Journal of Korea Multimedia Society Vol. 22, No. 2, February 2019(pp. 292-298)
https://doi.org/10.9717/kmms.2019.22.2.292

MDA기반 모바일 크로스 프레임워크를 위한 메타모델 설계

송유진[†], 한덕수[††], 이은주[†††]

Design MetaModel for MCF (Mobile Cross Framework) Based MDA

Yujin Song[†], Deoksoo Han[††], Eunjoo Lee[†††]

ABSTRACT

Mobile-based software development methodology has been vigorously researched from using object-oriented development methodology and component-based development methodology previous structural developing methodology. There are two types of OS in mobile platform which are android and iOS. There is a problem that the application to be developed is developed depending on the device type. To resolve this problem, first, the system structure and design method should be managed effectively. Second, a basic design guide that can be commonly adapted to the each project is required. In this paper, we define a mobile cross platform meta model based on MDA-development methodology, focusing on reusability, portability and interoperability about non - dependent part of the mobile platform. If the proposed meta-model is applied to manage the related information and all the types of Mobile-Apps become available through independent mobile app development process, henceforward, it will be much of help establishing formulaic mobile-app developmental methodology.

Key words: Mobile-based Software Development, Meta-Model, Model Driven Architecture

1. 서 론

효과적인 정보처리를 위한 다양한 소프트웨어 개발 방법론들이 꾸준히 출현하고 있다. 이른바 컴퓨터 시대가 시작되던 1970년대 소프트웨어 위기가 다가오면서 소프트웨어 개발 방법론이라는 개념이 도입된 이래 구조적 개발 방법론으로부터 객체지향 개발 방법론과 컴포넌트 기반 개발 방법론에 이르기까지 활발히 연구되었다. 현재도 소프트웨어 개발 방법론은 도메인에 따라 다양한 방법으로 연구되고 개발되고 있다[1].

최근 모바일 환경에서의 소프트웨어 개발 방법론은 기존의 컴포넌트 기반 개발 방법론을 응용하여 새롭게 연구되는 추세이다. 그러나 모바일 플랫폼의 종류가 크게 안드로이드와 아이폰 운영체제로 양분돼있어 개발하고자 하는 어플리케이션(이하 앱이라고 함)은 디바이스에 따라 종속적으로 개발이 이루어지고 있는 현실이다. 여기에는 재사용성과 이식성이라는 소프트웨어 개발 방법론의 장점이 적용되지 않는다는 문제점이 있다[2]. 이 문제를 해결하기 위해서는 시스템의 구조와 설계 방법을 효과적으로 관리하기 위한 각 프로젝트에 공통적으로 적용할 수

※ Corresponding Author : Eunjoo Lee, Address: (41566) E9-535, 80 Daehak-ro, Buk-gu, Daegu, Korea, TEL : +82-53-950-7548, FAX : +82-53-957-4846, E-mail : ej-lee@knu.ac.kr
Receipt date : Oct. 30, 2018, Revision date : Jan. 13, 2019
Approval date : Jan. 15, 2019

[†] School of Computer Science and Engineering Kyung-pook National University (E-mail : yujinkor@naver.com)
[††] Dept. of Computer Eng. Korea Army Academy at Yeong-Cheon (E-mail : dshan123123@naver.com)
[†††] School of Computer Science and Engineering Kyung-pook National University

있는 기본적인 설계 가이드가 요구된다. 이에 따라 본 논문에서는 MDA개발 방법론을 바탕으로 모바일 플랫폼의 비종속적인 부분에 대한 재사용성과 이식성 그리고 상호운용성에 초점을 두어 모바일 크로스 플랫폼의 메타모델을 정의하고자 한다.

본 논문의 구성은 2장에서 관련 연구로써 CWM에 대하여 정의하고, 3장에서 제안하는 모바일 크로스 프레임워크에 사용되는 메타모델을 설명한다. 여기에서는 제안한 메타모델의 전체적인 개요와 모바일 앱을 개발하는 프로세스의 각 단계에 따른 산출물을 서술한다. 4장에서는 앱 개발 과정에서 제안한 메타모델에 대한 기대효과를 논하고 5장에서 결론을 맺는다.

2. 관련연구

2.1 MDA(Model Driven Architecture)

그동안 사용된 여러 소프트웨어 개발 방법론 중에서 객체지향 개발방법론과 컴포넌트기반 개발 방법론에 많은 관심이 집중되었다. 그 중 컴포넌트 개발 방법론은 컴포넌트를 개발하고 개발된 컴포넌트들을 조립하는 형식으로 소프트웨어의 재사용성에 초점을 두어 개발되는 방식이다[3]. 최근 적용되는 소프트웨어 개발 방법은 MDA로 여기서는 모델지향 접근으로써 큰 서비스 구현을 위해 서비스를 정의하거나 객체지향 모델링 기술을 표현하기에 Fig. 1과 같이 CWM, MOF, UML, QVT와 같이 꼭 필요한 기술을 정의한다[4].

MDA방식은 플랫폼에 비종속적인 모델인 PIM (Platform Independent Model)과 특정 플랫폼에 종속적인 모델인 PSM(Platform Specification Model)

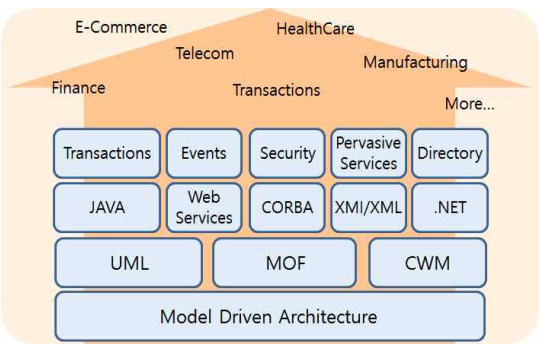

Fig. 1. MDA Architecture [5].

로 구분하여 플랫폼간의 이식성을 높이는데 목적을 둔다[5,8].

2.2 CWM(Common Warehouse Metamodel)

CWM은 사용자와 개발자의 의사결정에 필요한 데이터베이스 저장 데이터를 공통의 기술 방식으로 정의하고 변환하기 위한 데이터 웨어하우스의 메타데이터를 의미한다. 문법과 의미가 완성된 문서를 가져오고 내보내기 위한 메타모델로써 관계성과 객체지향성 그리고 계층적인 운영 데이터 모델을 나타낸다. CWM의 설계 기본은 OMG 메타모델을 확장해 놓은 구조로써 지능형 비즈니스 도메인과 데이터웨어하우징을 사용자의 요구사항과 목적에 맞도록 정의해 놓은 구조이다. CWM은 OMG 메타모델 구조의 영향으로 객체지향을 기본개념으로 모듈화 또는 패키지의 아키텍처를 갖기 때문에 CWM은 4개의 계층으로 OMG 메타모델의 기본구조를 갖는다. Fig. 2와 같이 OMG 메타모델의 M0계층은 컴포넌트의 저장 장소에 대한 정의를 나타낸다. M1계층은 XMI로 구성되며 커스터머 파일에 대한 정보를 저장한다. M2계층은 UML을 이용한 설계와 XMI에 대한 정보를 저장하며 M3계층은 MOF와 XMI내용을 저장한다[6].

2.3 메타모델

소프트웨어 개발과정에서 필요한 의미론적 접근을 이해하는데 필요한 메타모델은 소프트웨어와 SPEM2.0(System Process Engineering Meat-Model 2.0)의 종합적인 정의를 제공하는데 그 목적이 있다. 메타모델은 객체지향적 접근으로 UML 표기법을 사용하여 설계단계에 적용한다.

SPEM2.0은 UML2.0(Unified Modeling Language 2.0)프로파일과 마찬가지로 메타모델로 정의하며 모델을 설계함에 있어 준수해야 할 제약사항은 스테레오타입으로 적용하여 설계를 한다. SPEM2.0의 메타

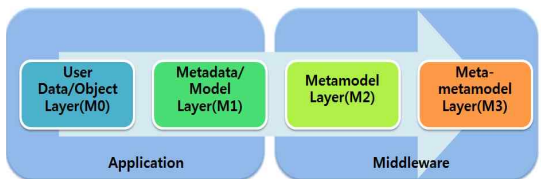

Fig. 2. OMG Metamodeling Architecture.

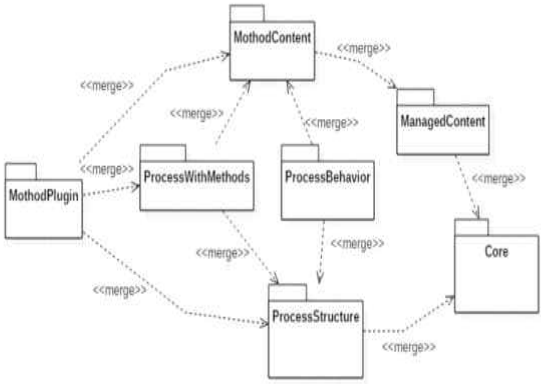

Fig. 3. Structure of SPEM2.0 MetaModel.

모델은 개발방법과 서로 다른 개발배경 그리고 개발 단계별 서로 다른 구문형식, 설계단계마다 발생하는 모델들을 수용하기 위함이다.

SPEM2.0 메타모델의 구조는 Fig. 3과 같이 7개의 주요 메타모델 패키지로 구성되어 있다. 이 구조는 논리적 구조를 바탕으로 구성되며 각 패키지는 추가적인 요소에 의해 생성된다. 각 패키지는 설계모델의 최하위 단계부터 정의되며 최상위 단계와는 상관없이 단계적으로 핵심패키지를 구성하여 합병하는 형식으로 주요 메타모델 패키지가 생성된다[7].

객체지향적 접근에 사용하는 UML2.0 프로파일은 각 해당 요소의 의미를 메타모델 MOF로 정의하였다. 메타모델은 해당 모델의 의미를 쉽게 이해하고 컴포넌트 기술의 핵심을 보다 정형화된 형태로 표현하기 위해 모델로 표현한다.

3. MCF를 위한 메타모델 설계

3.1 모바일 어플리케이션 설계 개념도

모바일 크로스 프레임워크는 핵심 자산을 재활용할 수 있도록 어플리케이션의 특정 기능을 모아 표준적인 부분을 설계하고 모바일 어플리케이션 개발단계 중 설계부터 구현단계까지 각 단계마다 발생할 수 있는 개발의 어려움을 MDA를 적용을 통하여 해결하고자 Fig. 4와 같이 제안하였다. 사용자가 요구하는 기능을 분석하고 그 기능에 대한 컴포넌트를 추출하여 CIM(Computation Independent Model)단계와 PIM단계의 결과물을 산출한다. 컴포넌트 기술 요소의 표준 메타모델을 정의하여 호환성과 재사용성을 보장하기 위해 CIM단계에서 비즈니스 요구사

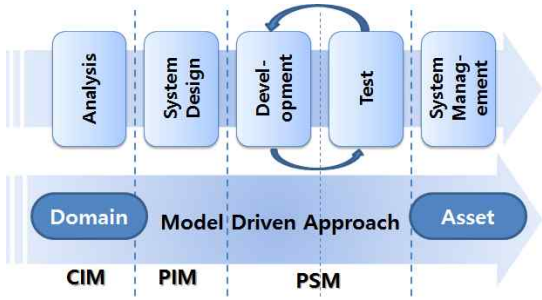

Fig. 4. Mobile Cross Framework Development Process.

항의 기능적 설계를 표현한다[8].

이를 위해서는 먼저 컴포넌트 개발 방법론의 과정을 토대로 도메인분석단계에서 필요한 기능을 추출하고 각 요소마다 구별할 수 있는 내용정리가 필요하다. 그 후 요구사항 분석단계에서 사용자의 요구사항이 반영되고 PIM과 PSM단계에서 아키텍처 설계가 시작된다. 아키텍처 설계 단계는 시스템 구현을 위해 시스템의 전반적인 구조를 정의하고 설계하며 유지보수에 대한 가이드라인을 제시하는 단계이다. Fig 5는 개발하고자 하는 시스템의 논리적 구조를 나타낸다. 모바일 크로스 프레임워크 아키텍처는 사용자의 요구사항을 나타내는 UI Layer와 앱 기능을 플랫폼에 독립적인 설계영역과 비독립적인 설계부분으로 분류하여 관리할 수 있는 메타데이터 설계가 포함되는 크로스 개발 영역으로 나뉜다. 마지막으로 개발 OS에 적합한 앱으로 구현될 수 있는 비즈니스 영역으로 나뉜다. 사용자의 요구사항에 따라 특정 언어를 사용하여 개발하는 모바일 앱이 있다고 가정하자. 모바일 앱 개발 프로세스에 따라 개발자는 서로 다른 플랫폼에 적용하도록 설계하고 구현한다면 개발초기 단계부터 서로 다른 두 가지로 개발해야 하는 문

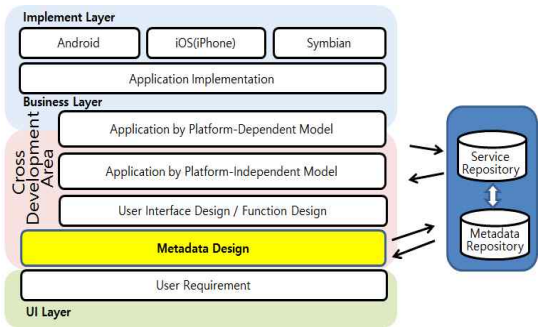

Fig. 5. Mobile Cross Framework Architecture.

제점이 있다. 이 문제를 해결하기 위해 개발 초기 단계에서 사용자가 필요로 하는 기능별 명세화작업이 필요하다. 명세화작업은 MDA 개발 단계에서 메타모델을 이용하여 나타낸다[8-10].

본 논문에서는 사용자의 요구사항을 파악하고 도메인 분석단계에서 사용될 메타데이타 모델을 추출하여 도메인 분석 단계에서부터 사용자의 요구사항을 반영하여 개발 초기 단계인 PIM단계는 좀 더 효율적으로 개발이 이루어질 것으로 기대한다.

3.2 모바일 어플리케이션 메타데이터

본 논문에서는 6가지의 메타모델 패키지를 구조화한 MCF meta model을 제안한다. 이는 모바일 어플리케이션을 개발할 때, PIM단계에서 새롭게 개발하려는 모바일 앱의 기능이나 특징을 식별할 수 있도록 정의하는데 목적을 둔다.

Table 1에서는 다음과 같이 사항을 나타낸다. 앱에 대한 고유한 식별 번호를 부여하고, 그 앱의 기능을 쉽게 알아볼 수 있도록 이름을 부여하고 특정 앱의 종류, 제작자 그리고 인터페이스에 대한 정보를 General info에 저장한다. 그리고 앱이 가지고 있는 가장 큰 특징과 가장 적합한 사용자를 구분할 수 있도록 Semantics에 저장한다. 앱이 개발되는 환경에 대한 정보 즉, 앱버전, 앱이 개발되는 단계중 어느 단계까지 개발되어 있는지, 앱을 만든 제작자의 소속이나 제작날짜와 같은 정보를 저장한다. Meta-Meta-data는 앱의 개발되는 중 최종 수정일을 기록하고 이러한 개발단계에서 필요로하는 정보를 메타데이터로 관리되도록 자료로 보관하게 된다. 그리고 재사용성을 고려하여 Technical은 앱의 유형, 개발된 앱

Table 1. MCF Application Meta-Model List

Category	Details	Explain
General Info	Identifier	ID number of App.
	Title	Name of App.
	CatalogEntry	Symbols to identify the type of App.
	Publisher	App Director
	Date	Development date
	Language	Language for the interface
	Description	Explanation of App.
	Keywords	Core word explaining App.
Semantics	Main Concept	Theme of App.
	End User Type	The fittest and final user
Lifecycle	Version	Version of App.
	Status	Review Level or Status of App.
	Contribute	Developer of App.
	Date	Developing date of App.
MetaMetadata	Identifier	ID number of meta-data
	CatalogEntry	ID symbol of meta-data
	Last Modified Data	Final correction data of App.
	Metadata Scheme	Schema name of meta-data
Technical	Format	Type of App.
	Location	Location of App.
	Requirements	Technical performance for used App.
	Installation Remarks	Explanation about installation of App.
Classification	Purpose	Purpose of App. classification
	Path	Path of App. classification
	Id	ID symbol for App. classification
	Description	Explanation of App.
	Keywords	Keyword of App.

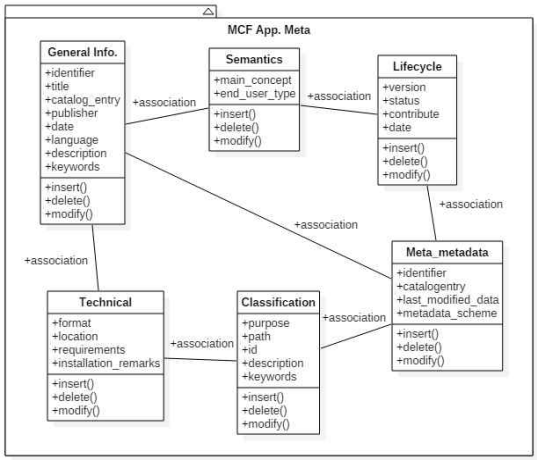

Fig. 6. MCF Application Meta-model.

이 저장된 위치, 앱을 사용하기 위한 기술적 성능과 앱 설치 또는 사용 설명에 대한 전반적인 기술정보를 저장한다. Classification은 앱을 분류할 목적으로 사용되는 정보를 저장하게 된다. 앱 분류에 대한 목적, 경로, 식별기호, 앱에 대한 설명, 앱을 찾기 위한 키워드 등을 저장한다.

이와 같이 메타데이타의 요소들을 6가지의 카테고리로 나누어 구조화하면 다음 Fig. 6과 같다. Fig. 6에서 정의한 메타데이터 모델은 비기능적 요구사항을 추출하여 개발의 편리성과 신뢰성, 효율성, 호환성에 목적을 두고 설계될 어플리케이션의 품질을 향상시키기 위한 요소들을 나타낸다.

위 Fig. 6에서 나타내는 메타모델은 사용자가 필요로 하는 앱을 개발할 때, 이미 설계단계에서 PIM 단계까지만 개발되어진 앱을 재사용할 수 있다면 개발자의 입장에서는 좀 더 개발 일정을 단축할 수 있으며 PSM단계에 대한 인터페이스만 접목된다면 보다 손쉽게 앱을 개발할 수 있다는 편리성을 갖을 수 있다.

따라서, 도메인 분석단계에서 유사한 모바일 어플리케이션 존재 여부를 확인할 수 있고, 설계에 대한 기초 데이터로써 생명주기를 파악할 수 있으며, 메타모델을 저장한 리파지토리에서 메타데이터 항목을 이용하여 유사도 분석을 할 수 있다. 이처럼 MCF 어플리케이션 메타모델은 6가지의 메타모델 패키지에서 구조화된 MCF 메타모델을 제시하여 소프트웨어나 시스템 개발시 컴포넌트를 정의하고 하는데 사용된다. 정의된 메타모델은 효율성과 추가적인 구조를 제공함으로써 소프트웨어 공학적 측면에서 접근해볼 때 관리자 측면에서 개발 및 관리를 보다 쉽게 소프트웨어 품질을 가늠할 수 있다. 이러한 소프트웨어공학적으로 접근하는 품질 척도는 '얼마나 사용자의 요구와 부합되는가?'에 중점을 두어 개발된 어플리케이션의 품질을 측정할 수 있는 요건이다. Table 2에서는 소프트웨어공학적 접근의 품질관리에서 정의한 항목을 바탕으로 첫째, 최소한의 실행시간과 기억장소를 소요하여 요구된 기능을 수행하는 앱의 효율성 둘째, 새로운 요구사항이 발생한다면 쉽게 수정될 수 있는 시스템의 유연성 셋째, 다른 시스템과 정보를 교환할 수 있는 상호운용성 넷째, 여러 하드웨어 환경에서 운용되기 위해 쉽게 수정될 수 있는 이식성 다섯째, 시스템 소프트웨어나 데이터의 독단적인 액세스를 제어할 수 있는 무결성 여섯째, 정확하고 일관된 결과로 요구된 기능을 수행하는 시스템의 신뢰성 일곱 번째, 사용자의 요구사항을 충족할 수 있는 정확도 마지막으로 시스템의 일부나 전체를 여러 응용부분에서 사용할 수 있는 재사용성을 품질 측정 척도를 나타낸다[11].

Table 2. Software Quality Metrics based on MCF

Efficiency	Performance ability of system that performs functions using minimum computing and memory
Flexivility	System ability that easily be corrected adjacent to new requiement
Integrity	System ability that can control system software or independent access of data
Interoperability	System ability that can exchange information with other system
Portability	System ability that can be easily corrected to be used to one or more hardware environment
Reliability	System ability that performs required functions identical to result
Correctness	Certain degree that salsifies user's required function
Reusability	Ability of system being used in many applications

4. 기대효과

본 논문에서 제안하는 모바일 크로스 프레임워크는 개발하려는 앱의 특정 기능만을 쉽게 파악할 수 있도록 간단한 메타모델로 설계하였다. 이는 모바일 어플리케이션 개발단계 중 설계부터 구현까지 소프트웨어 개발 단계마다 발생 할 수 있는 중복되는 부분이나 통합의 어려움을 메타데이터 명세 목록을 이용하여 사용자의 요구사항을 쉽게 접근할 수 있다. 새롭게 추가되는 요구사항도 기존의 메타데이터의 명세 목록과 비교하여 쉽게 수정할 수 있는 장점을 갖는다. 이는 통합성, 융통성, 효율성을 높여줄 것이다.

또한 설계단계에서는 어플리케이션에 개발에 사용되는 개발언어에 구애받지 않는 수행 기능을 명세하고, 개발 및 구현단계에서 적용되는 개발언어에 맞게 다른 플랫폼으로 앱을 개발할 수 있는 장점을 가짐으로써 이식성, 상호운용성을 보장한다.

모바일 앱 개발할 때, 이중으로 하나의 기능을 수행하는 소프트웨어를 서로 다른 프레임워크를 고민하며 설계 단계에서부터 구현단계까지 똑같은 소프트웨어를 동일하게 구현하게 된다는 문제점을 개선하고자 본 논문에서 제안하는 개발방법은 보다 쉽고 신속하게 새로운 소프트웨어를 개발할 수 있는 방안이 될 것이다.

즉, 중복되는 설계 및 구현 과정에서 메타데이터를 이용하여 효율적이고 신속하게 필요한 정보를 찾을 수 있고 재사용할 수 있기 때문에 생산성이 높을 것으로 기대한다.

5. 결 론

전통적인 소프트웨어 개발 방법은 구조적 개발 방법론으로부터 객체지향 개발 방법론과 컴포넌트 기반 개발 방법론에 이르기까지 활발하게 연구되고 있다. 현재도 소프트웨어 개발 방법론은 도메인에 따라 다양한 방법으로 연구되고 개발되고 있다.

최근 모바일 환경에서의 소프트웨어 개발 방법론은 기존의 컴포넌트 기반 개발 방법론을 응용하여 새롭게 연구되는 추세이다. 그러나 모바일 플랫폼의 종류가 크게 안드로이드와 아이폰 운영체제로 양분돼있어 개발하고자 하는 앱은 디바이스에 따라 종속적으로 개발이 이루어지고 있는 현실이다. 이는 재사용성과 이식성이라는 소프트웨어 개발 방법론의 장점이 적용되지 않는다는 문제점이 있다. 이 문제를 해결하기 위해서는 시스템의 구조와 설계 방법을 효과적으로 관리하기 위한 각 프로젝트에 공통적으로 적용할 수 있는 기본적인 설계 가이드가 요구됨에 따라 본 논문에서는 MDA개발 방법론을 바탕으로 모바일 플랫폼의 비종속적인 부분에 대한 앱의 특정 기능만을 쉽게 파악할 수 있는 장점과 설계단계에서 산출된 결과물을 재사용할 수 있고 이종간의 시스템과의 이식성 및 상호운용성에 초점을 두어 6가지의 메타모델 패키지로 구조화한 모바일 크로스 플랫폼의 메타모델을 제안하였다.

제안된 메타모델을 적용하여 관련 정보를 관리하고 플랫폼에 독립적인 모바일 앱 개발 프로세스에 따라 구현된다면 향후 정형화된 개발방법론 정립에 도움이 될 것이다.

향후, 소프트웨어 개발 단계에서 생성될 수 있는 결과물을 토대로 유사도를 분석하고 그 분석 결과에 따라 설계 초기단계에서부터 수정하거나 재사용할 수 있는지 여부를 판단하기 위한 유사한 앱을 찾을 수 있는 방법을 연구하고자 한다.

REFERENCE

[1] E.S. Cho, “Design of Methodology Framework Based on Meta-Model,” *Journal of the Korea Academia-Industrial Cooperation Society*, Vol. 16, No. 10, pp. 6969-6976, 2015.

[2] S. Sultana, F. Arif, and I. avaid, “The Proposed Blended-MDA for Software Modeling in Architecture Phase,” *American Journal of Engineering Research*, Vol. 5, No. 7, pp. 275-279, 2016.

[3] S.H. Lee, E.S. Cho, and C.J. Kim, “A Design and Adaptatijon Technique of UML-based Layered Meta-Model for Component Development,” *Journal of the Korea Society for Simulation*, Vol. 15, No. 2, pp. 59-69, 2006.

[4] M. Belaunde and P. Falcarin, “Realizing an MDA and SOA Marriage for the Development of Mobile Services,” *Proceedings of 4th European Conference on Modelling Driven Architecture-Foundations and Applications,* pp. 393-405, 2008.

[5] W.S. Kim, O.C. Kwon, and K.S. Shin, “An Introduction to the Model Driven Architecture,” *Electronics and Telecommunications Research Institute Electronics and Telecommunications Trends*, Vol. 17, No. 6, pp. 11-19, 2002.

[6] P. John. *Common Warehouse Metamodel*, Wiley, New York, 2002.

[7] OMG, Software and Systems Process Engineering Metamodel Specification(SPEM) Version 2.0, https://www.omg.org/spec/SPEM/2.0/PDF, April 2008(Feb. 2019)

[8] Y.J. Song, E.J. Lee, and D.S. Han, “Design Mobile Cross Framework Based MDA,” *Journal of Korea Multimedia Society*, Vol. 19, No. 8, pp. 1445-1452, 2016.

[9] B. Gallina, M.A. Javed, F.U. Muram, and S. Punnekkat, “A Model-driven Dependability Analysis Method for Component-based Architectures,” *Proceeding of 2012 38th Euromicro Conference on Software Engineering and Advanced Applications*, pp. 233-240, 2012.

[10] E. Breton and J. Bezivin, “Model-Driven Process Engineering,” *Proceeding of Computer Society Signature Conference on Computers, Software and Applications 25th Annual International*, pp. 225-230, 2001.

[11] H.K. Kim, *Software Engineering Based on Object and Component*, Greenpress, 2010. (Seoul Korea)

송 유 진

2003년 대구가톨릭대학교 컴퓨터정보통신공학부(학사)
2006년 대구가톨릭대학교 전자계산교육전공(석사)
2008년 3월~현재 경북대학교 IT대학 컴퓨터학부 (박사수료)
관심분야 : Mobile-based software development, Model Driven Architecture, Component Based Design

한 덕 수

1988년 금오공과대학교 전자계산기공학과(학사)
1998년 대구가톨릭대학교 전산통계학과(석사)
2003년 연세대학교 컴퓨터과학과(박사)
1992년 6월~현재 육군3사관학교 컴퓨터공학과
관심분야 : Multimedia, Virtual Reality, HCI, e-Learning, Software Engineering

이 은 주

1997년 서울대학교 계산통계학과(학사)
1999년 서울대학교 전산과학과(석사)
2005년 서울대학교 전기컴퓨터공학부 (박사)
2005년 3월~2005년 10월 서울대학교 공과대학 BK 박사후 연구원
2005년 11월~2006년 2월 삼성종합기술원 전문연구원
2006년 3월~현 재 경북대학교 IT대학 컴퓨터학부(전임강사, 조교수, 현재 부교수)
관심분야 : Mining software repository, Change pattern, software metric, software evolution