한국시뮬레이션학회 논문지
Vol. 15, No. 2, pp. 59-69 (2006. 6)

컴포넌트 개발을 위한 UML 기반의 계층형 메타 모델 설계 및 적용기법

이숙희[1†] · 조은숙[2] · 김철진[3]

A Design and Adaptation Technique of UML-based Layered Meta-Model for Component Development

Sook Hee Lee · Eun Sook Cho · Chul Jin Kim

ABSTRACT

Component-based software development is introduced as a new development paradigm in software development method. This approach is different from existing software development approach because it is based on reusable and autonomous unit, component. Therefore, component-based development(CBD)is divided into two stages; component development process and component assembly process; application development process. Component development process is the core of CBD because component has a key for good software. Currently many methodologies or tools have been introduced by various academies or industries. However, those don't suggest systematic and flexible modeling techniques adaptable easily into component development project. Existing approaches have a unique orarbitrary modeling technique or provide heuristic guidelines for component modeling. As a result, many component developers are faced with a difficult problems; how to developcomponent models, when develop which diagrams, and so on. In order to address this problem, we suggest a meta-model driven approach for component development in this paper. We provide meta- models according to both layer and development phase. We expect that suggested meta-models allow component developers to develop appropriate models of the time.

Key words : Component, Connector, Meta-Model, MDA

요 약

새로운 개발 패러다임으로 컴포넌트 기반 소프트웨어 개발이 소개되었다. 이 접근방식은 재사용 가능하면서 독립적인 단위의 컴포넌트들을 기반으로 소프트웨어를 개발하기 때문에 기존의 소프트웨어 개발 방식과는 차이가 있다. 따라서, 컴포넌트 기반 개발(CBD)은 크게 두 단계인 컴포넌트 개발 프로세스와 컴포넌트 조립 프로세스 즉, 어플리케이션 개발 프로세스로 구분된다. 컴포넌트 개발 프로세스는 컴포넌트가 품질 좋은 소프트웨어를 위한 열쇠이기 때문에 컴포넌트 기반 소프트웨어 개발에서 핵심적인 부분이라고 할 수 있다. 현재 이와 관련하여 학계 및 산업체에서 많은 방법론이나 도구들이 소개되고 있다. 그러나, 이러한 방법론이나 도구들은 실제 컴포넌트 개발 프로젝트에 손쉽게 적용할 수 있을 정도의 체계적이면서 유연한 모델링 기법들을 제시하고 있지 못하고 있다. 또한 현존 기법들은 컴포넌트 모델링에 있어서 독단적인 기법을 갖고 있거나 혹은 경험적 지침 정도만을 제공하고 있는 수준이다. 그 결과 많은 컴포넌트 개발자들이 컴포넌트 모델을 어떻게 개발해야 할지, 언제 어떠한 다이어그램을 개발해야 할지 등에 대한 어려움을 직면하고 있다. 본 논문에서는 이러한 문제를 해결하기 위해 메타 모델 기반의 접근법을 제시하고자 한다. 특히 계층과 개발 단계에 따른 메타 모델들을 제시하고자 한다. 이로써 개발자들은 적절한 시기에 적합한 모델들을 개발할 수 있게 된다.

주요어 : 컴포넌트, 프레임워크, 프레임워크 참조 모델, 가변성

2006년 6월 3일 접수, 2006년 6월 12일 채택

[1] 서경대학교 인터넷정보학과
[2] 서일대학 소프트웨어학과
[3] 삼성전자 디지털솔루션센터
주 저 자 : 이숙희
교신저자 : 조은숙
E-mail; escho@seoil.ac.kr

1. 서 론

소프트웨어 재사용을 위한 새로운 기술로 컴포넌트 기술이 소개되면서, 소프트웨어 개발 기법에 있어서도 컴포넌트 기반 소프트웨어 개발 기법에 대한 연구가 활발해졌다. 기존 소프트웨어 개발 방식과 달리 컴포넌트 기반 소프트웨어 개발 방법은 컴포넌트 개발 과정과 개발된 컴포넌트들을 조립하여 애플리케이션을 개발하는 컴포넌트 기반 애플리케이션 개발 과정으로 나누어진다[1-3]. 이 두 과정에서 컴포넌트 개발 과정은 소프트웨어 개발을 위한 사전 과정으로서, 컴포넌트 기반 개발 과정에서 매우 핵심적인 과정이다. 만일, 컴포넌트 개발 과정에서 재사용성이 높은 고품질의 컴포넌트가 개발되지 않으면 컴포넌트를 기반으로 한 소프트웨어 개발은 고품질의 소프트웨어를 개발할 수 없게 된다. 따라서, 컴포넌트 기반 개발 기법에서 컴포넌트 개발과 관련된 다양한 기법들이 소개되고 있다. 그러나 현재까지의 대부분의 컴포넌트 개발 기법들을 보면 개발자들의 경험에 기반을 둔 경험적 방식을 채택하고 있거나, 아니면 너무 이론적이어서 실제 개발자들에게 있어서 실무 적용에 잘 적응이 안되고 있는 실정이다. 더욱이 컴포넌트 개발과 관련하여 여러 개발 방법론들이나 설계 기법들이 소개되고 있다. 그러나 각기 방법론마다 의미가 중복된 서로 다른 모델들을 제시하고 있을 뿐만 아니라 모델들 간의 연관성이 제대로 설정되어 있지 않고 제시되어 있다. 이로 인해 개발자들은 단계별로 필요한 모델들이나 혹은 개발한 모델들 간의 일관성을 제대로 파악하기가 매우 어려운 상황이다.

본 논문에서는 이러한 문제점을 해결해 보고자 소프트웨어 개발 방식에서 최근에 새롭게 도입하고 있는 MDA[4,5]와 UML[6,7]과 같은 모델 주도형 개발 방식을 적용하고자 한다. 즉, 컴포넌트 개발에 필요한 UML 모델들에 대해 단계별 및 계층별로 메타 모델을 제시함으로써 컴포넌트 개발 시 단계별로 필요한 모델들을 효율적으로 개발할 수 있을 뿐만 아니라 개발된 모델들 간의 일관성이나 적합성을 쉽게 확인할 수 있다.

이 논문의 나머지는 다음과 같이 구성된다. 2장에서는 컴포넌트 개발 기법이나 메타 모델에 관련된 기존 연구들에 대해 살펴본다. 3장에서는 컴포넌트 개발 단계 별 계층형 메타 모델들을 제시한다. 이 논문에서는 COMO[8]에서 제시한 3단계를 기반으로 단계를 분류하여 제시하고 있으며, 또한 제시하고 있는 모델 즉, 다이어그램들은 UML의 다이어그램들을 기반으로 하고 있다. 4장에서는 제시한 기법이 실제 컴포넌트 개발에 어떻게 적용되는 지에 대한 사례 연구와 적용 결과 도출된 평가 결과들을 제시한다. 마지막으로 5장에서는 결론 및 향후 연구 과제를 제시한다.

2. 관련 연구

이 장에서는 본 논문에서 제시하는 메타 모델과 관련된 기존의 여러 연구들과 컴포넌트 개발 방법론들을 제시하고, 기존 연구들이 지니고 있는 한계점들을 제시한다.

2.1. 메타 모델 관련 연구

2.1.1. MDA

컴포넌트를 조합하는 방식으로 컴포넌트 기반 개발(CBD) 방법론이 널리 부각되면서 플랫폼에 독립적인 표준화 기술인 MDA(Model Driven Architecture)가 새롭게 대두되고 있다[4,5]. MDA는 설계 환경이나 시스템에 종속적이지 않고 소프트웨어를 개발하기 위한 기술이라고 할 수 있다. CBD가 J2EE[9,10]나 .NET, CORBA 등 미들웨어 프레임워크에 독립적이지 못한 문제점들을 극복하기 위한 것이 MDA이다.

MOF(Meta Object Facility)라는 메타 모델 정의 언어를 통해 플랫폼에 종속되지 않는 비즈니스 모델을 설계한 뒤 필요에 따라 프레임워크에 매핑해 어플리케이션을 개발한다. 정의된 모델을 기반으로 다양한 플랫폼의 어플리케이션을 생산해 낼 수 있으므로 개발 생산성을 높일 수 있다. MDA에서는 시스템의 아키텍쳐, 비즈니스 모델의 정적, 동적 모델을 모델링 하는데 UML[6,7]을 이용하도록 하고 있다. 이외에도 MDA의 핵심 기술로는 XML기반 데이터관리를 위한 표준인XMI(XML Metadata Interchange)와 비즈니스 메타데이터와 기술적 메타데이터를 표현하기 위한 표준인CWM(Common Warehouse Metamodel)이 있다.

OMG는 integration문제를 해결하기 위한 아키텍쳐인 OMA(Object Management Architecture)를 제시해 왔으나, 처음 설계된 이후로 많은 변화가 이루어 지지 않았다. 이러한 상황 속에서 MDA는 시스템의 life cycle (Business Modeling, Design, Assembly, Integration, Deployment, Management) 동안에 통합과 상호 연동을 제공하기 위한 OMA의 진화된 형태라고 할 수 있다. MDA는 통합, 상호 연동성, 이동성을 위해 관련된 부분들을 효과적으로 분리하는 모델을 만들기 위한 아키텍쳐를

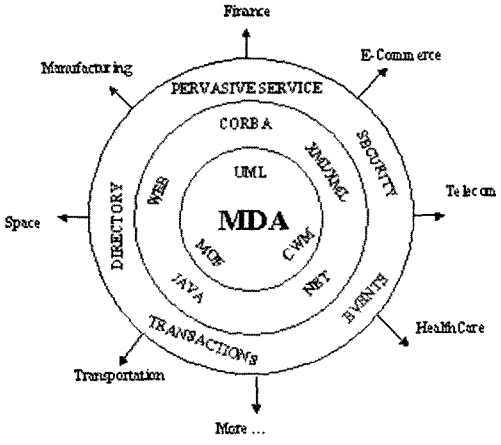


그림 1. 모델 주도형 아키텍처

정의한다.

그림 2는 MDA 스펙에서 제공하는 MDA의 기본 아키텍처를 나타내고 있다. 이 그림을 통해서 MDA의 철학을 이해할 수 있는데, 제일 중앙에 있는 원을 기준으로 밖으로 퍼져 나갈수록 특정 기술이나 비즈니스 도메인에 종속적으로 변해가고 있다는 것을 볼 수 있을 것이다. 제일 중앙 원에 있는 기술들은 특정 도메인이나 기술에 종속되지 않는 OMG의 모델링 기술들이며, 그 다음 원은 목표 플랫폼을 나타내고 있다. 가장 밖의 원은 특정 서비스들이 플랫폼에 종속되지 않도록 모델링 수준에서 이용될 수 있도록 OMG에서 정의되고 있는 서비스들이다. 개발되는 비즈니스 시스템은 다양한 플랫폼에서 운영될 수 있어야 한다. 그러기 위해서는 설계된 모델이 특정 플랫폼에 종속적이지 않아야 하며 비즈니스 모델에 플랫폼의 종속적인 정보들이 포함되지 않아야 한다. MDA에서는 플랫폼에 독립적인 모델을 PIM(Platform Independent Model)이라고 하며 일반적으로 UML을 사용해 모델링 한다. 그리고 특정 플랫폼에 종속적인 모델을 PSM(Platform Specific Model)이라고 한다.

2.1.2. UML-F

UML-F는 프레임워크와 프러덕트 라인(Product-Line) 아키텍처 설계를 위해 D'Souza에 의해 개발된 것으로서, 기존 UML 확장 메커니즘을 이용해서 추가 모델링 요소들을 표현하고 있다[11,12]. 특히 프레임워크 설계에 있어서 협력도(Collaboration Diagram)을 이용한 패턴이나 혹은 프레임워크의 가변 포인트(Variation Point) 등과 같은 메커니즘들을 UML-F에서 추가로 정의하여 제시하고 있다. 또한 프레임워크를 UML 1.3의 서브시스템 개념을 확장하여 표현하고 있다.

이처럼 UML-F는 프레임워크나 제품 라인 아키텍처 설계에 필요한 여러 가지 추가 모델링 요소들을 확장하여 제시하고 있다는 점에서 큰 의의가 있다고 말할 수 있다. 그러나, 컴포넌트 개발 프로세스와 같은 방법론 상에서의 모델들을 적용하는 차원에서의 메타 모델 접근은 제시하지 못하고 있는 실정이다.

2.2. 컴포넌트 개발 방법론

2.2.1. CBD96

스털링 소프트웨어(Sterling Software) 사에서 개발한 컴포넌트 모델링 기법으로서 이 기법은 COOL:Spex이라는 모델링 도구를 이용해서 모델링이 가능하다[13,14]. 이 기법은 Catalysis 방법론을 기반으로 한 기법으로서 컴포넌트 모델링을 위한 주요 단계를 요구사항 정의, 분석, 행위 명세, 아키텍쳐 모델링으로 구분하고 있다. CBD의 한계점은 컴포넌트를 모델링하기 위한 업무들이 정의되어 있으나 업무에 대한 지침이 구체적으로 명시되어 있지 않으며 특히 도메인 모델링과 use case 모델링에서 컴포넌트를 추출하기 위한 기법이 제시되어 있지 않다. 그리고 행위 명세 단계의 인터페이스 모델링 업무에서 인터페이스를 어떻게 정의할 것인지에 대한 지침이 빈약하다.

2.2.2. SCIPIO

새로운 시스템은 정보 기술 전략과 정보 기술 요구 사항들을 컴포넌트 기반의 소프트웨어 개발로 자연스럽게 모델링 할 수 있는 개발 프로세스와 개발 기법을 요구하고 있다[15]. 이러한 요구사항을 만족시키기 위해 SCIPIO 컨소시엄에서 SCIPIO 방법론을 개발하였다. SCIPIO 방법론은 비즈니스 요구사항을 컴포넌트 기반의 소프트웨어 개발로 연결하는 방법론이다. SCIPIO의 한계점은 업무들간의 입.출력물이 어떻게 반영되는지 구체적으로 명시되어 있지 않다. 그리고 컴포넌트 개발과 관련된 구체적인 모델링 지침이 충분히 기술되어 있지 못함으로써 컴포넌트 추출이나 설계에 대한 지침이 부족하다.

2.2.3. Catalysis

Catalysis는 UML을 기반으로 컴포넌트 기반의 소프트웨어 개발을 지원하는 방법론이다[16]. 이 방법론은Fusion과 Syntropy와 같은 2세대 방법론들을 확장시켰는데 프

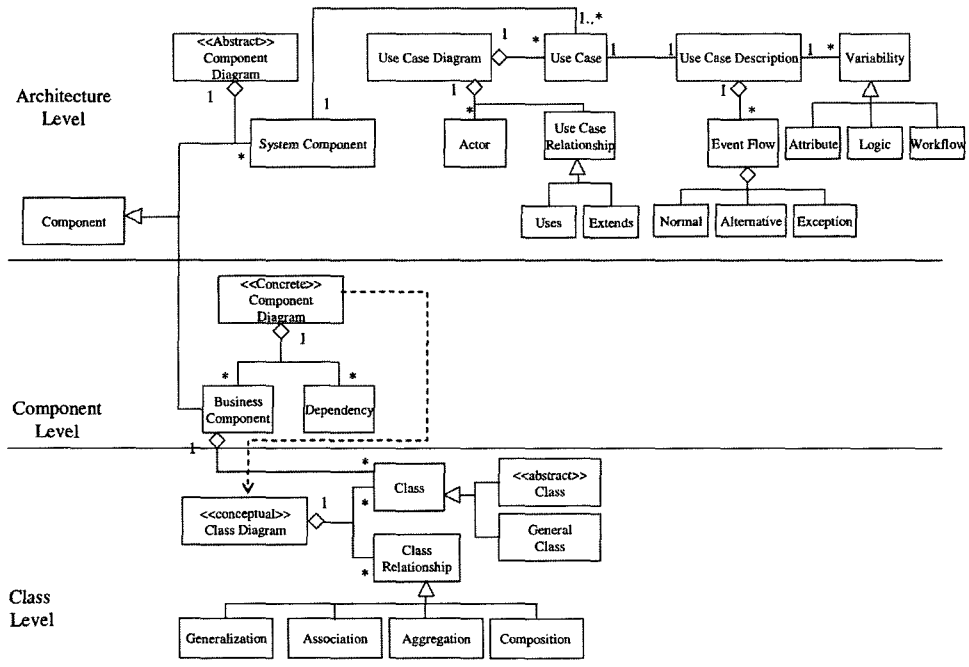


그림 2. 도메인 분석 단계 메타 모델

레임워크 기반의 개발 형태를 지원하고 분석에서부터 구현까지 이르는 모델링 기법들을 정의하며 컴포넌트들을 명세화하는 기법 등을 제시하고 있다. Catalysis 방법론에서는 소프트웨어 개발 프로세스를 여러 다양한 프로젝트에 적용하기 위해 단일 프로세스보다는 프로세스 패턴(Process Pattern)을 제공하고 있다. Catalysis의 한계점으로는 우선 업무들 가운데 컴포넌트를 추출하는 방법이 구체적으로 제공되어 있지 않다. 둘째, 컴포넌트 타입 모델에서 클래스들과 오퍼레이션들을 분리시킴으로 인해서 컴포넌트 인터페이스와 클래스의 오퍼레이션의 구분이 불분명하다.

3. 계층형 메타 모델

본 논문에서 컴포넌트 개발 단계에서 적용되는 UML 다이어그램들의 요소들에 대해서 각 단계별로 필요한 요소들을 분류하여 메타 모델로 정의하고자 한다. 기존 UML 메타 모델들에서는 각 다이어그램들에 대한 메타 모델 표현만 존재하는 반면에, 이 장에서는 단계별로 필요한 다이어그램과 단계별로 필요한 다이어그램의 모델링 요소들을 분류하여 표현한다. 이렇게 함으로써, 각 단계별로 설계되어야 할 모델링 요소들이 보다 정확하게 기술됨으로써 단계 내 혹은 단계 간 모델들 간의 일관성이 보장되게 된다. 또한 MDA 접근 방법처럼 컴포넌트 개략 설계 단계의 메타모델은 PIM (Platform Independent Model)을 적용하고, 컴포넌트 상세 설계 단계의 메타 모델은 PSM(Platform Specific Model)으로 구분하여 제시한다.

3.1. 도메인 분석 단계 메타 모델

도메인 분석 단계의 목표는 여러 유사한 어플리케이션들이 속하는 문제 도메인을 정확히 파악해서 컴포넌트에 관한 요구사항들을 추출하는 것이다. 이 단계의 가장 핵심적인 부분은 컴포넌트 추출이라 할 수 있다. 특히 비즈니스 컴포넌트는 그 도메인에서 여러 어플리케이션들이 공통적으로 사용하는 기능들을 재사용할 수 있는 단위로 구성되어야 하기 때문에 도메인 분석이 매우 중요한 부분을 차지한다. 이러한 업무들이 수행될 때 개발되는 모델들에 대한 메타 모델이 그림 2에 제시되고 있다.

이 단계의 메타 모델은 3계층인 아키텍처 레벨, 컴포넌트 레벨, 그리고 클래스 레벨로 구성되어 있다. 아키텍처 레벨은 도메인에 대한 아키텍처를 표현하는 계층으로서, 이 때 적용되는 다이어그램은 컴포넌트 다이어그램과 유

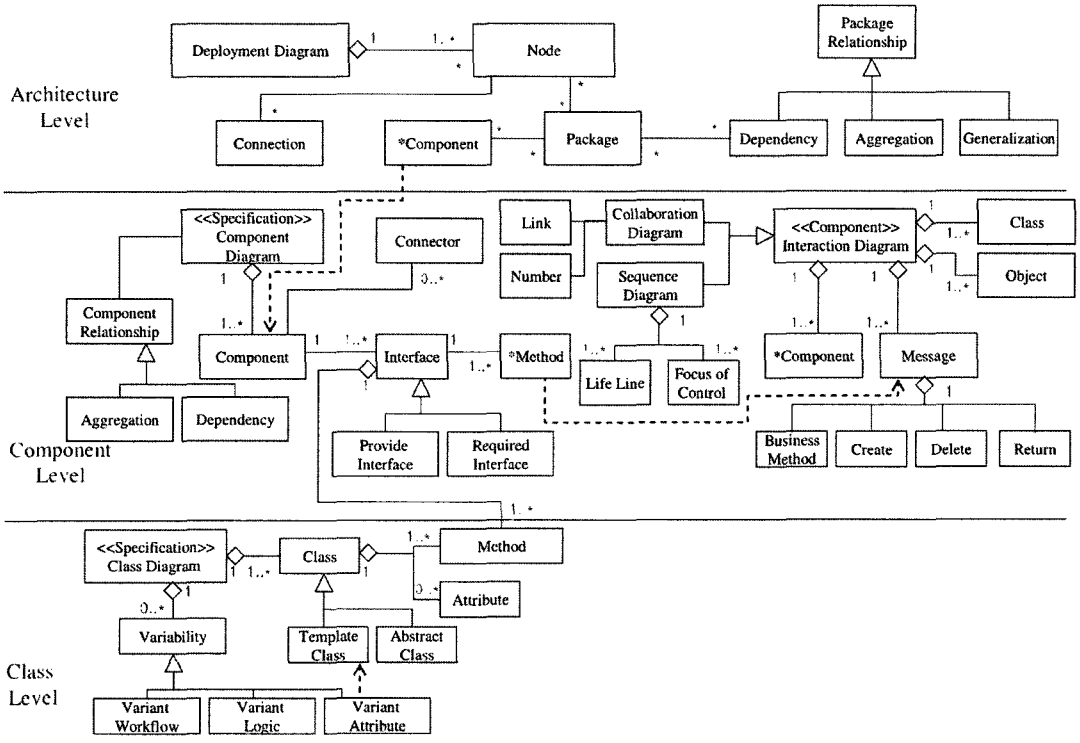


그림 3. 개략 설계 단계 메타 모델

스 케이스 다이어그램이다. 그러나 이 단계에서의 컴포넌트 다이어그램은 실제 구현 단위의 컴포넌트가 아닌 업무 단위의 컴포넌트 즉, 시스템 컴포넌트 단위로 표현한 컴포넌트 다이어그램이다. 따라서, 컴포넌트 다이어그램은 여러 개의 시스템 컴포넌트들로 구성되며, 하나의 시스템 컴포넌트는 하나 이상의 유스 케이스들의 집합으로 표현된다. 이 단계에서의 컴포넌트 계층은 비즈니스 컴포넌트들과 이들 간의 관계 정도를 표현하는 컴포넌트 다이어그램으로 표현되며, 클래스 계층은 비즈니스 객체들의 집합으로 구성된 클래스 다이어그램으로 표현된다.

3.2. 컴포넌트 개략 설계 단계 메타 모델

이 단계는 특정 컴포넌트 플랫폼에 종속되지 않은 컴포넌트 설계 모델을 개발하는 단계이다. 이 단계에서는 컴포넌트의 인터페이스 설계, 컴포넌트 내부 작업 흐름(Workflow) 설계, 컴포넌트 명세 설계 등이 주요 핵심 업무로 수행된다. 따라서 이러한 업무들의 결과물로 산출되는 모델들에 대한 메타 모델이 그림 3와 같이 설계되어야 한다.

개략 설계 단계의 메타 모델은 도메인 분석 단계 모델을 보다 구체화 시킨 모델로서, 이전 단계와 동일하게 계층은 3계층으로 표현된다. 그러나 이전 단계와 달리 이 단계에서는 아키텍처 계층을 표현하는 다이어그램으로 배치도(Deployment Diagram)과 패키지다이어그램이 적용된다. 또한 컴포넌트 계층을 표현하는 다이어그램으로는 컴포넌트 다이어그램과 상호 작용도(시퀀스 다이어그램 혹은 협력도)가 적용된다. 이 단계에서의 컴포넌트 다이어그램을 명세 수준의 컴포넌트 다이어그램이기 때문에 스테레오 타입으로 <<specification>>으로 표현하였다. 컴포넌트 레벨의 상호 작용도는 기존 UML의 객체들 간의 상호 작용도가 아닌 컴포넌트들 간의 컴포넌트 단위로 확장시켜서 표현했기 때문에 스테레오 타입으로 <<component>>로 표현하였다. 이 단계에서는 클래스에 대하여 가변성 설계가 이루어지기 때문에 클래스 다이어그램에 가변성(Variability)에 대한 항목이 추가되며, 가변성에서 속성 가변성은 템플릿 클래스로 설계되어야 하기 때문에 클래스에 대한 분류도 표현되고 있다. 참고로, 그림에서 박스 안의 *의 의미는 동일한 클래스

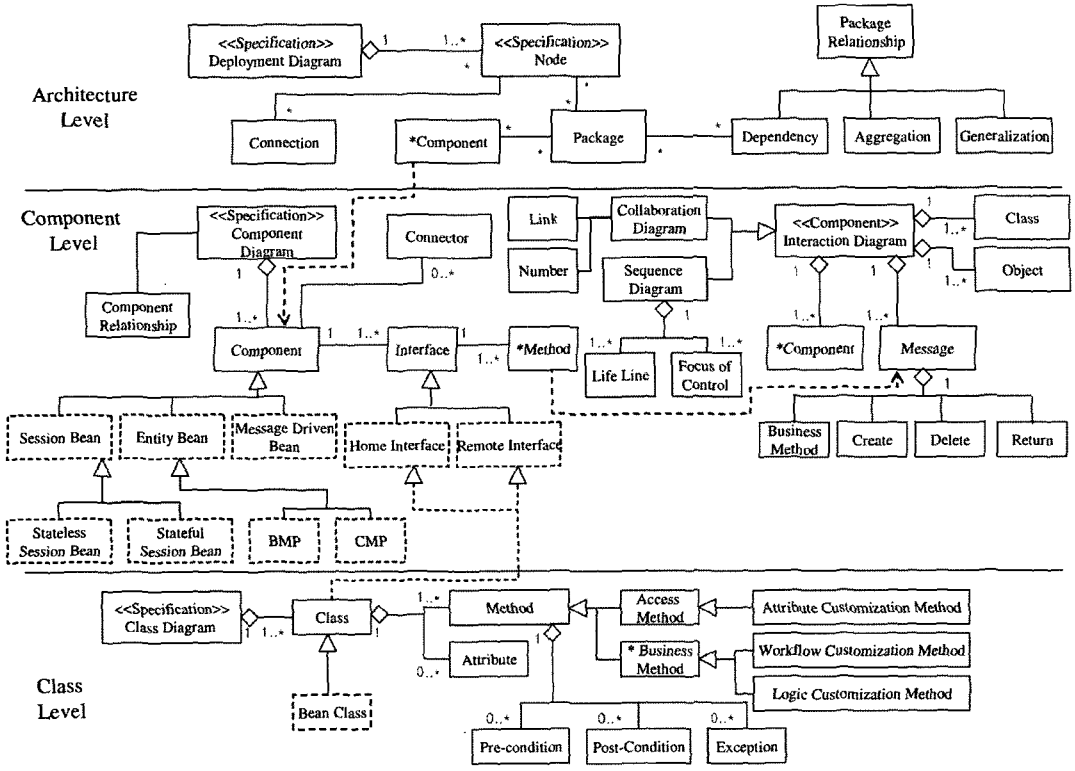


그림 4. 상세 설계 단계 메타 모델

명을 의미한다.

3.3. 컴포넌트 상세 설계 단계 메타 모델

컴포넌트 상세 설계 단계는 개발할 컴포넌트 플랫폼의 아키텍처에 기반을 두고 컴포넌트를 설계하는 단계이다. 즉, EJB, .NET 혹은 CCM 등과 같이 개발 플랫폼에 따라서 추가되어야 할 모델링 요소들이 필요하게 된다. 예를 들어, *EJB*의 경우에 *Entity Bean*과 같은 경우는 컴포넌트의 지속성을 보장하기 위해 컴포넌트 타입을 CMP (Container-Managed Persistence) 타입으로 할 것인지 BMP(Bean-Managed Persistence) 타입으로 할 것인지에 대한 부분도 명세 되어야 하며, 기타 여러 가지 속성 정보들이 모델에 반영되어야 한다.

그림 4는 EJB 플랫폼을 기반으로 하여 EJB 컴포넌트를 개발할 경우에 있어서 이러한 요소들을 반영하한 메타 모델을 나타내고 있다. 그림 4에서 점선 박스는 그림 3의 개략 설계 메타 모델에 EJB와 관련하여 새롭게 추가된 요소들을 의미한다. 모델링 요소로서의 의미는 클래스의 의미와 동일하다. 예를 들어, EJB 컴포넌트인 경우는 컴포넌트의 유형이 3가지 형태로 분류되기 때문에, 그림 4에서 Component 클래스의 하위 클래스로 Entity Bean, Session Bean, Message Driven Bean으로 설정하였으며, 인터페이스 유형 또한Home 과 Remote 형태로 분류하여 표현하였다.

4. 실험 및 평가

이 장에서는 3장에서 제시한 계층형 메타 모델을 실제 도메인에 어떻게 적용했는지에 대한 사례 연구와 여러 도메인에 적용한 결과 얻어진 내용들을 기반으로 기존 연구와 비교하고자 한다.

4.1. 사례 연구

이 절에서는 전자 상거래 도메인을 위한 컴포넌트 개발 과정에 계층형 메타모델을 적용한 사례를 제시한다. 특히, 상세 설계 단계에서는EJB 기반 플랫폼 메타 모델을 적용하였다.

4.1.1. 도메인 분석 단계

도메인 분석 단계에서는 컴포넌트 개발에 필요한 요구사항들을 추출하는데, 특히 컴포넌트의 재사용성을 고려하여 도메인에서의 공통성과 가변성에 대한 요소들을 추출하게 된다. 본 논문에서 제시한 메타 모델을 기반으로 도메인 분석 단계에서는 유스케이스 모델과 개략적인 컴포넌트 다이어그램, 그리고 컴포넌트 다이어그램 내에 들어가는 클래스들을 모델링하게 된다. 그림 5는 전자 상거래 도메인을 대상으로 하여 추출한 공통 기능들을 유스케이스 다이어그램으로 표현한 것이다.

그림 6은 유스케이스 다이어그램을 통해 표현된 유스케이스들에 대해 각 유스케이스 별로 작업 흐름과 유스케이스 내의 가변성을 모델링하기 위해 유스케이스 명세서를 작성한 것이다. 그림 6에 보면 Alternative Flows 부분이 있는데 이는 가변 흐름을 표현하는 것으로, 컴포넌트의 로직(또는 행위) 가변성을 표현한 부분이다.

그림 7은 도메인 분석 단계에서 개발된 컴포넌트 다이어그램이다. 이 예에서는 3개의 컴포넌트인 Customer Management, Product Management, 그리고 Order Management이다.

4.1.2. 개략 설계 단계

그림 8은 도메인 분석 단계에서 개발된 'Order Product' 유스케이스에 대한 시퀀스 다이어그램을 모델링한 것이다.

그림 9는 개략 설계 단계의 메타 모델 가운데 클래스 다이어그램에 대한 메타 모델을 기반으로 상세 클래스 다이어그램을 작성한 것이다.

그림 10은 컴포넌트 인터페이스를 찾고 모델링하기 위해 컴포넌트 수준에서의 상호 작용도(시퀀스 다이어그램)를 작성한 것이다. 그림 10에서 보면 컴포넌트 수준에서의 상호 작용도에서는 특정 컴포넌트 내의 클래스로의 메시지가 해당 컴포넌트의 인터페이스로 추출되는 것을 알 수 있다.

그림 11은 상세 클래스 다이어그램과 상호 작용도를 기반으로 컴포넌트 다이어그램을 작성한 것이다. 이 단계

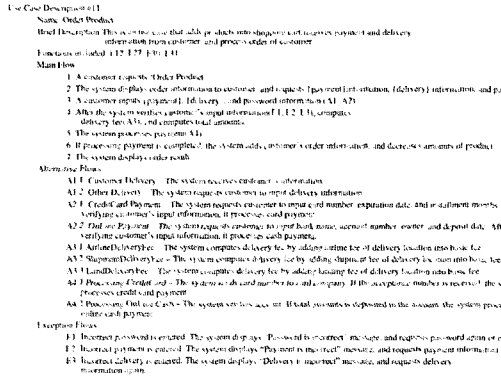


그림 6. 유스 케이스 명세서

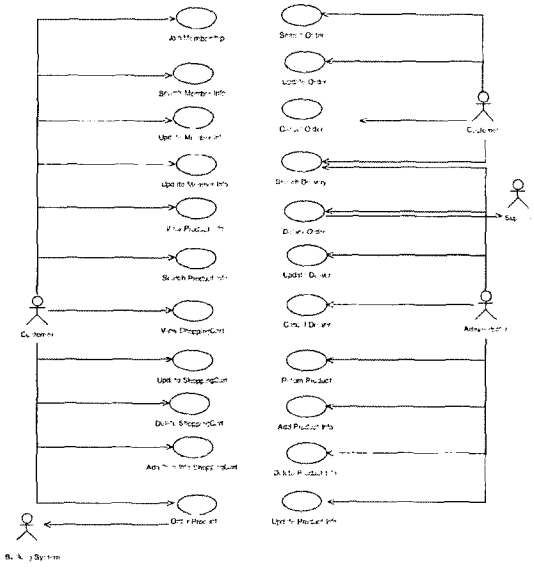


그림 5. 유스 케이스 다이어그램

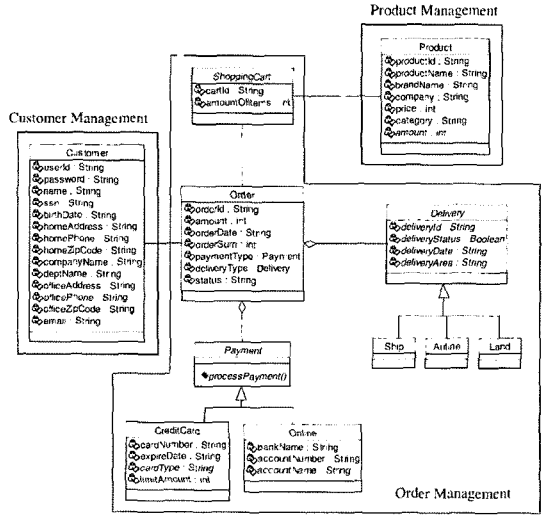


그림 7. 컴포넌트 다이어그램

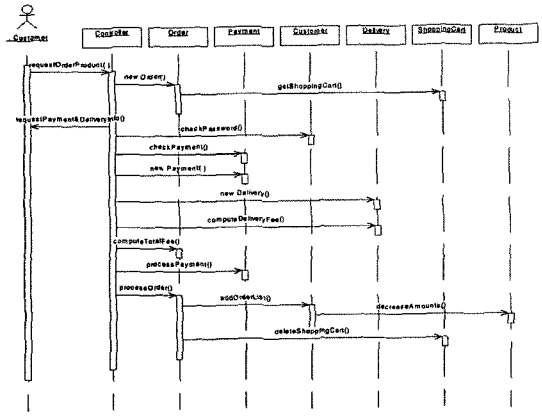


그림 8. 'Order Product' 유스 케이스에 대한 시퀀스 다이어그램

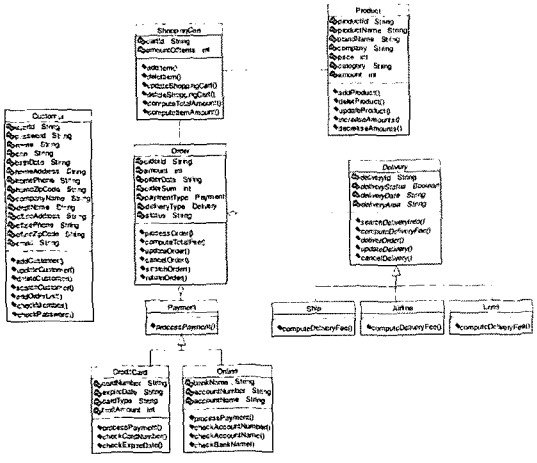


그림 9. 상세 클래스 다이어그램

에서의 컴포넌트 다이어그램은 도메인 분석 단계에서의 컴포넌트 다이어그램보다 보다 많은 정보들이 추가되어 표현되고 있다. 해당 컴포넌트의 인터페이스와 컴포넌트 내의 클래스들 간의 관계 등에 대한 정보가 더 포함되고 있다. 여기서 인터페이스 정보는 그림 10의 상호 작용도를 통해 추출된 인터페이스를 의미한다.

4.1.3. 상세 설계 단계

상세 설계 단계는 구현 단계 이전의 설계 단계로서 개략 설계 단계에서 개발된 다이어그램에 실제 컴포넌트 구현 플랫폼에 맞도록 세부 설계 정보들을 추가하는 단계이다. 본 논문의 사례 연구에서는 EJB 기반 컴포넌트 개발을 실험하였기 때문에 EJB 플랫폼에 맞는 세부 사항들이

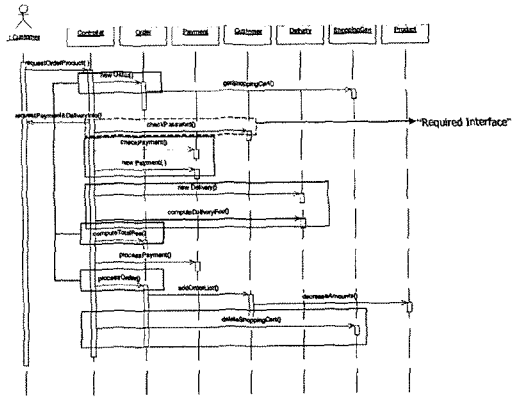


그림 10. 컴포넌트 레벨 상호 작용도

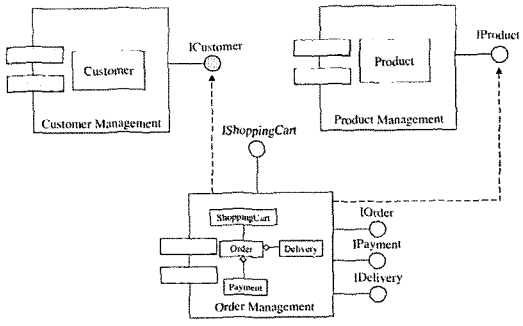


그림 11. 컴포넌트 다이어그램

다이어그램에 추가되었다.

그림 12는 EJB 기반 클래스 다이어그램의 예로서, 클래스 타입이 엔티티 빈(Entity Bean)인지 세션 빈(Session Bean)인지 세부적으로 표현되었으며, 내부 속성이나 함수들 또한 EJB 기반에 맞춰 정제된 것이다.

그림 13은 개략 설계 단계의 시퀀스 다이어그램을 EJB 기반 시퀀스 다이어그램으로 정제한 형태이다. 개략 설계 단계의 시퀀스 다이어그램에 비해 EJB 특성에 맞는 함수들이 추가로 표현되었다.

4.2. 평 가

이 절에서는 사례 연구를 실제 여러 도메인에 적용한 결과들을 바탕으로 기존 연구들과의 비교 평가를 제시한다. 이 절에서는 2가지 관점인 메타 모델 관점에서의 비교와 메타 모델을 적용한 CBD 방법론 관점에서의 비교로 분류하여 평가하고자 한다.

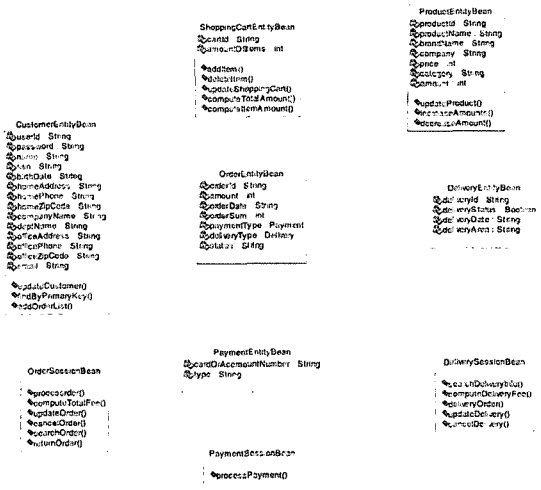


그림 12. EJB 기반 클래스 다이어그램

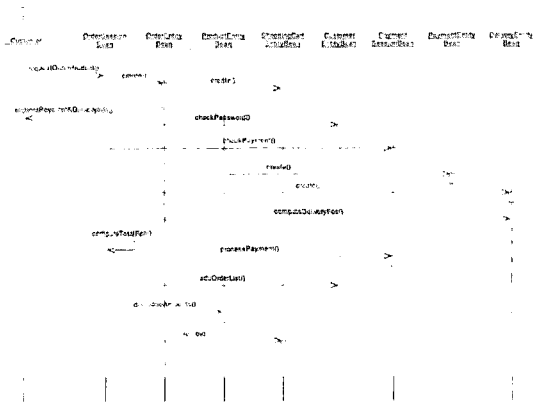


그림 13. EJB 기반 시퀀스 다이어그램

4.2.1. 메타모델 관점에서의 비교

본 논문에서는 비교 평가 항목을 컴포넌트 개발 프로세스에서 요구되는 내용들을 기반으로 메타모델이 지원해야 할 부분들을 정의하여서 비교한다. 표 1에 제시된 바와 같이 기존 연구와 본 연구에서 지원하는 메타 모델을 모두 UML을 기반으로 하고 있다. 이점에서는 모두 객체지향 패러다임을 기반으로 한다는 점에서는 공통점이다.

그러나, 기존 연구들에서 이루어지지 못한 부분들로 MDA나 UML-F 등과 같은 기존 연구들에서는 메타 모델을 계층화하거나, 단계별로 모델을 분류하여 추상화 시키지 못하고 있다. 이로 인하여 많은 개발자들이 컴포넌트 개발 과정 동안에 어느 단계에서 어떤 모델을 어느 정도

표 1. 메타 모델 간의 비교

	MDA	UML-F	계층별 메타모델
모델링 언어	UML	UML	UML
모델의 계층화	-	-	3계층
단계별 모델 분류	-	-	3단계
모델간의 연관성 표현	개별 모델별로 표현	개별 모델별로 표현	개별 및 모델 간 연관성 표현
모델의 추상화 수준	1 level	1 level	3 level(단계별 추상화)
모델의 통합 수준	개별적	개별적	통합적

표 2. CBD 방법론들 간의 비교

방법론모델 요소	CBD96	Catalysis	Fusion	UNIFACE	SCIPIO	제안 기법
컴포넌트	Yes	Yes	Yes	Yes	Yes	Yes
인터페이스	Yes	Yes	Yes	Yes	Yes	Yes
클래스	Yes	Yes	Yes	Yes	Yes	Yes
메소드	Yes	Yes	Yes	Yes	Yes	Yes
공통성	Partial	Partial	Partial	Partial	Partial	Yes
가변성	No	No	No	No	No	Yes
컴포넌트 플랫폼 요소	No	No	No	No	No	Yes
커넥터	No	No	No	No	No	Partial

까지 표현하여야 하는지에 대한 의구심을 갖게 할 뿐만 아니라, 개발한 모델에 대해 모델의 정확성이나 완성도를 파악하기가 매우 어렵다. 더욱이 기존 연구들에서는 각각의 다이어그램 혹은 모델 들에 대해서는 메타 모델로 표현되어 있지만, UML에 있는 다이어그램들 간의 연관성에 대해서는 메타 모델로 표현되어 있지 못하다. 본 논문에서는 이러한 문제점을 해결하기 위하여 모델 간의 연관성에 대한 정보 또한 메타 모델에 반영함으로써, 개발자들로 하여금 모델 간의 연관성을 보다 효율적으로 체크할 수 있도록 지원한다.

4.2.2. CBD 방법론 관점에서의 비교

이 절에서는 본 논문에서 제안한 계층형 메타 모델 기반 컴포넌트 개발 방법과 기존의 컴포넌트 개발 방법들과의 차이점을 살펴본다. 기존 방법론들에서 컴포넌트 개발과 관련된 메타 모델링 요소들을 얼마나 지원하고 있는지 표2에 제시되어 있다.

표2에 제시된 바와 같이 비교 항목으로 표현된 컴포넌트 모델 요소들은 컴포넌트 개발에 있어서 꼭 필요한 모델링 요소들이다. 따라서 CBD 개발에 있어서 이러한 요소들이 제대로 반영되지 않으면, 재사용성이 높은 컴포넌트 개발이 어렵게 된다. 표2에 나타난 것처럼 기존 CBD 방법론들에서는 이러한 모델링 요소들이 일부는 반영되어 있지만, 공통성, 가변성, 컴포넌트 플랫폼 요소 및 커넥터 등과 같은 요소들은 거의 반영되고 있지 못하다. 실제 컴포넌트 기반 애플리케이션 개발할 때 이러한 특성들이 매우 중요한 이슈로 떠오르고 있다. 따라서, 본 논문에서는 단순 컴포넌트 개발을 넘어서 컴포넌트 조립에 있어서 유연성과 재사용성을 높이기 위해 공통성이나 가변성, 컴포넌트 플랫폼 요소 및 커넥터 등과 같은 모델링 요소들을 지원하도록 메타 모델에서 제시하고 있다.

5. 결론 및 향후 연구 과제

지금까지 컴포넌트 개발에 있어서 각 단계별로 개발되어야 할 모델들에 대한 계층형 메타 모델들에 대해서 살펴보았다. 정리해 보면, 도메인 분석 단계에서는 개략 컴포넌트 다이어그램, 유스케이스 모델, 개략 클래스 다이어그램이 도출되고, 개략 설계 단계에서는 디플로이먼트 다이어그램, 컴포넌트 다이어그램, 시퀀스 다이어그램, 상세 클래스 다이어그램이 도출되고, 마지막으로 상세 설계 단계에서는 플랫폼 기반 컴포넌트 다이어그램, 플랫폼 기반 클래스 다이어그램, 상세 시퀀스 다이어그램이 도출되었다. 기존 모델들이 단지 모델 자체에 대한 메타 모델만 제시한 반면에, 여기서 제안한 계층형 메타 모델은 컴포넌트 개발에 한정적으로 특화되어서 적용할 수 있는 메타 모델을 제시하고 있다. 특히 메타 모델을 컴포넌트 개발 단계별로 분류하여 각 단계별로 필요한 모델들에 대한 정보와 모델들 간의 관련성을 메타 모델을 설계함으로써 모델들 간의 일관성이나 모델의 정확성이 보다 더 향상될 것이라 기대된다. 앞으로 향후 연구과제는 이 메타 모델을 정형화 하여 검증하고, 또 이를 자동화 도구로 구현하여 모델 주도 기반의 컴포넌트 개발 기법으로 발전시키고자 한다.

참 고 문 헌

1. Heineman, G. T. and Council, W T., Component-based Software Engineering, Addison Wesley, 2001.
2. Whitehead, K., Component-based Development: Principles and Planning for Business Systems, Addison-Wesley, 2002.
3. Jacobson, I., Griss, M., and Jonsson, P., Software Reuse: Architecture, Process, and Organization for Business Success, ACM Press, New York, June 1997.
4. Frankel, D. S., *Model Driven Architecture-Applying* MDA to Enterprise Computing, Wiley Publishing, Inc., 2003.
5. Kleppe, Anneke G., et al., MDA Explained, Addison-Wesley, 2003.
6. Scott, Kendall, UML Explained, Addison-Wesley, 2001.
7. Richter, Charles, Designing Flexible Object-Oriented Systems with UML, Macmillan Technical Publishing, 2001.
8. Cho, E. S., Kim, S. D., and Rhew, S. Y., "A Domain Analysis and Modeling Methodology for Component Development", International Journal of Software Engineering and Knowledge Engineering, Vol.14, No.2, pp. 221~254, April 2004.
9. Sun *Microsystems Inc.*, "*Enterprise* JavaBeans Specifications", at URL: http://www.javasoft.com.
10. Szyperski, C., Component Software: Beyound Object-Oriented Programming, Addison Wesley Longman, Reading, Mass., 1998.
11. Fayad, M. E. and Schmidt, D. C., "Object-Oriented Application Frameworks", Communications of the ACM, Vol.40, No.10, Oct. 1997.
12. Fontoura, M. F., Pree, W., and Rumpe, B., "UML-F: A Modeling Language for Object-Oriented Frameworks", 14th European Conference on Object Oriented Programming (ECOOP 2000), Lecture Notes in Computer Science 1850, Springer, 63-82, Cannes, France, 2000.
13. Short K., Component Based Development and Object Modeling, Sterling Software, 1997.
14. Harmon P., "Visual Modeling Tools, CASE vendors, and component methods," Component Strategies, 1999.
15. Veryad R., "SCPIO: *Aims, Principles and Structure*," SCPIO Consortium, April 1998.
16. D'souza D. F. and Wills A. C., Objects, Components, and Components with UML, Addison-Wesley, 1998.

이 숙 희 (oleesh@skuniv.ac.kr)

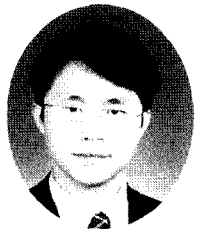
1979년 숙명여자대학교 독문학과졸업(학사)
1982년 동국대학교 경영대학원 정보처리학과졸업(석사)
1991년 성균관대학교 대학원 통계학과 졸업(박사)
1987년~1993년 동신대학교 전자계산학과 교수
1993년~현재 서경대학교 인터넷정보학과 교수

관심분야 : 객체지향 소프트웨어 개발 방법론, 소프트웨어 테스트, 컴포넌트 기반 소프트웨어공학

조 은 숙 (escho@seoil.ac.kr)

1993년 동의대학교 전자통계학과 이학사
1996년 숭실대학교 대학원 컴퓨터학부 공학석사
2000년 숭실대학교 대학원 컴퓨터학부 공학박사
2002년~2003년 한국전자통신연구원 초빙연구원
2000년~현재 서일대학 소프트웨어공학과 전임강사

관심분야 : 개발방법론, 컴포넌트 기반소프트웨어 공학, 소프트웨어 아키텍처, 유비쿼터스 컴퓨팅



김 철 진 (chuljin777.kim@samsung.com)

1996년 경기대학교 전자계산학과 이학사
1998년 숭실대학교 대학원 컴퓨터학부 공학석사
2004년 숭실대학교 대학원 컴퓨터학부 공학박사
2004년 카톨릭대학교 컴퓨터 정보공학부 강의전담교수
2004년~현재 삼성전자 DSC(Digital Solution Center) 책임연구원

관심분야 : 컴포넌트 기반 소프트웨어 공학, 컴포넌트 커스터마이제이션, 개발 방법론