제36회 한국정보처리학회 추계학술발표대회 논문집 제18권 제2호 (2011. 11)

MDA+SOA 응용 플랫폼 독립적인 스마트폰 어플리케이션 개발에 대한 연구

탁지우*, 김행곤**
대구가톨릭대학교 컴퓨터정보통신공학과
e-mail:{lebbenle*, hangkon**}@cu.ac.kr

A Study on Platform Independent Smart Phone Application Development Using MDA+SOA

Ji-Uoo Tak, Haeng-Kon Kim
Dept of Computer information & Communication Engineering
Catholic Univ. of Daegu, Korea

요 약

스마트폰 작동 플랫폼의 다양화와 빠른 진화속도로 인하여 어플리케이션의 개발 및 유지보수에 많은 시간과 노력이 요구되고 있다. 스마트폰 어플리케이션 개발은 플랫폼(iOS, Android 및 Window) 별로 각각 진행하고 있다. 따라서 한 어플리케이션을 다른 플랫폼에서 사용하기 위해서는 그 플랫폼에 맞게 다시 개발을 해야 하는 번거로움을 갖고 있다. 본 논문에서는 플랫폼 독립적인 스마트폰 어플리케이션 개발을 할 수 있도록 MDA와 SOA를 응용한 프레임워크를 제시하여 플랫폼에 종속되지 않고 어플리케이션을 개발할 수 있는 방법을 제시한다.

1. 서론

최근 모바일 산업은 음성통신 위주에서 데이터 통신위주로 산업의 중심이 옮겨오고 있고 산업 패러다임의 변화에 따라 음성 중심의 휴대폰도 인터넷 통신을 위한 기기로 탈바꿈하여 스마트폰의 시대를 맞고 있다[1].

꾸준한 기술의 발전과 사용자의 편의를 위한 요구사항을 충족시킴에 따라 여러 멀티미디어 기기들의 대부분의 기능이 휴대폰에 흡수되었고 휴대가 용이하도록 소형화되기를 요구하여 오늘날의 스마트폰으로 발전하게 되었다. 컴퓨터로만 하던 작업들은 휴대폰으로 급속도로 이동하고 있으며 기존의 휴대폰과 달리 스마트폰은 사용자가 필요한 어플리케이션을 설치함으로써 쉽게 휴대폰에 기능을 추가할 수 있는 커스터마이즈(Customize) 휴대폰을 만들 수 있다.

다양한 기능을 탑재한 휴대폰의 발전 흐름에 발맞추어 네트워크의 기술도 발전하였다. 네트워크 기술은 휴대폰을 항상 인터넷에 연결하여 다양한 정보를 검색하고 찾을 수 있고, 많은 서비스를 사용자가 제공받을 수 있다.

현재 컴퓨터의 어플리케이션은 단순히 컴퓨터 자체의 기능만을 이용하는 것이 아니라 서비스 기반으로 변화되어 항상 연결되어 있는 인터넷을 활용하고 있다. 이러한 기술들은 모바일 디바이스에도 적용될 수 있는데 모바일 디바이스의 특성에 맞게 적용되어야 할 것이다.

모바일 디바이스의 특징에는 '제한된 리소스', '물리적 위험', '풍부한 네트워크 연결성', '한정된 배터리 용량'이 있는데 여기서 모바일 어플리케이션을 개발함에 있어서 가장 많이 고려되는 것은 '제한된 리소스'이다. 컴퓨터의 기능을 모바일 디바이스가 일부 가지고 있다고는 하지만 컴퓨터의 자원에 비해 지극히 소형화 된 모바일 디바이스는 그 만큼 여러 자원들에 제한을 갖는다. 이러한 특성 때문에 컴퓨터 어플리케이션을 모바일 디바이스에 똑같이 적용을 할 수 없어 모바일 디바이스에 맞게 개발을 해야 한다[2].

이러한 모바일 디바이스에 적합한 어플리케이션을 개발하기 위해서 SOA (Service-Oriented Architecture)와 MDA(Model-Driven Architecture)를 접목하여 모바일 디바이스에 적합한 아키텍처를 제안하고 플랫폼에 종속되지 않아 재사용 가능한 프레임워크를 구성하는 것을 목표로 한다.

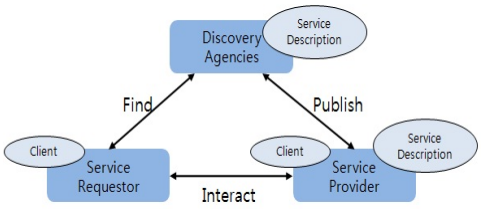
2. 관련연구

2.1 SOA(Service-Oriented Architecture)

기업 인프라의 복잡성과 유지비용을 최소화하고 기업의 생산성을 높일 것으로 전망되어 차세대 소프트웨어 아키텍처로 주목받으며 소프트웨어의 구성요소 및 이들의 가시적 속성과 이들의 관계로 구성된 시스템 구조를 의미한다. SOA기반의 차세대 소프트웨어 기술들의 등장은 기존 소프트웨어 시장의 침체된 상황을 정리하고 신규시장의 확대와 시장의 활성화 유도하고 최근의 비즈니스 환경은 과거 독립적인 조직 및 프로세스에 의해 주도되는 수직적 통합에서 고객, 공급자, 파트너 등 다수 기업과의 관계적 협업 관계가 중시되는 수평적 통합 환경으로 변화

하고 있다. 이러한 관계적 협업이 중시되는 비즈니스 환경에서 경쟁력 있는 기업은 급변하는 시장요구에 민첩하게 대응할 수 있어야 한다. 이러한 비즈니스 요구에 효율적으로 대응하기 위한 기업 IT 아키텍처로 대표되는 것이 SOA이다. SOA는 전통적인 프로그램 중심의 설계·개발 방식에서 비즈니스 프로세스 관점에서 재활용 가능한 단위로 서비스를 설계·개발하게 함으로써 특정 프로세스나 서비스 변경 또는 내부·외부 시스템과의 비즈니스 통합 시 효율적이고 빠른 대응이 가능하다는 점에서 그 의미가 깊다

SOA는 특정 기술이나 플랫폼에 종속되지 않고 느슨한 결합(Loosely Coupled)을 가지고 상호 연동할 수 있는 서비스들의 조합으로 어플리케이션 개발을 가능하게 하는 정보 시스템 아키텍처이다. SOA는 그림1과 같이 구조를 가지며 한 덩어리의 방대한 코드로 이루어진 어플리케이션들을 각각 개발하는 대신 각각의 비즈니스 기능을 수행하는 서비스를 구성하고 이 서비스를 조합하거나 분리함으로써 비즈니스 프로세스들을 구현할 수 있게 하는 정보 시스템 구축을 목표로 한다[3].

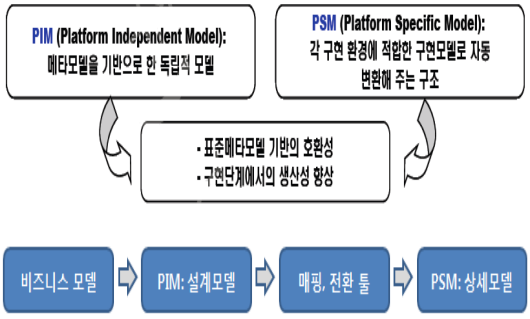


(그림 1) Service Oriented Architecture

2.2 MDA(Model Driven Architecture)

소프트웨어 애플리케이션은 금융에서 국방에 이르기까지 광범위한 영역의 문제들을 대상으로 만들어진다. 게다가 이들 애플리케이션의 복잡성 또한 더욱 커지고 있다. 또한, 소프트웨어 기술 자체의 복잡성이 폭발적으로 증가하고 있으며 개발자들은 HTML, XML, WML, 다계층 아키텍처, J2SE, J2EE, J2ME, .NET, 컴포넌트와 오브젝트 모델, 커넥터, 메시지 브로커 등과 같은 여러 가지 기술을 익혀야만 한다. 소프트웨어 개발의 주요 과제는 생산성을 증대하면서 동시에 유연성과 표준화를 조화시키는 것이 필요하다. MDA가 소개되기 전까지 기존 언어는 이런 요구사항 중 한 두 가지는 맞추었지만 모두를 만족시키지는 못하였다. MDA의 출현으로, 완벽한 유연성을 유지하면서, 표준 환경 내에서 높은 생산성이 가시적으로 현실화되었다. 또한, 수동적으로 작성되는 코드가 적어지고 최상의 사례를 적용할 수 있는 패턴을 사용함으로써 애플리케이션 품질도 개선된다[4].

MDA(Model Driven Architecture)는 OMG(Object Management Group)에서 공식 발표된 이후로 소프트웨어 업계에서는 매우 유망한 패러다임으로 받아들여지고 있다. MDA가 소프트웨어의 개발과 유지보수 업무 방식에 대변혁을 가져오는 잠재성을 가졌다고 말할 수 있으며 MDA가 빠른 속도로 대중화되어가고 있다. MDA는 새로운 소프트웨어를 개발하는데 있어 더 큰 생산성을 얻기 위해서 필요하다. 그림2는 MDA 모델을 나타내며 PIM과 PSM 그리고 표준 메타모델 기반의 호환성과 구현단계에서의 생산성 향상을 위한 구조를 가진다.



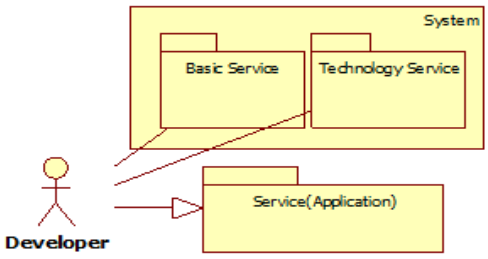
(그림 2) MDA Model

3. 플랫폼 독립적 어플리케이션 프레임워크 개발

3.1 프레임워크 아키텍처

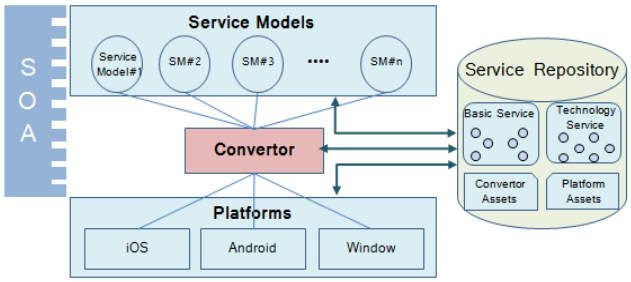
현재 스마트폰 어플리케이션의 개발에 있어서 플랫폼은 아이폰, 안드로이드, 윈도우 모바일 등으로 구분되며, 각 플랫폼 별로 어플리케이션을 개발하고 전용 앱스토어를 통해 다운로드를 받아 사용하는 방식이다[5].

이러한 현재의 상황을 통해 본 논문에서 어플리케이션을 플랫폼별로 개발을 해야 하는 작업의 번거로움과 개발 비용의 절감을 위해 플랫폼에 독립적으로 어플리케이션을 개발할 수 있는 프레임워크를 설계 한다.



(그림 3) 개발자 use case

그림 3에서 개발자가 어플리케이션을 개발함에 있어서 개발자는 기본서비스(Basic Service)와 기술 서비스(Technology Service)에 저장된 각각의 서비스들에 접근을 할 수 있으며 그 서비스들을 제공받아(served) 새로운 서비스 즉, 어플리케이션을 개발할 수 있다. 이러한 것은 SOA방식에 기반 한 개발방법으로 플랫폼에 독립적인 어플리케이션 개발을 가능케 한다.



(그림 4) 플랫폼 독립적 어플리케이션 개발 프레임워크

그림 4는 본 논문에서 제시한 플랫폼에 독립적인 어플리케이션 개발을 위한 프레임워크이다. Service Repository는 서비스 저장소로써, 스마트폰에서 제공하는 여러 서비스들과 자산(assets)을 저장해 둔 저장소로 두 종류의 서비스(Basic service, Technology service)와 Convertor 접근에 필요한 Convertor assets, Platform접근에 필요한 Platform assets가 있다.

(1) Service Repository(서비스 저장소)

Service Repository의 Basic Service는 스마트폰(mobile phone)에서 기본적으로 제공되는 서비스(가장 기본적인 기능들인 통화, 문자, 모바일 웹 접속, 전화번호부, 일정, 알람, 사전 등 피쳐폰(Feature Phone)에서도 제공되는 mobile 기기의 기본적인) 기능들의 다양한 자산 저장소이다. Technology Service는 여러 기술적인 서비스들을 저장하는 저장소로 흔히 요즘 스마트폰에서 새로 제공되고 있는 서비스(음성처리, 영상처리, HCI, 위치기반, 유비쿼터스, 클라우드컴퓨팅, 이러닝 등)들의 저장소이다. Convertor Assets는 Service model이 Convertor에 접근 시 필요한 정보들(assets)을 가지고 있어 convertor접근에 용이하도록 한다.

Platform assets는 Convertor Assets과 같이 Convertor에서 각각의 Platform에 맞게 생성된 의사코드(Psedo Code)를 전달함으로써 개발자가 의사코드를 수정하여 플랫폼에 적용시켜 코딩의 부담을 줄여준다.

(2) Service Model(서비스 모델)

Service Models는 서비스 모델이 n개의 집합으로 구성되며 각 서비스 모델들은 Service Repository의 Basic Service와 Technology Service에 저장된 서비스를 제공 받아서 하나의 모델로 구성이 되며 이 모델들의 집합 즉, Service Model은 모바일 어플리케이션의 설계라고 할 수 있다.

(3) Convertor(컨버터)

Convertor에서는 의사코드(Psedo Code)가 생성되는데 Service Model들이 각각 Convertor에서 변환과정을 거치고 Platform에서는 Convertor로부터 전송받은 의사코드(Psedo Code)를 기반으로 각 플랫폼에 적절하게 적용되어 개발자가 수정작업을 거쳐 개발자에게 코딩에 대한 부담을 덜어준다.

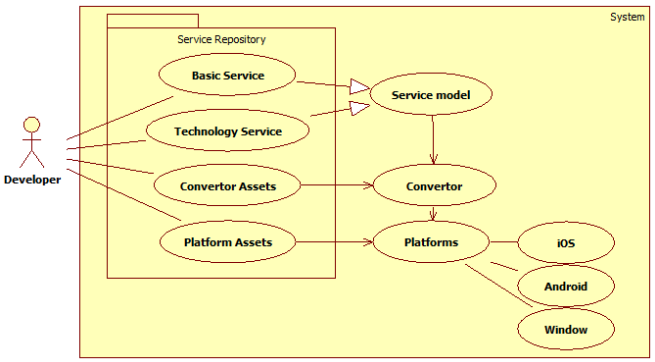
(4) Platforms(플랫폼)

Platform에서는 Convertor(컨버터)로부터 전송 받은 의사코드를 바탕으로 각 언어에 맞도록 개발자에 의해 수정이 되어 각 플랫폼에 적용되어 어플리케이션 개발을 완료한다.

3.2 프레임워크 모델링

(1) Use case

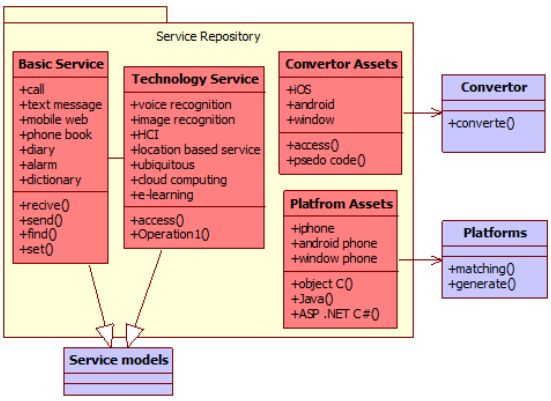
그림 5는 본 논문에서 제시한 프레임워크의 use case이다. 개발자(Developer)는 Service Repository내의 Basic, Technology, Convertor assets, Platform assets에 접근을 할 수 있고 Basic과 Technology에서 Service를 제공받아 Service Model을 구성한다. Service Model은 Convertor에서 변환되는데 convertor에 전송 시 서비스 저장소(Service Repository) 내의 Convertor Assets에서 필요한 정보를 받아서 접근한다. 각각의 서비스 모델들은 convertor에서 변환되어 의사코드로 생성되는데 이 의사코드는 바로 플랫폼으로 전송되어 각 플랫폼에 맞도록 개발자가 수정작업을 하여 어플리케이션 개발을 완료한다. 그러므로 어플리케이션 개발 시 플랫폼에 종속되지 않게 각각의 서비스들로 서비스 모델을 만들고 이 모델은 컨버터에서 의사코드를 생성하여 플랫폼에 맞는 언어로 개발자가 약간의 수정을 함으로써 개발자에게는 코딩의 부담을 줄이고 플랫폼 독립적인 어플리케이션을 개발 할 수 있다. 예를 들어 iPhone에서 구동 가능한 어플리케이션을 Android에도 개발 할 경우 Convertor로부터 생성된 의사코드를 보고 Android에 맞게 수정작업을 하면 시간절약과 개발자의 수고를 덜 수 있다.



(그림 5) usecase

(2) Class Diagram

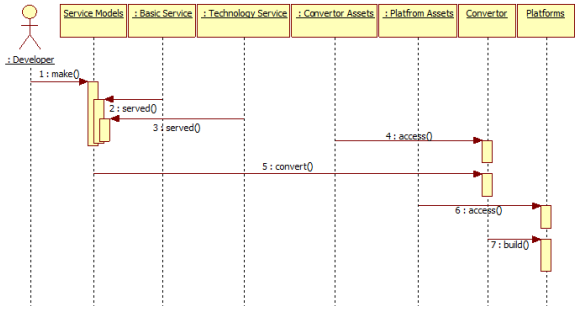
그림 6은 use case에 대한 class model로 서비스 저장소 내의 서비스(Basic&Technology Service)와 자산(Convertor &Platform Assets)이 가지는 기능과 동작(operation)의 관계를 보여준다.



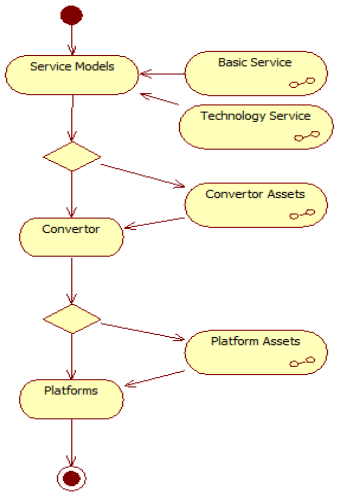
(그림 6) Class Model

(3) Sequence 및 Activity Diagram

그림 7과 8은 본 논문에서 제시한 프레임워크의 sequence diagram과 activity diagram으로 개발자가 어플리케이션의 개발을 완료할 때까지의 과정이다. Activity diagram을 보면 개발자는 서비스 모델 생성 시 기본서비스와 기술서비스에서 서비스를 제공받고 convertor 접근 시에는 convertor assets에서 필요한 자산을 제공 받아 서비스 모델로부터 의사코드를 생성하고 생성된 의사코드는 platform assets를 거쳐 각 플랫폼에 맞는 언어로 개발자에 의해 수정되며 어플리케이션 개발을 완성하게 된다.



(그림 7) Sequence Diagram



(그림 8) Activity Diagram

Activity diagram을 보면 개발자는 서비스 모델 생성 시 기본서비스와 기술서비스에서 서비스를 제공받고 convertor 접근 시에는 convertor assets에서 필요한 자산을 제공 받아 서비스 모델로부터 의사코드를 생성하고 생성된 의사코드는 platform assets를 거쳐 각 플랫폼에 맞는 언어로 개발자에 의해 수정되며 어플리케이션 개발을 완성하게 된다.

5. 결론 및 향후연구

스마트폰 어플리케이션 개발 시 어떠한 플랫폼에서 구동하게 할지를 정하여 그 플랫폼에 맞게 개발하고, 후에 다른 플랫폼에서도 구동하려 할 때 작업의 번거로움을 줄이는 것이 절대적으로 필요하다.

본 논문에서는 MDA+SOA를 기반 한 서비스 저장소와 컨버터 서비스 모델을 가지는 프레임워크를 제안하였다. 서비스 저장소에서는 어플리케이션에서 쓰이는 기본기능과 기술적 기능을 두어 개발 할 어플리케이션에 필요한 기능을 제공받아서 하나의 모델을 생성하고 각각의 모델들은 convertor에서 변환과정을 거쳐 의사코드를 생성한고 생성된 의사코드는 각각의 플랫폼에 맞도록 개발자의 수정을 거쳐 어플리케이션의 개발을 완료한다. 그리하여 새 어플리케이션을 다른 플랫폼에 구동하고자 할 때 convertor의 결과물인 의사코드를 토대로 플랫폼에 맞게 수정을 함으로써 개발 비용과 시간의 절약, 개발자의 수고를 줄이는 효과를 얻을 수 있다. 본 논문에서 제시한 서비스 저장소 내 자산의 보다 명확한 정의와 자세한 동작에 대한 연구를 계속 하고 있으며 convertor에 대한 기능을 더 확실히 하고자 한다. 최종적으로는 각각의 모델이 convertor에서 의사코드로 변환 시 각 기능을 클래스 별로 구성하고 저장하여 각 클래스를 플랫폼에 사용 시 각 클래스만 재사용하여 플랫폼에 독립적이며 개발자의 수고를 더는 재사용성이 강조된 플랫폼을 설계 및 구현하였다.

향후에는 제안한 프레임워크 모델을 통해 다양한 응용앱들을 개발, 적용하여 평가할 계획이다.

참고문헌

[1] 전종홍, 이승윤, “차세대 모바일 웹 애플리케이션 표준화 동향”, 전자통신동향분석, 제 25권, 제1호, pp100-113, 2010

[2] 이호중, 라현정, 김수동,“서비스 기반 모바일 어플리케이션의 MVC 아키텍쳐”,한국컴퓨터종합학술대회, 제37권, 제1(A)호, pp74-79, 2010

[3] 최은만, “Service Oriented Architecture와 재사용.” 정보과학회지, 정보과학회지, 제 24권, 제11호, pp32-37, 2006

[4] Gabor Karsai, Sandeep Neema, David Sharp, "Model-driven architecture for embedded software: A synopsis and an example" Science of Computer Programming, Volume 73, No.1, pp.26-38, 2008

[5] 최민, 정영식, 이용주, 정성태, 임승호, “플랫폼 독립적인 스마트폰 응용 개발을 위한 SOA기반구조”, 한국컴퓨터종합학술대회, 제 37권, 제1(B)호, pp333-336, 2010