이슈가 어떻게 해결되었는지를 보이기 위하여, Mobile Mate Service(MMS) 프로젝트에 적용한 사례를 보여준다. 본 논문에서 제안하고 있는 서비스 기반 모바일 어플리케이션 개발 프로세스 및 상세 지침은 서비스와 클라이언트 어플리케이션으로 구성된 다소 복잡한 구조의 모바일 어플리케이션을 적은 노력을 이용하여 효과적으로 개발하도록 도와준다. 그리고, 안드로이드 플랫폼의 특징을 반영하여 프로세스를 제안함으로써, 플랫폼에 특화된 설계 이슈를 해결하는데 도움이 되도록 한다.

2. 관련 연구

Abrahamsson 등의 연구에서는 모바일 개발의 기술적인 제약사항을 보완하기 위하여 애자일(Agile) 기반의 방법론인 Mobile-D를 제안하였다[6]. 제안된 방법론은 익스트림 프로그래밍 (Extreme Programming), 크리스탈 방법론 (Crystal Methodologies), RUP (Rational Unified Process)를 기반으로 하고 있고, set-up, core, core2, stabilize, wrap-up의 다섯 단계로 구성되어 있다. 그리고, 다섯 단계를 수행하기 위한 아홉 가지 실천사항을 제안하였다. 본 논문은 모바일 어플리케이션을 개발하기 위해 애자일 방법론을 특화 시킨 방법론을 제안하였지만, 구체적인 활동, 상세 지침 등을 제공하지 않는다.

Jeong 등의 연구에서는 모바일 플랫폼에 배포될 어플리케이션 소프트웨어를 개발하는 방법론인 MASAM (Mobile Application SW Based Agile Methodology)를 제안하였다[7]. MASAM 방법론은 다양한 플랫폼에서 실행되는 어플리케이션 개발하기 위해 모델 기반 아키텍처(Model Driven Architecture) 개념, 신속한 개발을 위한 애자일 방법론, 도메인 지식을 활용하기 위한 패턴 기반 기법을 이용하여 정의되었고, 개발 준비 단계, 구체화 단계, 상품 개발 단계, 상품화 단계로 구성되어 있다. 그리고, 방법론을 지원하기 위한 도구를 제안하였다. 본 연구는 프로세스의 상세 지침 보다는 개략적인 프로세스와 이를 지원하는 도구를 제안하는데 중점을 두고 있다.

Braun과 Eckhaus의 연구에서는 모바일 데이터 서비스 기반의 모바일 어플리케이션 개발을 위한 아키텍처를 제안하였다[8]. 제안된 아키텍처는 클라이언트-서버 아키텍처 스타일을 기반으로 하고 있으며, 클라이언트에는 GUI, 비즈니스 로직으로 구성되어 있고 서버에는 비즈니스 로직과 데이터베이스로 구성된다. 그리고, 이 모바일 어플리케이션을 개발하기 위한 모델 기반 프로세스를 제안하였다. 모바일 어플리케이션 기능을 클라이언트와 서버에 나누어 배치시킴으

로써 모바일의 자원 제약사항을 극복하였지만, 이에 대한 구체적인 설계 지침을 다루어지지 않았다.

Natchetoi 등의 연구에서는 J2ME 를 실행시킬 수 있는 모바일 디바이스에서 실행되는 비즈니스 어플리케이션을 위한 경량의 SOA (Service-Oriented Architecture) 기반의 아키텍처를 제안하였다[9]. 이 연구에서는 주로 모바일 디바이스 고유의 특징을 해결하기 위한 설계 기법을 정의하였다. 본 논문에서는 모바일 디바이스의 제한적인 자원 문제를 해결하는데 주요 초점을 맞추고 있지만, 모바일 어플리케이션 개발 프로세스 상에서 제안된 기법들이 어떻게 적용되는지가 구체적으로 명시되지 않았다.

Ughetti 등의 연구 [10]와 Shu 등의 연구 [11]에서는 위치 기반의 모바일 어플리케이션을 개발하기 위한 구현 방법을 제안하였다. Ughetti 등의 연구에서는 상호작용 가능한 지적인 에이전트 시스템 개발을 위한 JADE 프레임워크와 안드로이드 플랫폼을 이용하여 P2P 어플리케이션을 개발하는 방법을 제안하였고, 이를 위치 기반의 채팅 시스템에 적용하였다. 이 두 연구는 구체적으로 안드로이드 기반 어플리케이션 개발을 위한 접근 방법을 제안하였지만, 분석 및 설계 지침보다는 구현 지침을 주로 정의하였다.

모바일 어플리케이션 개발 프로세스에 대한 연구들에서는 범용적인 모바일 어플리케이션을 개발하는 프로세스를 제안하지만, 대부분 애자일 방법론을 택하고 있으며 개략적인 프로세스만을 제안하고 있다. 모바일 어플리케이션의 제약사항을 해결하기 위한 연구들에서는 SOA를 채택해 제약사항에 대한 문제를 해결하고 있지만 전체적인 설계에 대해서는 다루지 않는다. 안드로이드 개발 관련된 연구들에서는 안드로이드 기반의 어플리케이션을 개발하는 순서적인 프로세스를 제안하기 보다는 특정 도메인에 국한된 개략적 설계 수준의 어플리케이션 설계 방법만을 다루었다.

3. 서비스 기반 안드로이드 어플리케이션의 기술적 이슈

안드로이드 어플리케이션은 대표적인 객체 지향 언어인 자바로 개발되기 때문에 기존의 OOAD를 따라 개발하는 것이 효과적이다. 그러나 서비스 기반의 안드로이드 어플리케이션이 가지는 특징 때문에, 기존 OOAD를 실제 개발에 그대로 적용하기는 어렵다. 본 장에서는 서비스 기반 안드로이드 어플리케이션의 특징을 반영하여 OOAD를 적용하기 위해 해결해야 하는 주요한 기술적 이슈 7개를 도출한다.

- 리소스 사용의 최소화: 모바일 디바이스는 부족한 리소스로 인해, 복잡한 기능을 수행할 수 없고 많은 양의 데이터를 저장하고 사용하는데 제한적이다[3]. 그러므로, 적은 자원을 효과적으로 활용하고 보완할 수 있는 설계 기법이 필요하다.

- 안정적 네트워크 연결 설계: 서비스 기반 모바일 어플리케이션은 안정적인 네트워크 연결성을 보장하지 않고는 기능을 제공하기 힘들다[12]. 그러나, 모바일 사용자의 잦은 이동으로 네트워크 연결이 불안정하기 때문에, 네트워크 연결을 보장하는 기법을 제공해야 한다.

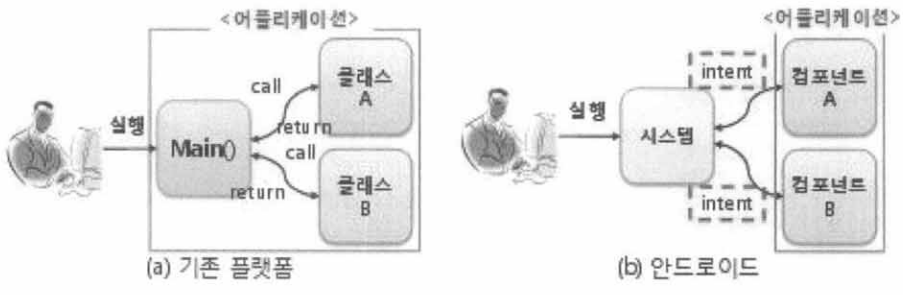
- 서비스 설계 및 이용: 서비스 기반 어플리케이션으로 모바일 디바이스의 리소스 사용을 최소화하기 위해 복잡한 기능을 서비스로 분할한다[13]. 분할된 기능을 수행하는 서비스에 대한 설계와 서비스를 이용할 수 있는 인터페이스의 정의가 필요하다.

- 효율적인 사용자 인터페이스 설계: 모바일 다비이스의 작은 화면 때문에 필요한 정보만을 효율적으로 출력해야 하므로[14], 이를 위한 설계 지침을 제공해야 한다.

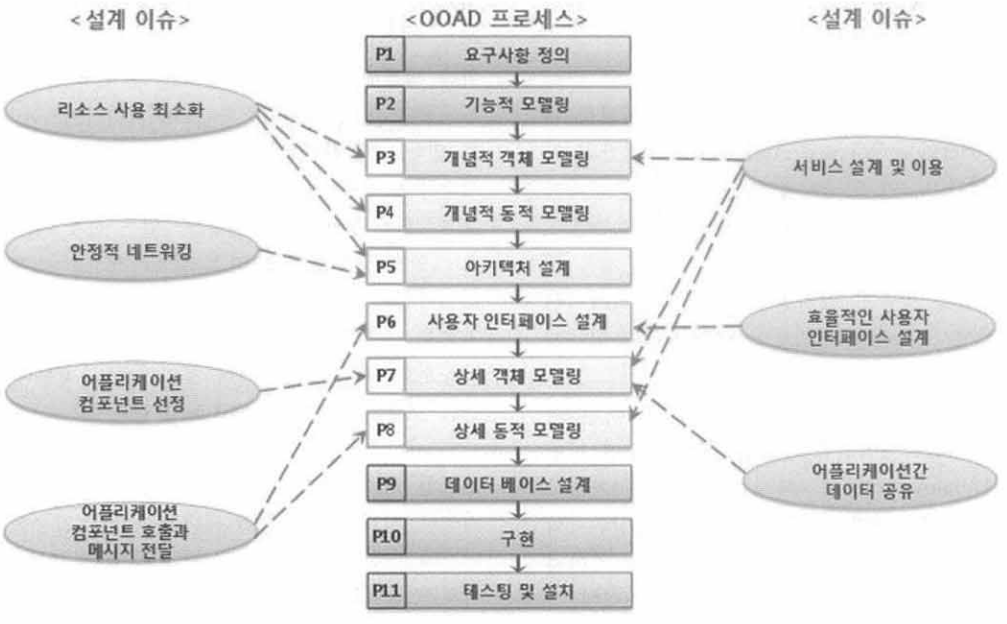
- 어플리케이션 컴포넌트 선정: 어플리케이션 컴포넌트는 안드로이드 어플리케이션을 구성하는 가장 중요한 4개의 요소로, 액티비티, 서비스, 컨텐트 프로바이더, 브로드케스트 리시버가 있다[1]. 각 요소들은 기능에 따라 정확하게 구분되기 때문에, 이를 고려한 설계 지침이 필요하다.

- 어플리케이션 컴포넌트 호출과 메시지 전달: (그림 1)의 (a)는 기존의 대부분의 플랫폼이 가지는 어플리케이션 실행 과정이다. 어플리케이션을 실행하면 내부적으로 모든 호출이 일어난다. 그러나, 안드로이드 환경에서는 (그림 1)의 (b)와 같이 모든 호출에는 시스템이 관여하며 인텐트라는 메시지를 통해 이루어진다[1]. 따라서 컴포넌트 호출 시 사용되는 인텐트 기반 메시지 전달을 고려한 설계 기법이 필요하다.

- 어플리케이션 간 데이터 공유: 어플리케이션에서 사용하는 데이터를 타 어플리케이션에서도 사용되는 경우가 있다. 예를 들면 주소록과 같은 정보들이다. 안드로이드는 다른 어플리케이션 간의 직접적인 데이터 공유를 허용하지 않는다[1]. 이는 기존 어플리케이션에서는 발생하지 않는 현상이므로, 데이터 공유를 위한 설계 지침이 필요하다.



(그림 1) 어플리케이션 컴포넌트 실행 과정



(그림 2) 설계 이슈에 따른 OO 개발 프로세스에 끼치는 영향 관계

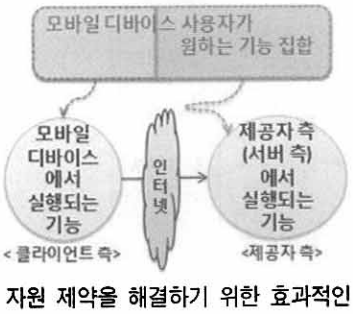
4. 설계 이슈의 해결책을 반영한 OOAD 설계 프로세스

본 장에서는 (그림 2)와 같이 3장에서 도출한 기술적 이슈가 OOAD 프로세스에 미치는 영향 관계를 고려하여 이를 해결하는상세 지침을 제안한다. 따라서 안드로이드 어플리케이션 설계 이슈가 반영되지 않은 부분은 기존의 OOAD프로세스와 동일하기 때문에 생략하고 해당하는 단계를 위주로 설명한다.

(그림 2)의 OOAD 프로세스는 대표적인 객체지향 프로세스인 RUP를 기반으로 도출되었다[15]. 도출된 OOAD 프로세스는 UP의 주요 단계를 추출한 분석과 설계 과정을 가지고 있으며 이를 통해 효과적인 객체 지향 소프트웨어 개발이 가능하다.

4.1 P3. 개념적 객체 모델링 및 P4. 개념적 동적 모델링

안드로이드 어플리케이션은 제한된 자원을 이용해 효과적으로 실행되어야 한다. 그러므로 (그림 3)과 같이 효과적인 기능 분할을 통해 서버의 풍부한 자원을 활용하여 복잡도가 높은 기능도 실행할 수 있다.



(그림 3) 자원 제약을 해결하기 위한 효과적인 기능 분할

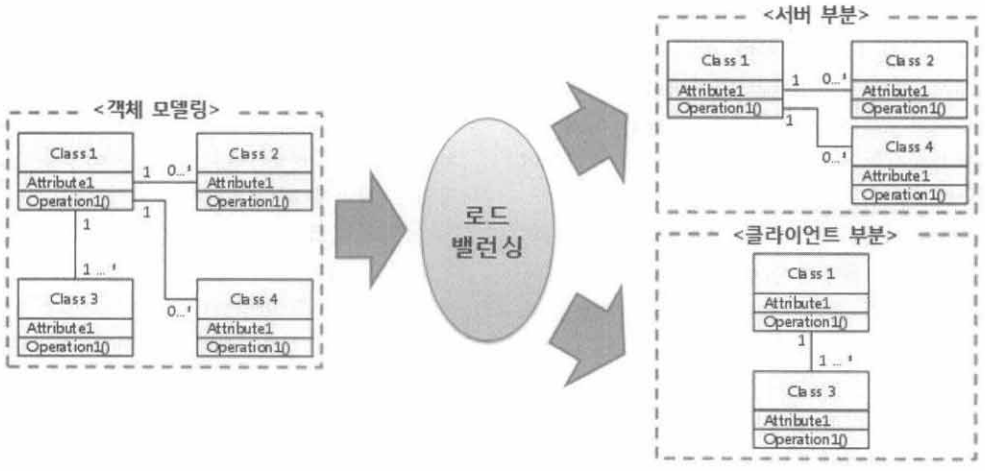
기존 어플리케이션과 달리, 안드로이드 어플리케이션은 개략적인 설계 시에 응집도가 높고 결합도가 낮은 객체와 객체 간의 관계의 식별뿐만 아니라 추가적으로 서버와 클라이언트의 기능 로드 밸런싱(Load balancing)이 필요하다. 따라서 로드 밸런싱은 기존의 객체 모델링 지침을 이용하여 클래스 다이어그램을 작성한 후, 클래스 또는 클래스 내의 오퍼레이션 별로 다음 항목을 고려하여 수행한다.

- 자원 소모량: 오퍼레이션 별로 자원 소모량을 예측하여, 복잡한 계산을 수행하는 오퍼레이션은 서버에 배치하도록 한다.
- 재사용성: 여러 모바일 어플리케이션을 고려하면, 공통적으로 사용되는 기능을 발견할 수 있다 [1]. 이런 기능은 서버에 위치하여 범용적으로 사용하는 것이 자원의 효율성이 더 높다. 그러므로, 객체에 속한 오퍼레이션의 기능을 분석하여, 잠재적인 재사용성을 예측한 뒤, 서버에 위치시킬 기능을 식별해야 한다. 재사용성은 간단하게 다음과 같은 식을 통해 예측할 수 있다[16].

$$\text{재사용성} = \frac{\text{해당 오퍼레이션 기능을 요구하는 어플리케이션 수}}{\text{해당 도메인과 관련된 총 어플리케이션 수}}$$

또한, 재사용성은 시장성 조사나 도메인 분석을 통한 개략적인 예측값으로도 판단할 수 있다.

- 여러 어플리케이션/클라이언트가 공유하는 데이터 관리: 여러 어플리케이션 또는 클라이언트가 공통적으로 사용하는 데이터를 관리하는 객체 및 기능은 서버에 배치하도록 한다.
- 성능 관련 요구사항: 엄격한 시간 등의 성능을 요구하는 기능은 가급적이면 클라이언트에 배치하도록 한다.
- 보안 관련 요구사항: 보안 관련 기능은 클라이언트에 배치하도록 한다.



(그림 4) 로드 벨런싱 후 객체 모델 분할

• 사용자와의 상호작용 횟수: 상호작용 횟수가 많은 기능의 경우에는 사용자와 어플리케이션 간의 데이터 전달이 빈번하게 이루어지므로, 이런 기능은 클라이언트에 배치하도록 한다.

이외에도 목표 어플리케이션의 고유한 특성에 따라서 추가적인 사항이 고려될 수 있다. 이렇게 고려한 결과를 <표 1>의 양식을 사용하여 명세한다.

〈표 1〉 기능 로드 밸런싱 결과표

클래스 이름	오퍼레이션 이름	배치 장소		근 거
		클라이언트	서버	
클래스 1	오퍼레이션 1		✓	

위의 결과표를 분석하면, (그림 4)와 같이 객체 모델을 서버와 클라이언트 부분으로 분할할 수 있다. 또한, 이 결과는 개념적 동적 모델링에도 영향을 미치게 된다.

4.2 P5. 아키텍처 설계

본 단계에서는 비기능적 요구사항을 고려하여 아키텍처를 설계한다. 안드로이드 어플리케이션 역시 기존의 소프트웨어의 한 가지 형태이기 때문에, (그림 5)와 같은 전형적인 아키텍처 설계 과정 [14]을 따라 수행한다.

4.2.1 활동 1. 아키텍처 드라이버 도출

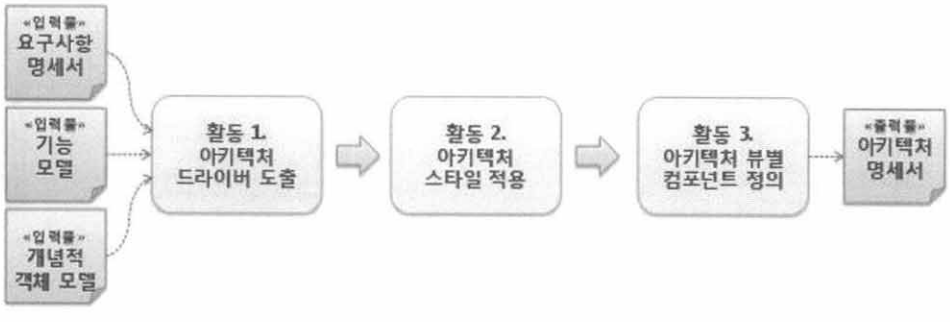
이 활동에서는 안드로이드 어플리케이션 요구사항 명세서의 비기능적 요구사항으로부터 아키텍처 드라이버를 도출한다.

어플리케이션 종속적인 항목은 목표 어플리케이션의 요구사항에 종속적이므로, 본 논문에서는 다루지 않는다. 먼저, 안드로이드 어플리케이션은 일종의 모바일 어플리케이션이기 때문에, 모바일 어플리케이션 특징을 고려하여 다음과 같은 아키텍처 드라이버를 도출할 수 있다.

• 자원 효율성을 고려한 아키텍처 설계
• 불안정한 네트워크 및 부족한 전원을 보완하기 위한 높은 신뢰도를 가진 아키텍처 설계
• 다양한 인터넷 접근성을 이용한 높은 이용성을 가진 아키텍처 설계
• 모바일 어플리케이션에서 호출하는 서비스 또는 서버 기능의 높은 확장성을 고려한 아키텍처 설계

그리고, 안드로이드 어플리케이션은 운영 환경인 안드로이드 플랫폼에 종속적이므로, 이와 관련된 표준, 규격, 기술 문서 등의 광범위한 분석을 통하여, 다음의 아키텍처 드라이버를 정의할 수 있다.

• 안드로이드 플랫폼의 아키텍처 관련 지침, 규격을 준수한 설계



(그림 5) 아키텍처 설계의 주요 활동

4.2.2 활동 2. 아키텍처 스타일 적용

이 활동에서는 목표 안드로이드 어플리케이션을 위한 사용 가능한 아키텍처 스타일을 식별한다.

먼저, 활동 1에서 식별된 아키텍처 드라이버를 고려하여 적용 가능한 아키텍처 스타일을 선택한다. 식별된 아키텍처 드라이버가 가진 요구사항의 특징을 각 아키텍처 스타일이 지원하는 특징과 비교한다.

안드로이드 어플리케이션의 자원 효율성을 고려하여, MVC 아키텍처 스타일을 확장한 Balanced MVC 아키텍처 스타일을 적용하고 서버와 클라이언트간의 로드 밸런싱 작업을 통해 아키텍처를 설계한다[17].

신뢰도, 이용성, 확장성이 높은 아키텍처를 설계하기 위해서는 서버의 기능이 한 서버에 집중되기 보다는 분산시키는 것이 효율적이다. 이는 서버나 서버의 기능 모듈이 제대로 된 기능을 수행하지 못할 경우에 다른 곳에서 그 기능을 대신 수행할 수 있기 때문이다. 이 경우, Master-Slave 아키텍처 스타일을 사용하여 서비스 간의 로드 밸런싱이나 실시간 업무이관 (Task Migration)을 지원할 수 있다[18].

4.2.3 활동 3. 아키텍처 뷰별 컴포넌트 정의

일반적으로 소프트웨어 아키텍처는 여러 개의 아키텍처 뷰 (Architecture View)를 사용하여 명세한다[19][20]. 대표적인 아키텍처 뷰로는 기능(Functional) 뷰, 정보(Information) 뷰, 동시(Concurrency) 뷰, 개발(Development) 뷰, 배치(Deployment) 뷰, 운영(Operational) 뷰가 있다[20]. 활동 2에서 선택된 여러 개의 아키텍처 스타일들을 조합하고, 각 아키텍처 스타일에 적절한 컴포넌트를 할당하면 아키텍처 뷰를 정의할 수 있게 된다.

먼저, 선정된 아키텍처 스타일은 아키텍처 뷰의 정의에 기반이 된다. 예를 들어, Balanced MVC 아키텍처 스타일은 클라이언트와 서버에 최적화된 기능 배포를 위한 것으로, 기능 뷰, 정보 뷰, 배치 뷰 정의에 근간이 되고, 클라이언트와 서버 간의 정보 교류 및 연동 기법을 정의하는 개발 뷰 정의에 기반이 된다. 그러나, 아키텍처 스타일이 모든 종류의 아키텍처 뷰의 정의에 영향을 미치는 것은 아니다.

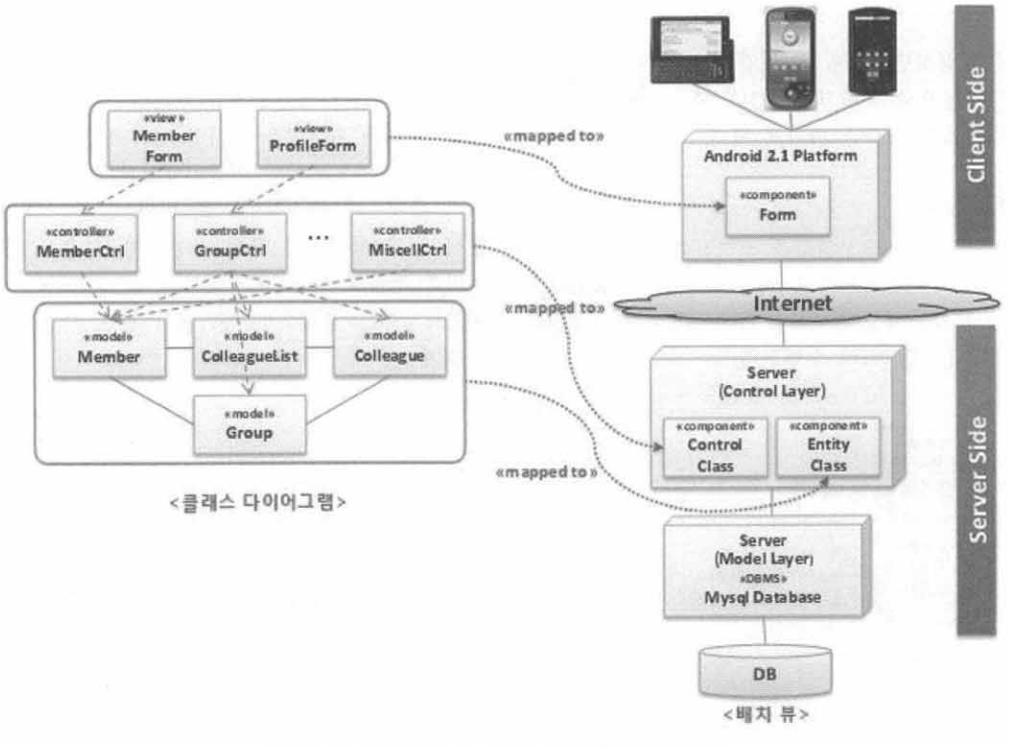
그리고, 기능 뷰와 배치 뷰의 경우는 각 컴포넌트에 해당되는 클래스들을 할당한다. 따라서, 객체 모델에 나타난 클래스를 관련된 컴포넌트에 배정한다. 이 과정에서, 안드로이드 어플리케이션의 경우, 자원 활용의 효율성을 위해서 로드 밸런싱 결과에 따라 클래스를 배정하면 된다. (그림 6)은 이러한 과정을 거쳐서 나온 Balanced MVC 아키텍처를 반영한 배치 뷰를 나타낸다.

4.3 P6. 사용자 인터페이스 설계

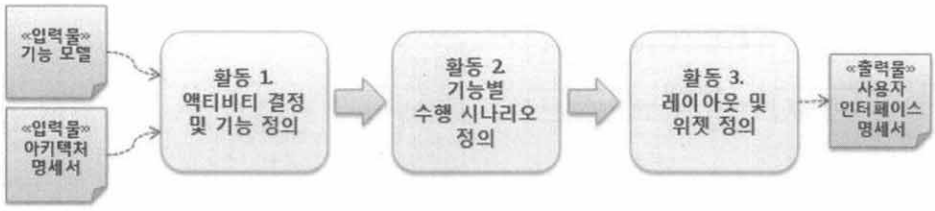
사용자 인터페이스는사용자에게 정보를 출력하거나 입력을 통해 정보를 얻는 역할을 하는 MVC 아키텍처의 뷰에 해당한다. 화면 제약성의 단점을 고려해 (그림 7)와 같이 세 가지 활동을 통해 안드로이드에 특화된 사용자 인터페이스 설계 지침을 제공한다.

4.3.1 활동 1. 액티비티 결정 및 기능 정의

액티비티는 안드로이드에서 뷰의 기능을 포함하고 있는 컴포넌트로, 기능 모델과 아키텍처 명세서를 통해 추출한다. 다음은 입력물에 따른 액티비티 추출 과정이다.



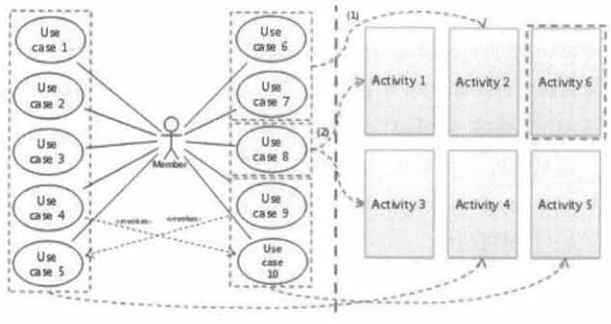
(그림 6) 아키텍처 스타일에 컴포넌트 할당 과정



(그림 7) Android 앱 사용자 인터페이스 설계 과정

- 기능 모델: 각 기능을 액티비티로 추출한다. (그림 8)의 (1), (2)와 같이 필요에 따라 하나의 기능에서 여러 액티비티가 추출될 수 있으며, 여러 기능에서 하나의 액티비티가 추출되는 경우도 있다.
- 아키텍처 명세서: 아키텍처 설계시 구분된 기능 그룹에 따라 액티비티를 분류한다.
- 기능 모델 명세서: 기능 흐름에 따라 각 기능 그룹을 연결 할 수 있는 액티비티를 추출한다.

(그림 8)은 기능 모델을 이용한 액티비티의 추출 과정을 보여주고 있으며, 기능 모델과 연결이 없는 Activity6과 같은 액티비티는 아키텍처 명세서와 기능 모델 명세서를 통해 추출된다.



(그림 8) 기능 모델과 액티비티의 관계

효율적인 사용자 인터페이스 설계를 위해 다음과 같은 지침을 제안한다.

- 작은 화면에 출력할 수 있도록 정보의 양을 최소화하고 정보가 많다면 액티비티를 분할한다.
- 메모리의 효율적인 사용을 위해 비슷한 기능, 비슷한 모습을 한 사용자 인터페이스는 재사용한다.
- 효과적인 화면 출력을 위해 추가적인 기능 또는 화면 이동을 위한 버튼은 메뉴를 이용한다.
- 입력 기능을 가진 경우 키보드의 화면 출력을 고려한다.
- 2D/3D 그래픽을 지원하지만 과도한 그래픽 사용은 피한다.

이에 따라 액티비티를 결정하고 결정 사항을 <표 2>와 같은 양식을 이용하여 설계서에 작성한다.

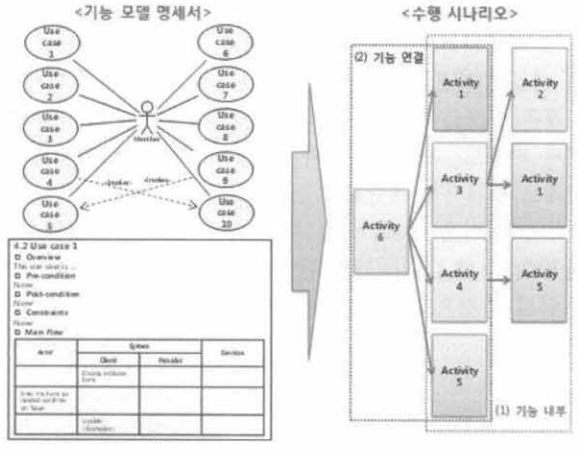
〈표 2〉 액티비티 및 기능 명세 양식

액티비티 이름	기능
Activity 1	기능 설명
...	...

4.3.2 활동 2. 수행 시나리오 정의

수행 시나리오는 어플리케이션의 기능 수행을 위한 흐름으로 활동 1에서 추출된 액티비티를 바탕으로 정의되며 다음과 같이 두 개의 시나리오로 구성된다.

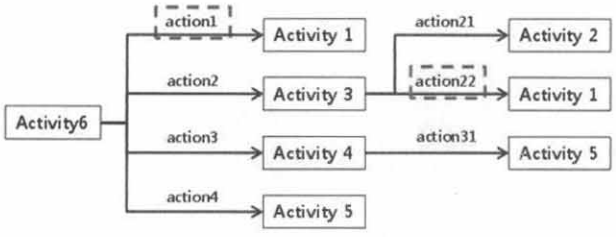
- 기능 내부 수행 시나리오: (그림 9)의 (1)과 같이 기능 모델 명세서에 따라 정의한 액티비티들의 흐름을 나타낸다.
- 기능 연결 수행 시나리오: (그림 9)의 (2)와 같이 정의된 기능 내부 수행 시나리오를 연결하는 역할로 기능 모델 명세서와 아키텍처 명세서에 따라 정의한다.



(그림 9) 수행 시나리오 추출

수행 시나리오는 (그림 9)의 수행 시나리오에서 보는 것과 같이 기능 연결 수행 시나리오로 시작된다. 기능 연결 시나리오로부터 연결된 기능 내부 수행 시나리오는 각 기능의 수행 순서를 의미한다. 시나리오 정의 시, 안드로이드 어플리케이션은 다른 어플리케이션이 가진 액티비티를 사용할 수 있으므로 이미 존재하는 같은 기능의 액티비티가 있다면 그것의 재사용을 고려해야 한다.

수행 시나리오는 각 액티비티를 호출하는 과정을 표현하고 있다. 안드로이드는 컴포넌트를 호출함에 있어서 인텐트를 사용한다 따라서 액티비티 호출을 위해 인테트를 설계해야 한다. (그림 10)은 각 액티비티를 호출하기 위한 액션에 대한 설계를 보여준다. 인텐트는 기본적으로 액션으로 그 호출을 구분하므로, Activity 6에서 Activity 1을 호출하기 위해서 action1라는 액션을 정의하였다. 또한 액티비티를 재사용하기 위해 Activiy 3에서 Activity 1을 호출하기 위해



(그림 10) 액티비티 호출을 위한 인텐트 설계 예시

서는 action22라는 액션을 정의하여 같은 액티비티를 호출하지만 다른 기능을 수행하도록 설계할 수 있다.

4.3.3 활동 3. 레이아웃 및 위젯 정의

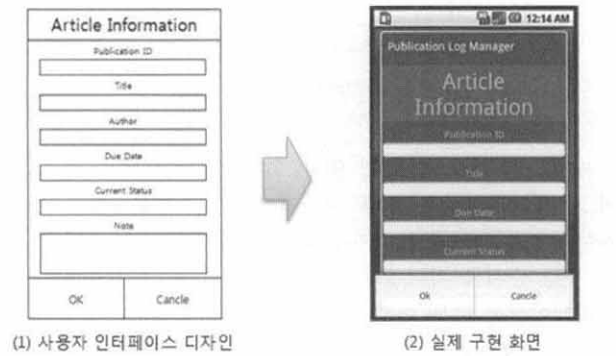
액티비티를 구성하는 가장 큰 틀인 레이아웃을 결정하고 그 틀 안에 뷰의 기능 요소들인 위젯을 추가함으로써 완성된다. 레이아웃과 위젯 정의 과정은 다음과 같다.

1) 액티비티에서 사용할 레이아웃을 결정: 화면을 구성하는 특성을 반영한 레이아웃을 결정한다. 만약 한 액티비티에 서로 다른 레이아웃의 특징을 가진 부분이 있다면 레이아웃 안에 레이아웃을 포함되는 구조로 설계하며 (그림 11)의 (3)과 같이 리스트에 명세한다.

2) 액티비티의 기능에 맞는 위젯 결정: 액티비티의 기능에 따라 위젯을 결정하여, (그림 11)의 (3)과 같이 명세한다. 위젯은 안드로이드가 제공하는 것 이외에도 직접 작성한 위젯의 사용도 고려해야 한다. 직접작성시 해당 위젯에 대한 설계도 명세서에 포함한다.

3) 레이아웃 위에 위젯 배치: 선택된 레이아웃과 위젯을 배치해 (그림 11)의 (1)과 같이 액티비티를 구성한다.

위의 과정으로 설계된 액티비티는 (그림 11)의 (3)과 같은 화면으로 구현된다.



(1) 사용자 인터페이스 디자인 (2) 실제 구현 화면

뷰 이름	레이아웃	위젯
Article Information	LinearLayout	TextView, EditView, Menu
...	...	...

(3) 레이아웃과 위젯 리스트

(그림 11) 사용자 인터페이스 디자인 및 구현

4.4 P7. 상세 객체 모델링

안드로이드 어플리케이션은 기존 어플리케이션과는 달리 네 가지의 어플리케이션 컴포넌트로 이루어져있다. 따라서,

(그림 12) 상세 객체 모델 작성 과정

(그림 12)와 같이 두 가지 활동을 수행하여 기존의 상세 객체 모델링에 어플리케이션 컴포넌트를 추가하고 명시해야한다.

4.4.1 활동 1. 개념적 객체 모델에서 클래스 추가

우선 기존의 OOAD를 통한 상세 객체 모델링 과정과 동일하다. 하지만 클라이언트의 경우는 기존 상세 객체 모델링 과정 이후 안드로이트의 특성을 반영한 어플리케이션 컴포넌트 클래스를 추가한다.

안드로이드는 <표 3>와 같이 다른 기능을 가지는 네 가지 컴포넌트를 가지고 있으므로, 각 기능을 고려하여 적절한 컴포넌트를 선정한다. MVC 아키텍처 스타일에 따라 설계된 어플리케이션에서는 다음과 같이 분류할 수 있다.

- 액티비티: 사용자 인터페이스를 담당하는 뷰 레이어와 센서, 카메라 등의 시스템에 직접으로 접근해야하는 컨트롤 레이어의 기능들이 해당한다.
- 서비스: 지속적으로 수행되어야 하는 컨트롤 레이어의 기능이 해당된다.
- 컨텐트 프로바이더: 다른 어플리케이션과의 데이터 공유가 필요한 모델 레이어의 기능이 해당된다.
- 브로드케스트 리시버: 시스템에서 발생된 이벤트를 핸들링하는 컨트롤 레이어의 기능이 해당된다.

〈표 3〉 안드로이드 어플리케이션 컴포넌트의 기능

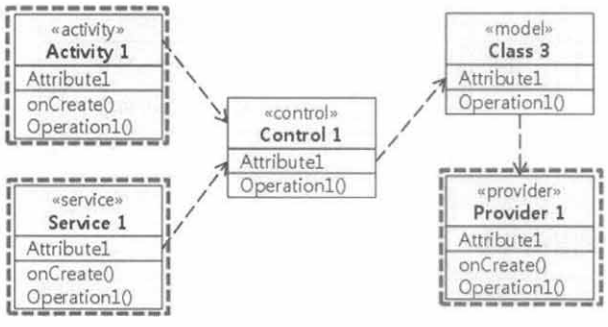
어플리케이션 컴포넌트	기능
액티비티	사용자 인터페이스를 포함, 사용자와 직접 소통하며 기능 수행
서비스	백그라운드에서 지속적 기능 수행
컨텐츠 프로바이더	사용자가 사용하는 데이터를 저장 및 공유
브로드케스트 리시버	전화나 문자, 전원 낮음과 같은 이벤트 처리를 위한 정보를 전달

추가된 안드로이드 어플리케이션 컴포넌트 별로 액티비티는 ≪activity≫, 서비스는 ≪service≫, 컨텐트 프로바이더는 ≪provider≫, 브로드케스트 리시버는 ≪receiver≫의 스테레오 타입을 사용하여, 클래스 다이어그램에 추가된 클래스의 기능을 정확히 명시한다.

4.4.2 활동 2. 생명 주기에 따른 메소드 추가

안드로이드 어플리케이션을 구성하는 네 가지의 컴포넌트 중 액티비티, 서비스, 컨텐트 프로바이더는 생명주기를 가지고 있으며 시스템에 의해 관리된다. 또한 생명주기에 따라 실행되는 메소드가 정의되어 있어 이것들을 재정의하여 기능 사용뿐만 아니라 효과적으로 리소스를 사용할 수 있다.

(그림 13)는 상세 클래스 다이어그램 작성 과정을 거쳐 추출된 클래스 다이어그램의 예제이다. Balanced MVC 아키텍처 스타일을 적용한 클라이언트 부분으로 아키텍처에 명시된 컨트롤 클래스가 추가되었다. 또한 어플리케이션 컴포넌트 가 스테레오 타입과 함께 추가된 것을 볼 수 있다.



(그림 13) 상세 클래스 다이어그램 작성 예제 (클라이언트)

4.5 P8. 상세 동적 모델링

서비스 기반 안드로이드 어플리케이션 특성에 따라 기능 수행, 어플리케이션 컴포넌트의 생명주기를 고려해야 한다. 또한 일부 기능을 서비스로 분리해 사용하기 때문에 그에 따른 서비스 이용 방법에 대한 결정이 필요하다. 그러므로, 본 활동을 (그림 14)와 같이 세 가지 활동을 거쳐 수행한다.

(그림 14) 상세 동적 모델 작성 과정

4.5.1 활동 1. 상세 클래스 다이어그램에 따라 정제

상세 객체 모델링 과정을 통해 안드로이드에 맞게 클래스의 추가와 삭제, 병합, 분할이 이루어진다. 따라서 객체 모델을 바탕으로 하고 있는 동적 모델도 수정되어야 한다. 추가된 클래스에 대한 부분을 확장하고 빌딩블록의 상태에 따라 호출되는 생명주기 관련 메소드의 호출을 추가한다.

4.5.2 활동 2. 컴포넌트의 생명 주기 확인

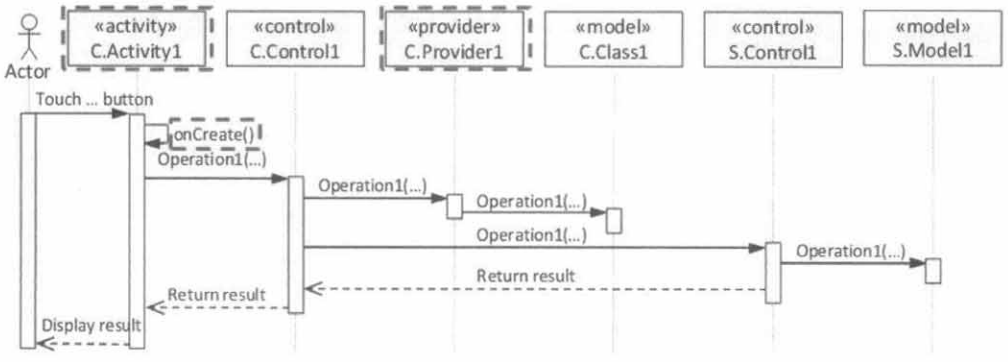
안드로이드의 컴포넌트 중 액티비티와 서비스는 각각 생명주기를 가지고 있다. 생명주기는 컴포넌트의 상태에 따라 호출되는 메소드를 이용하여 리소스와 데이터를 관리할 수 있음을 뜻한다. 따라서 이러한 생명주기에 따라 컴포넌트를 관리해 메모리를 효율적으로 사용하고 데이터 손실을 방지해야 한다.

(그림 15)은 상세 시퀀스 다이어그램으로 기존 OOAD의 시퀀스 다이어그램과 다른 점은 각 오브젝트에 안드로이드의 컴포넌트가 추가되어 있으며 (그림 15)에 표시된 것처럼 사용자가 호출하지 않아도 자동으로 시스템이 호출하는 메소드가 존재한다는 것이다. 따라서 각 클래스 오브젝트에 대해 클래스 다이어그램 작성과 마찬가지로 ≪activity≫, ≪service≫, ≪provider≫, ≪receiver≫로 스테레오 타입을 표시하고 각 컴포넌트의 상태 변화 시 호출되는 메소드를 표시함으로써 더 효과적인 리소스를 관리할 수 있다.

4.5.3 활동 3. 원격 서비스 이용 방법 결정

안드로이드의 어플리케이션은 리소스 사용의 최소화를 위해 Socket, SOAP, RESTful 기술을 이용한 원격 서비스를 사용하게 된다.

<표 4>은 세 가지 기술을 비교한 결과를 보여준다. "안드로이드 제공" 열은 안드로이드가 라이브러리 제공하고 있는지의 여부를 보여준다. 이를 제공하고 있지 않다면 외부 라이브러리를 추가해야 하기 때문에 오버헤드가 발생한다. "호출방향" 열은 클라이언트와 서버 간의 지원 가능한 호출 방향을 나열한다. "매개변수 전달" 열은 서비스로 전달할 수 있는 데이터 타입의 종류를 보여준다. 데이터 타입을 라이브러리에서 제공하는 경우는 '제공', 제공하지 않아 추가 구현이 필요한 경우는 '구현'이라 표시한다.



(그림 15) 상세 시퀀스 다이어그램 작성 예제

〈표 4〉 원격 서비스 사용 기술 비교

	안드로이드 제공	호출 방향		매개변수 전달
		클라이언트 -> 서버	서버 -> 클라이언트	
Socket	○	○	○	기본 데이터 타입: 제공 사용자 정의 타입: 제공
SOAP	×	○ (외부 라이브러리 추가 시)	×	기본 데이터 타입: 제공 사용자 정의 타입: 구현
RESTful	△ (클라이언트용)	○	○ (외부 라이브러리 추가 시)	기본 데이터 타입: 제공 사용자 정의 타입: 구현

〈표 5〉 원격 서비스 사용 명세 예

기능	제공/구현	사용기술	호출방향	사용 매개변수
기능 1	구현	RESTful	클라이언트->서버	기본 데이터 타입
기능 2	구현	Socket	Both	사용자 정의 타입
...	...	...	...	...

〈표 6〉 기능 로드 밸런싱 결과표

클래스 이름	오퍼레이션 이름	배치 장소		근 거
		클라이언트	서버	
Member	retrieveMember		✓	여러 클라이언트가 공유하는 데이터 관리하므로
	getTextMessage		✓	사용자에게 전달되지 않은 메시지를 서버가 저장/관리하고 있으므로

위의 원격 서비스의 특성들을 파악하고 서비스 기반 어플리케이션 설계 시 고려해야 한다. 따라서 서비스 기반 어플리케이션 설계 시 <표 5>와 같이 사용할 원격 서비스 사용 명세를 한 후 적당한 사용 기술을 결정한다.

5. 사례연구

본 장에서는 4장에서 제안한 설계 프로세스를 이용해 Mobile Mate Service(MMS)를 설계하고 구현한 내용을 설명한다. MMS는 안드로이드 디바이스에서 구동하는 위치 기반 소셜 네트워킹 서비스이다. 다음과 같이 크게 4가지 기능을 나누어져 있다.

- 사용자 관리: 사용자가 서비스를 이용하기 위해 가입하고 그 정보들을 관리할 수 있는 기능이다.
- 사용자 추가 정보 관리: 사용자의 추가 정보인 취미와 자신의 위치의 공개 여부의 관리 기능이다.
- 그룹 관리: 다른 사용자를 초대하고 그룹으로 분류하는 기능이다. 따라서 그룹을 관리할 수 있으며 그룹에 해당하는 사용자들의 위치를 지도에 표시하거나 다른 사용자와의 거리를 계산할 수 있는 기능을 포함한다.
- 기타 기능: 사용자의 로그인, 로그아웃, 도움말 기능이다.

이 기능들을 통해 자신의 동료들을 관리하며 주변에 있는 동료들을 쉽게 찾고 연락할 수 있다.

5.1 P3. 개념적 객체 모델링 및 A4. 개념적 동적 모델링

개념적 객체 모델은 클라이언트 부분에서 5개와 서버 부분에서 7개로 나누어 설계되었다. 서버의 클래스들은 모든 정보들을 저장하고 있기 때문에 모든 부분에 대한 클래스가 필요하지만 클라이언트의 경우 어플리케이션 구동 시 필요한 정보를 서버에서 전송 받아 사용하기 때문에 일부분만을 포함한다.

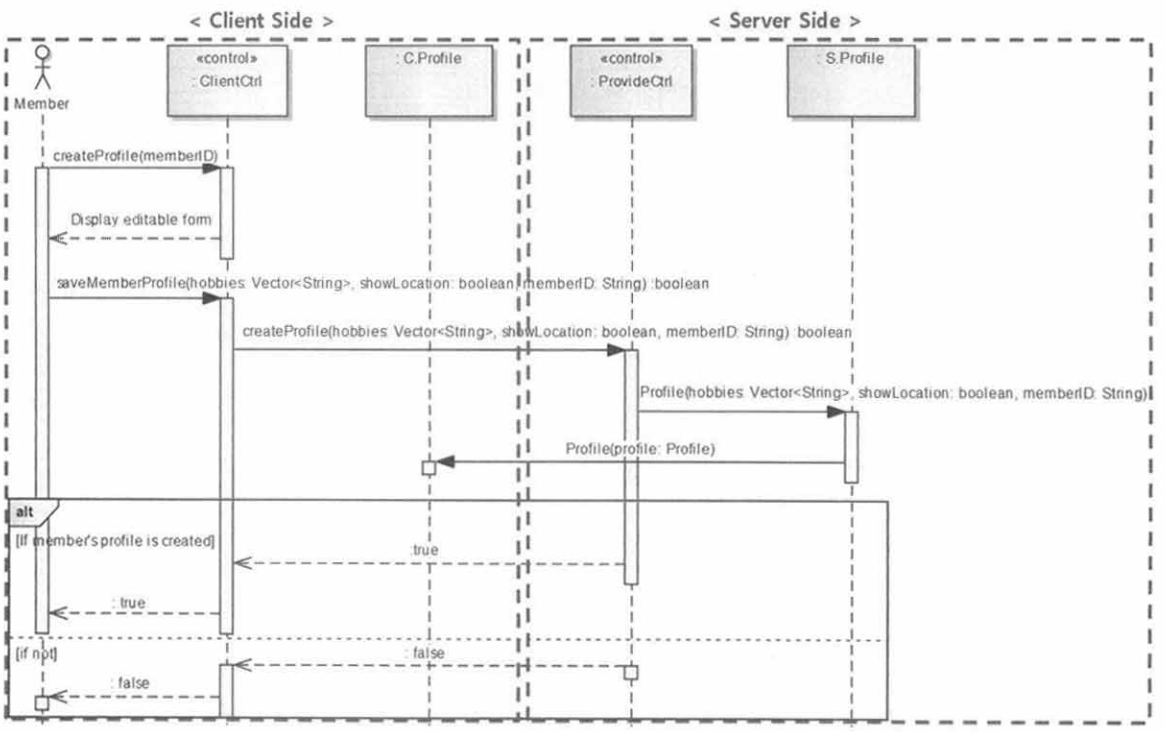
MMS에서는 대부분의 기능을 서버로 배치하고 클라이언트에는 서버를 활용할 수 있을 정도의 기능만을 배치하였다. <표 6>는 Member 클래스에 대한 로드 밸런싱의 결과를 보여준다.

개념적 동적 모델은 23개의 시퀀스 다이어그램으로 이루어졌으며, 기능 로드 밸런싱에 따라 (그림 16)와 같이 서버와 통신이 필요한 부분에 있어서는 호출 관계를 함께 나타낸다.

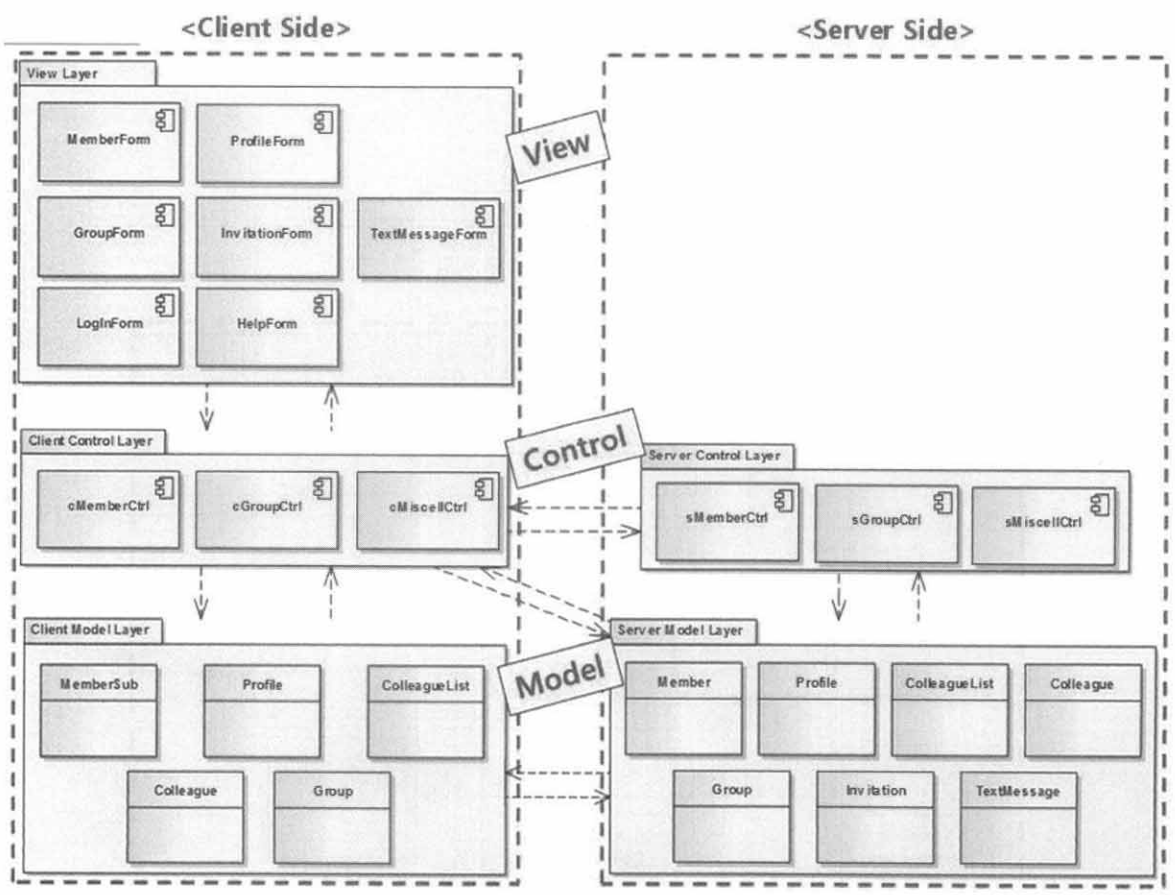
개념적 객체 모델에 따라 실제 개발에 해야 하는 클라이언트와 서버의 기능을 확실히 나눌 수 있으며 동적 모델은 객체들이 기능에 따라 어떻게 연동되는지를 보여준다. 따라서 하나의 어플리케이션의 설계가 아닌 클라이언트와 서버의 두 개의 어플리케이션에 대한 설계를 할 수 있다.

5.2 P5. 아키텍처 설계

MMS의 아키텍처는 모바일 디바이스의 제약사항을 최대



(그림 16) 사용자 추가 정보 생성의 개념적 시퀀스 다이어그램



(그림 17) 기능 뷰

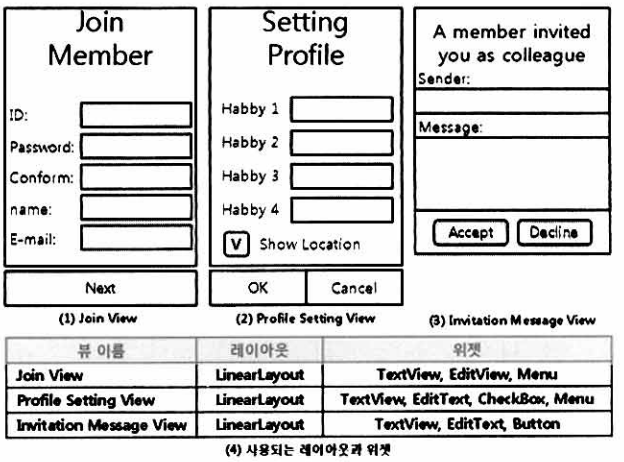
한 극복하고자 Balanced MVC 아키텍처를 적용하였다[17]. (그림 17)와 같이 클라이언트와 서버로 구분되어 있으며 클라이언트의 기능 부분들이 서버로 분할되어 있는 것을 볼 수 있다. 기능분할은 컨트롤과 모델 레이어에서 이루어졌으며 분할된 각 부분들은 서로 통신하며 정보를 교환한다.

5.3 P6. 사용자 인터페이스 설계

초기 사용자 인터페이스는 34개가 추출되었지만 재사용으로 17개로 축소되었다. 이는 같은 기능 또는 같은 사용자 인터페이스가 존재해 재사용이 가능했기 때문이다. (그림 18)는 3개의 사용자 인터페이스를 디자인한 모습과 각각에 포함되어 있는 구성 요소를 정의한 것이다.

그렇게 정의된 사용자 인터페이스를 사용할 수 있도록 (그림 19)과 같이 수행 시나리오를 정의하였다. MMS는 기능을 연결하는 부분을 메뉴로 처리해 기능 연결 수행 시나리오가 없이 기능 내부 수행 시나리오만을 정의하고 있다.

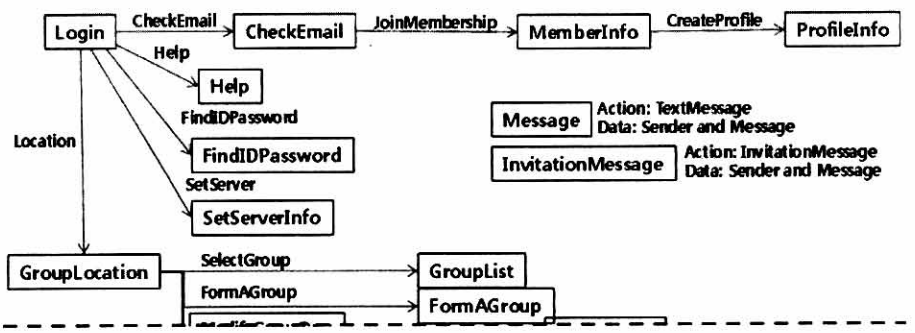
안드로이드 어플리케이션의 특성에 맞게 사용자 인터페이스를 설계하고 사용되는 구성 요소를 명시함으로써 실제 개발에 필요한 정보를 모두 제공한다. 또한 수행 시나리오를 통해 인터페이스별 호출 관계와 그것을 호출하기 위해 사용되는 인텐트 메시지를 정확히 명시함으로써 기능 흐름의 이해를 높인다.



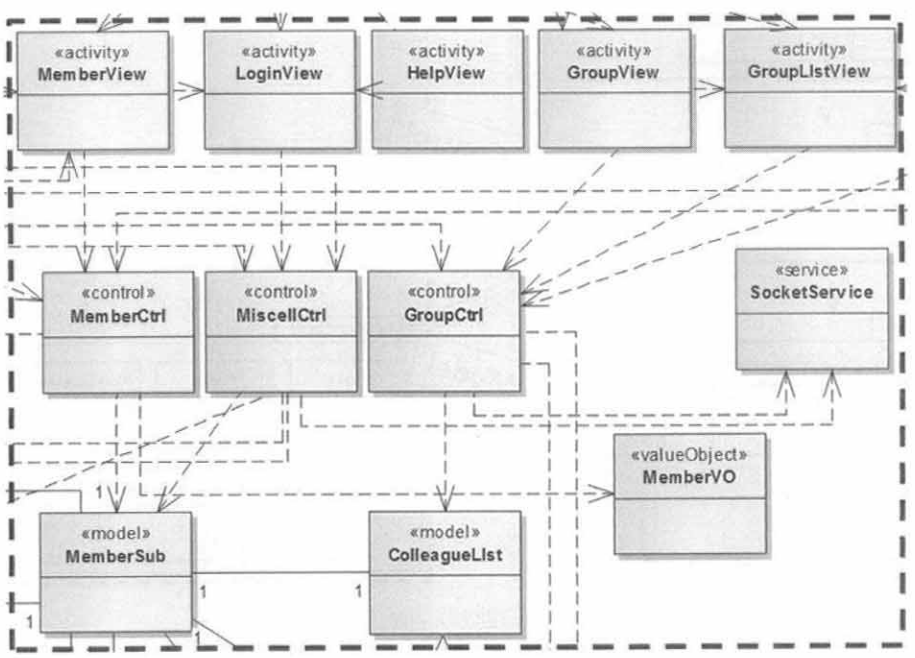
뷰 이름	레이아웃	위젯
Join View	LinearLayout	TextView, EditView, Menu
Profile Setting View	LinearLayout	TextView, EditText, CheckBox, Menu
Invitation Message View	LinearLayout	TextView, EditText, Button

(4) 사용되는 레이아웃과 위젯

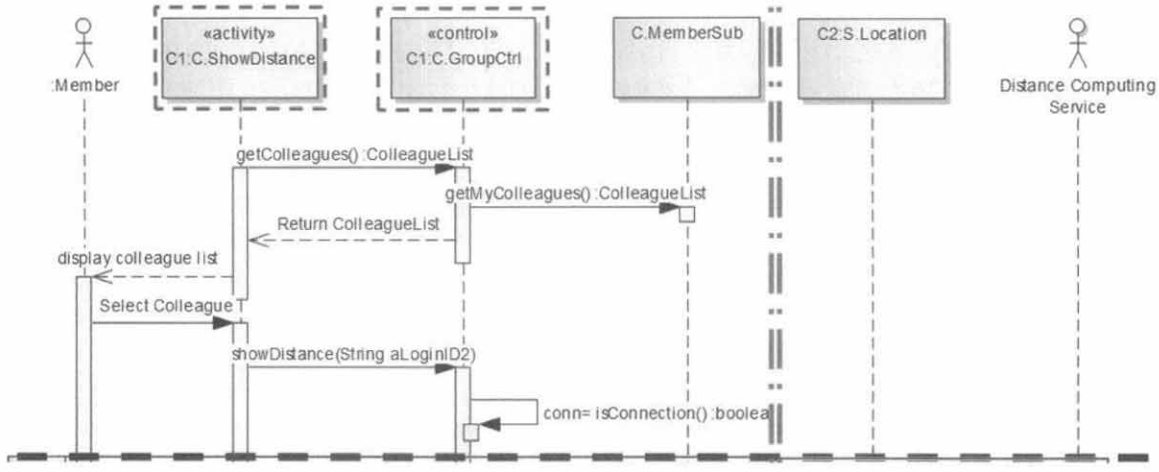
(그림 18) 사용자 인터페이스 레이아웃 및 위젯 정의



(그림 19) 수행 시나리오 정의



(그림 20) 상세 클래스 다이어그램 (클라이언트 - 상세 내용 제외)



(그림 21) Show Distance의 상세 시퀀스 다이어그램

5.4 P7. 상세 객체 모델링

개념적 객체 모델의 클라이언트 5개, 서버 7개의 클래스 다이어그램에서 상세 객체 모델에서는 클라이언트 26개, 서버 19개의 클래스다이어그램으로 증가되었다. (그림 20)은 클라이언트의 상세 클래스 다이어그램 중 일부로 안드로이드에 특화된 액티비티와 서비스 클래스를 볼 수 있다. 또한 아키텍처에서 설계된 아키텍처에 따라 클래스가 어떻게 호출되고 있는지를 나타내고 있다.

상세 객체 모델을 통해 클래스들의 관계를 인식할 수 있으며 기능들을 효과적으로 파악할 수 있다.

5.5 P8. 상세 동적 모델링

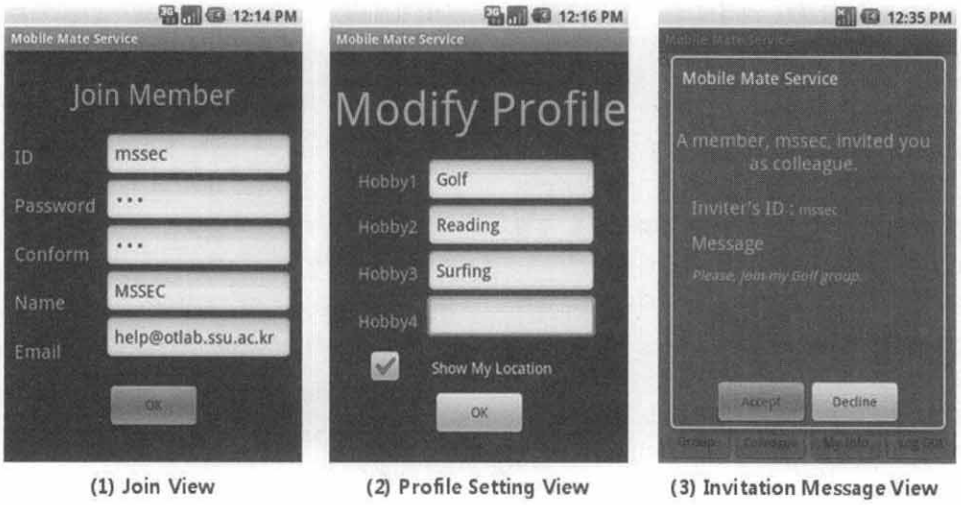
상세 동적 모델에서는 개념적 동적 모델과 앞서 설계된 상세 객체 모델까지의 결과들을 바탕으로 안드로이드 컴포넌트인 액티비티와 각 기능에 따른 컨트롤러들이 추가된다. 따라서 시퀀스 다이어그램의 개수는 23개로 동일하지만 그 내용은 더 상세하게 표현되었다. (그림 21)은 Show Distance에 대한 상세 시퀀스 다이어그램으로 추가된 액티비티와 컨트롤에 대한 내용이 추가된 모습을 볼 수 있다.

<표 7>은 기능 별로 원격 서비스를 사용하는데 특징들을 정의한 것이다. 이에 따라 MMS에서 사용하는 통신 방법을 Socket으로 결정하였다.

〈표 7〉 원격 서비스 사용 명세

기능	제공/구현	사용기술	호출방향	사용 매개변수
Login	구현	RESTful	클라이언트->서버	기본 데이터 타입
Text Message	구현	Socket	Both	사용자 정의 타입
Computing Distance	제공	RESTful	클라이언트->서버	기본 데이터 타입
…				

상세 동적 모델을 통해 기능별 흐름을 효과적으로 파악할 수 있다 또한 원격 서버의 사용 명세를 통해 서비스를 구현할 기술을 파악하고 효과적으로 결정할 수 있다.



(1) Join View (2) Profile Setting View (3) Invitation Message View

(그림 22) 설계에 따른 구현 결과

5.6 P10. 구현 및 P11. 실행 화면

앞의 과정을 통해 얻은 설계를 통해 구현된 MMS는 사용자 관리, 사용자 추가 정보 관리, 그룹관리, 기타 기능을 모두 포함하고 있다. (그림 22)은 회원 가입 화면, 추가정보 수정화면, 사용자 초대 메시지에 대한 화면을 보여주고 있다. 앞서 설계한 유저인터페이스 설계와 동일한 모습의 화면을 보여주고 있으며 서버와 클라이언트로 나누어져 요구사항에 나타난 대한 모든 기능을 수행하고 있다.

본 사례연구는 4장에서 제안한 상세 지침을 따라 어플리케이션을 설계 및 개발한 과정을 보여준다. 요구사항에서 재사용 가능한 서비스를 함께 표시함으로써 기능을 개발하는 노력을 최소화 하고 있으며 개념적 설계 과정에서부터 리소스 사용의 최소화를 위해 기능을 클라이언트와 서버로 분리하여 설계하였다. 또한 이러한 서비스 이용에 따른 아키텍처 드라이버를 추출하고 그에 맞는 아키텍처인 Balanced MVC를 적용하였다. 그 이후 과정에서는 안드로이드의 특성을 반영하여 작은 화면을 효과적으로 활용하기 위한 설계를 하였다. MMS에서는 사용자의 위치를 입력하는 것이 아니라 GPS를 이용해 위치정보를 전달하는 기능이 활용되었다. 상세 설계 부분에서는 컴포넌트의 선정과 호출, 메시지 전달에 대한 상세한 설계를 통해 명확한 설계를 통해 보다 빠른 구현을 할 수 있었다.

6. 결 론

본 논문에서는 서비스를 기반으로 안드로이드 어플리케이션을 설계하기 위한 프로세스와 상세 지침을 제안하였다. 프로세스 제안에 앞서 모바일에 특화된 안드로이드 플랫폼이 전통적인 소프트웨어의 아키텍처와 구성요소가 다르기 때문에 발생되는 이슈를 도출하였다. 도출한 이슈는 안드로이드 어플리케이션을 개발함에 있어서 해결되어야 하는 문제이기 때문에 어플리케이션을 개발하기 위한 분석과 설계 단계에서부터 이슈를 해결해 나가야 한다. 따라서 OOAD 프로세스를 따라 안드로이드 어플리케이션을 개발함에 있어서 이슈를 해결할 수 있는 단계를 파악하여 설계 프로세스를 제안하였다.

다음은 제기한 이슈들이 어떻게 프로세스에 반영되었는지를 보여준다.

- 리소스 사용 최소화는 개념적 객체 모델링와 개념적 동적 모델링, 아키텍처 설계시 반영되어 기능을 클라이언트의 기능 중 일부를 서버로 배치하며 이것을 효과적으로 수행하기 위해 로드밸런싱 에 대한 지침을 제공하였다.
- 안정적인 네트워킹은 아키텍처 설계시 반영되어 아키텍처 드라이버 도출시 필수 요소로 제안하였다.
- 서비스 설계 및 이용은 상세 동적 모델링에 반영되어, 원격 서비스를 호출는 방법을 결정하는 지침을 제공하였다.
- 효율적인 사용자 인터페이스 설계는 유저 인터페이스 설계시 반영되어, 재사용 가능한 액티비티 도출 방법과 수행 시나리오 작성법을 제공하였다.
- 어플리케이션 컴포넌트 선정은 상세 객체 모델링에 반영되어 기능에 따라 확장된 안드로이드 컴포넌트를 선정하고 그것을 구분하는 스테레오타입을 제공하였다.
- 어플리케이션 컴포넌트 호출과 메시지 전달는 유저 인터페이스 설계와 상세 동적 모델링에 반영되어 액티비티 호출시 사용되는 인텐트를 설계하고 수행 시나리오에 명세하는 것을 보여주고 실제 호출 관계를 시퀀스 다이어그램에 표시하는 것을 보여주었다
- 어플리케이션 간 데이터 공유는 상세 객체 모델에 반영되어 정보 공유에 사용되는 컨텐츠 프로바이더 사용을 정의하였다.

제안한 프로세스의 각 단계는 서비스 기반 안드로이드 어플리케이션 설계의 상세 지침이 포함되어 있으며 설계 시 필요한 명세 양식을 제공하여 명확한 설계를 할 수 있도록 제안하고 있다. 제안된 프로세스를 적용한 사례연구를 통해 이슈가 어떻게 해결되고 프로세스의 특징들을 잘 반영하고 있는지를 확인하였다. 제안된 설계 기법을 적용하여 안드로이드 서비스 어플리케이션 개발하면, 서버와 클라이언트 어플리케이션으로 구성된 다소 복잡한 구조의 모바일 어플리케이션을 적은 노력을 이용하여 효과적으로 개발할 수 있을 것으로 기대된다.

참 고 문 헌

[1] König-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] Zeidler, C., Kittl, C., and Petrovic, O., "An Integrated Product Development Process for Mobile Software," *In Proceedings of the 6th International Conference on the Management of Mobile Business (ICMB 2007)*, pp.23-30, 2007.

[4] Salmre, I., *Writing Mobile Code: Essential Software Engineering for Building Mobile Applications*, Addison-Wesley Professional, 2005 (chapter 2).

[5] Tran, Q.N.N. and Low, G., "MOBMAS: A Methodology for Ontology-Based Multi-Agent Systems Development," *Information and Software Technology (IST)*, Vol.40, No.7-8, pp.697-722, June, 2008.

[6] Abrahamsson, P., Hanhineva, A., Hulkko, H., Ihme, T., Jaalinoja, J., Korkala, M., Koskela, J., Kyllonen, P., and Salo, O., "Mobile-D: An Agile Approach for Mobile Application Development, " *In Proceedings of 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications (OOPSLA 2004)*,

pp.174-175, 2004.

[7] Jeong, Y.J., Lee, J.H., and Shin G.S., "Development Process of Mobile Application SW Based on Agile Methodology, " *In Proceedings of the 10th Internatioanl Conference on Advanced Communication Technology (ICACT 2008)*, pp.362-366, 2008.

[8] Braun, P., Eckhaus, R., "Experiences on model-driven software development for mobile applications," *In proceedings of the 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems, 2008 (ECBS 2008)*, pp.490-493, 2008.

[9] Natchetoi, Y., Kaufman, V., and Shapiro, A., "Service-oriented architecture for mobile applications," *In Proceedings of the 1st international workshop on Software architectures and mobility (SAM '08)*, pp.27-32, 2008.

[10] Ughett, M., Trucco, T., and Gotta, D., "Development of Agent-based, Peer-to-Peer Mobile Applications on ANDROID with JADE," *In Proceedings of the 2nd International Confernce on Mobile Ubiquitous Computing, Systems, Services, and Technologies (UBICOMM 2008)*, pp.287-294. 2008.

[11] Shu, X., Du, Z., and Chen, R., "Research on Mobile Location Service Design Based on Android," *In Proceedings of the 5th International Conference on Wireless Communications, Networking and Mobile Computing*, 2009.

[12] Chakrabarti S., Nordmark E., and Cohen, D., "Enterprise Mobility," Technical Report. Sun Microsystems, Inc., June, 2006.

[13] Yang, K., Ou, S., and Chen, H.H., "On Effective Offloading Services for Resource-Constrained Mobile Devices Running Heavier Mobile Internet Applications," *IEEE Communications Magazine*, Vol.46, No.1, pp.56-63, 2008.

[14] Fling, B., *Mobile Design and Development*, O'Reilly, 2009.

[15] Larman, C., *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*, Prentice Hall, 2004.

[16] Choi, S.W., and Kim, S.D., "A Quality Model for Evaluating Reusability of Services in SOA," *In Proceedings of IEEE Joint Conference on E-Commerce Technology and Enterprise Computing, E-Commerce and E-Service (IEEE CEC'08 and EEE'08)*, pp.293-298, 21-24, July, 2008.

[17] La, H. J. and Kim, S. D., "Balanced MVC Architecture for Developing Service-based Mobile Applications ," *In Proceedings of the 7th IEEE International Conference on e-Business Engineering (ICEBE 2010)*, pp.292-299, 2010.

[18] Tao, L., Fu, X., and Qian, K., *Software Architecture Design: Methodology and Styles*, Stipes Publishing L.L.C, 2006.

[19] Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Merson, P., Nord, R., and Stafford, J., *Documenting Software Architectures Views and Beyond*, Addison-Wesley, 2003.

[20] Rozanski, N. and Woods, E., *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives*, Addison-Wesley, 2005.

이 호 중

e-mail : leehojoong@gmail.com
2009년 숭실대학교 컴퓨터학부(학사)
2009년~현 재 숭실대학교 컴퓨터학과 석사과정
관심분야 : 서비스 지향 아키텍처, 모바일 컴퓨팅

라 현 정

e-mail : hjla80@gmail.com
2003년 경희대학교 우주과학과(이학사)
2006년 숭실대학교 컴퓨터학과(공학석사)
2011년 숭실대학교 컴퓨터학과(공학박사)
2011년~현 재 숭실대학교 모바일 서비스 소프트웨어공학 센터 연구 교수
관심분야 : 서비스 지향 아키텍처, 클라우드 컴퓨팅, 모바일 서비스

금 창 섭

e-mail : cskeum@etri.re.kr
1992년 2월 서울시립대학교 전산통계학과 (학사)
2005년 8월 카네기멜론 대학(석사, 소프트웨어 공학)
1994년 2월~현 재 한국전자통신연구원 서비스융합연구팀 책임연구원
2001년 7월~12월 University of Florida 방문 연구원
2006년~2007년 3GPP/ETSI CT5 "Multimedia Multicast Session Management Web Service" Rapporteur
2006년~현 재 지식경제부 IT 멘토 위원
2006년~2011년 Who's Who in the World 등재
관심 분야 : 소프트웨어 테스팅, 소프트웨어 아키텍처, 서비스 지향 아키텍처, 서비스 딜리버리 플랫폼

김 수 동

e-mail : sdkim777@gmail.com

1984년 Northeast Missouri State University 전산학(학사)

1988년~1991년 The University of Iowa 전산학(석사/박사)

1991년~1993년 한국통신 연구개발단 선임연구원

1994년~1995년 현재전자 소프트웨어연구소 책임연구원

1995년 9월~현 재 숭실대학교 컴퓨터학부 교수

관심분야 : S/W공학, 소프트웨어 아키텍처, 클라우드 서비스, 모바일 컴퓨팅