컴포넌트 설계를 MDA/PIM으로 명세하기 위한 UML프로파일

(A UML Profile for Specifying Component Design as MDA/PIM)

민 현 기 † 김 수 동 ††
(Hyun Gi Min) (Soo Dong Kim)

요 약 컴포넌트 기반 개발(CBD) 기술은 컴포넌트 재사용을 통해 S/W 개발 생산성을 높이는 기술로 각광을 받고 있다. 모델기반 아키텍처 (Model Driven Architecture, MDA)는 설계 모델을 점진적으로 변환하여 S/W를 자동으로 생성하는 새로운 개발 방식이다[1]. CBD기술은 재사용을 통하여, MDA 기술은 모델 변환을 통하여 S/W 개발 생산성을 높이므로, 이 두 기술의 접목은 SW 재사용과 자동 생산의 두 가지 장점을 모두 이룰 수 있다. 이를 위해서는 설계된 컴포넌트를 MDA의 플랫폼 독립적 모델 (PIM)로 명세하여야 하며, UML 확장 장치 즉 CBD용 UML 프로파일이 요구된다. 본 논문에서는 명세할 컴포넌트의 구성요소를 메타 모델로 정의하고, 각 구성요소를 PIM으로 명세하기 위한 컴포넌트용 UML 프로파일을 제안한다. 이 프로파일은 컴포넌트 명세를 위한 스테레오 타입, 구문(Syntactic), 의미 (Semantic), 규약(Contract) 및 표기법으로 이루어진다. 제안된 프로파일은 MDA 표준 규약의 기반인 Meta Object Facility (MOF)를 확장 적용한 것이므로 여러 MDA 기법과 도구들과 호환을 제공한다. 제안된 프로파일을 적용하면 CBD와 MDA의 고유 기능과 장점을 접목하여 높은 개발 생산성, 이식성, 상호 운용성, 및 유지보수성을 가질 수 있다.

키워드 : 컴포넌트 기반 개발 (CBD), 모델 기반 아키텍처 (MDA), UML 프로파일

Abstract Component Based Development (CBD) is appealing as a technology to improve the productivity of software development through component reuse. Model Driven Architecture (MDA) is a new development paradigm which automatically generates application by transforming design models incrementally. Since both reusability of CBD and model transformation of MDA increase software productivity, integration of two technologies is desirable. To enable this technology integration, we need to devise a UML profile for specifying component design as a PIM. In this paper, we first define a meta-model for components, and propose a UML profile which is used to specify elements of component design as PIM. Since the proposed profile is based on Meta Object Facility (MOF) from which is MDA is derived, it is consistent and compatible with existing MDA methods and tools.

Key words : Component, Model Driven Architecture (MDA), UML Profile

1. 서 론

CBD 기술은 컴포넌트 재사용을 통해 S/W 개발 생산성을 높이는 기술로 학계와 산업계에 보급되어 있다. CBD의 특징은 도메인 컴포넌트를 통해 큰 단위의 재사용이 가능하고, 여러 컴포넌트 사용자 마다 서로 다른 요구사항을 지원하기 위한 커스트마이제이션 기법을 제공하고, 인터페이스와 구현을 분리함으로서 유지보수성을 증가시킬 수 있다[2]. 그러나 컴포넌트 제품은 컴포넌트 미들웨어 플랫폼에 종속적이다. 따라서 컴포넌트의 미들웨어가 변경된다면, 그 컴포넌트 미들웨어에 적용하기 위해 컴포넌트를 새로 설계하고, 개발해야 하므로 중복 노력이 필요하고, 생산성이 낮아지게 된다.

소프트웨어 개발을 위한 아키텍처인 MDA는 Object Management Group(OMG)에 의해서 정의 되었다. MDA는 설계 모델을 점진적으로 변환하여 S/W를 자동으로 생성하는 새로운 개발 방식이다[4]. 이를 지원하기 위한 MDA의 핵심은 소프트웨어 개발 프로세스에서의 모델이다. MDA에서는 시스템을 모델링하는 행위가 곧

· 이 논문은 2004년도 한국학술진흥재단의 지원에 의하여 연구되었음 (KRF-2004-005-D00172)
† 정 회 원 : 숭실대학교 정보과학대학 연구원
hgmin@otlab.ssu.ac.kr
†† 종신회원 : 숭실대학교 컴퓨터학부 교수
sdkim@comp.ssu.ac.kr
논문접수 : 2004년 8월 18일
심사완료 : 2005년 1월 28일

제품을 구현하는 생산 작업이 된다. MDA는 PIM을 PSM으로 변환하고, 다시 PSM을 코드로 생성할 때, MDA 도구를 이용하여 자동으로 변환 작업을 수행할 수 있다. 설계 작업을 통해 구현까지 자동으로 생산이 가능한 MDA는 생산성, 이식성, 상호운용성, 유지보수성, 문서의 일관성의 특징을 갖는다.

CBD와 MDA기술이 접목된다면, CBD 생산성 및 유지보수성 등이 더욱더 증가할 것이다. 만약 PIM 설계시 컴포넌트의 단위가 고려되고, 컴포넌트의 인터페이스와 구현 객체를 분리하고, 커스트마이제이션을 위한 가변성(Variability)도 PIM에 설계가 된다면 MDA를 이용하여 이러한 정보를 모델을 통해 컴포넌트 구현에 적용할 수 있다. MDA의 PIM은 특정 플랫폼에 종속적이지 않게 설계한 모델이다. 즉, 컴포넌트의 단위, 인터페이스, 커스트마이제이션을 위한 가변성등의 CBD기술을 고려하지 않았다. 컴포넌트 기반의 PSM으로 자동 변환하기 위해서는 PIM 수준의 컴포넌트용 UML 프로화일이 필요하다.

본 논문에서는 명세할 컴포넌트의 구성요소를 메타모델로 정의하고, 각 구성요소를 PIM으로 명세하기 위한 컴포넌트용 UML 프로파일을 제안한다. 이 프로파일은 컴포넌트 명세를 위한 스테레오 타입, 구문(Syntactic), 의미(Semantic), 규약(Contract) 및 표기법으로 이루어진다. 제안된 프로파일은 MDA 표준 규약의 기반인 MOF를 확장 적용한 것이므로 여러 MDA 기법과 도구들과 호환을 제공한다. 제안된 프로파일을 적용하면 CBD와 MDA의 고유 기능과 장점을 접목하게 되므로 높은 개발 생산성, 이식성, 상호 운용성, 및 유지보수성을 가질 수 있다.

2장에서는 MDA에 관한 핵심 사상을 소개하고, 3장에서는 컴포넌트를 PIM에 명세하기 위한 컴포넌트 메타 모델을 제시하고, 컴포넌트 기반의 PIM의 필요성 및 재사용 기법은 4장에서 설명하며, 5장에서는 CBD용 UML 프로파일의 표현법을 제시하고, 6장에서는 본 논문을 평가하고, 7장에서는 결론을 내린다.

2. 관련 연구

2.1 Model Driven Architecture(MDA)

MDA는 소프트웨어의 설계 모델을 명세하고, 이를 상세 설계 모델과 코드로 변환하여 프로그램을 자동 생성하는 새로운 개발 기술이다. MDA는 두 개의 설계 모델을 사용한다. 플랫폼 독립적 모델(Platform Independent Model, PIM)은 구현에 사용될 프로그래밍 언어, 시스템 S/W, 네트워킹 등 특정 환경에 비종속적인 설계 모델이며, 플랫폼 종속적 모델(Platform Specific Model, PSM)은 구현 환경의 특징을 고려한 상세 설계 모델이다.

그림 1과 같이 PIM에서 PSM으로 변환 (Transformation)하고 PSM에서 코드로 변환 규칙을 사용하여 자동 변환함으로써, 기존의 개발자가 직접 상세 설계 및 코딩을 해야 하는 노력을 크게 절감해 준다. 또한, 코드에서 PSM으로, PSM에서 PIM으로 역공학을 통하여 기존 시스템의 설계 모델 추출이 가능하고, 유지 보수시에는 PIM의 수정만으로 코드의 재생성을 통해 유지 보수가 가능하다. 그러므로 유지 보수된 구현 산출물과 설계서가 일치하게 된다.

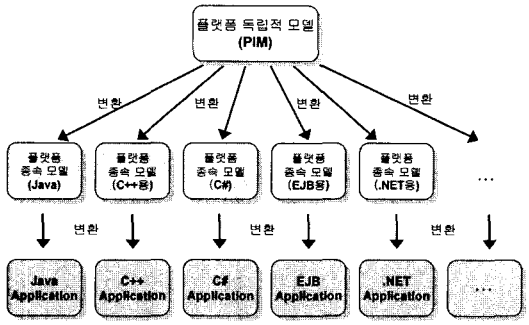


그림 1 변환을 통한 소프트웨어 생성 과정

2.2 UML 프로파일(Profile)

UML 프로파일은 UML 기반의 설계를 MDA의 PIM이나 PSM 모델로 표현하기 위해서 필요한 추가적인 명세의 표준장치들로 구성된다. UML 프로파일은 스테레오 타입(Stereo type), 제약 사항(Constraint), 꼬리표값(Tagged Value) 및 아이콘으로 구성된다. 각 확장된 명세 장치는 해당되는 구문, 그래픽 표기법, 의미(Semantics)로 구성된다[3].

스테레오 타입은 UML 메타모델의 각 요소에 첨부가 가능하며 이름을 가지고 있다. 예를 들면, UML 모델에서 각각의 클래스마다 이 스테레오타입을 적용할 수 있다. 스테레오 타입으로 정의된 클래스는 일반적인 의미의 클래스에서 새로운 의미를 갖는 클래스로 표현된다. 제약사항은 UML 메타모델의 각 요소의 제약적인 사항을 묘사한다. 제약사항은 Object Constraints Language (OCL)로 표현될 수 있으며, 시스템의 제약적인 내용을 서술한다[4]. 이름과 타입을 갖는 꼬리표값은 특정 스테레오 타입에 첨부되어 부가적인 속성값을 표현한다.

MDA를 이용한 개발에서는 소프트웨어 설계를 UML 프로파일의 정의에 따라 명세해야 매핑 규칙에 의해 자동으로 변환 될 수 있다. 현재의 UML 프로파일은 플랫폼 종속적인 정보를 표현하기 위한 명세 방법과 코드

생성시 필요한 추가적인 상세 설계 정보의 명세방법을 제공한다. 그림 2와 같이, 컴포넌트 기반 PIM은 컴포넌트 명세용 UML 프로파일의 명세 장치를 준수(Conforms to)해야 하며, PSM으로 변환될 때 변환 규칙 및 변환 도구는 이 프로파일을 참조(Refers to)하게 된다. 이렇게 만들어진 PSM은 컴포넌트 설계 정보를 가진 모델이 되어, 이를 기반으로 생성될 프로그램 코드는 컴포넌트 설계를 구현한 형태를 가지게 된다.

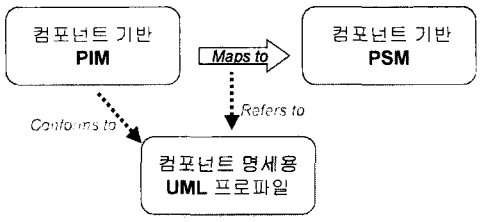


그림 2 UML 프로파일의 역할

2.3 EDOC를 위한 UML 프로파일

OMG에서 채택한 PIM 수준의 명세가 가능한 Enterprise Distributed Object Computing(EDOC)는 S/W 컴포넌트의 가변성을 표현하기 어려우며, PSM 수준의 컴포넌트 관련 프로파일인 EJB용 UML 프로파일[6]과 CORBA용 UML 프로파일[7] 등이 있다.

EDOC를 위한 프로파일(UML Profile for EDOC)은 MDA를 위한 UML 1.4 기반의 모델링 프레임워크이다[8]. EDOC는 7개의 스펙으로 구성된다. 첫째, Enterprise Collaboration Architecture(ECA)[9]은 모델링 프레임워크에 의해 EDOC 시스템을 간단하게 개발하기 위한 아키텍처이다. 둘째, Metamodel and UML Profile for Java and EJB[10]은 Java와 EJB에 관한 메타 모델이다. 셋째, Flow Composition Model(FCM)[11]은 어플리케이션간의 정보 흐름과 상호관계를 명세할 수 있다. 넷째, UML Profile for Patterns[12]은 UML 패키지 표기법을 이용해서 비즈니스 기능 객체 패턴(Business Function Object Patterns)를 표현할 수 있다. 다섯째, UML Profile for ECA[13]는 표 1과 같이 엔티티, 이벤트, 비즈니스에 관한 스펙을 제공한다. 여섯째, UML Profile for Meta Object Facility[14]는 UML과 MOF간의 매핑을 명세한다. 마지막으로 UML Profile for Relationships[15]는 비즈니스 모델에서의 관계에 관한 표준을 명세한다.

표 1 UML Profile for ECA 스펙에서의 요소 매핑

Metamodel Element	UML Profile Element	UML Base Class
Data Manager	Data Manager	Class
Entity Data	Entity Data	Class
Entity	Entity	Class
Entity Role	Entity Role	Class
Class Key	Key	Class
Key Element	Key Element	Attribute
Key Attribute	Key Attribute	Attribute
Foreign Key	Foreign Key	Attribute
Data Probe	Data Probe	Class

2.4 Fontoura의 UML-F[16]

프레임워크는 유사한 특성의 어플리케이션의 소프트웨어 아키텍처를 구현한다. 이러한 특성은 어플리케이션에 종속적인 코드를 통해 특화될 수 있다. UML을 확장하여 제안한 객체지향 프레임워크를 위한 모델 언어(A Modeling Language for Object-Oriented Frameworks, UML-F)는 객체지향 프레임워크의 설계에 사용된다. UML-F는 프레임워크의 가변점(Variation Point)의 표현이 가능하다. UML-F에서의 프레임워크는 큰 단위로 미리 정의된 패턴과 함께 구현된 몇 개의 컴포넌트로 정의한다.

UML-F에서는 설계를 위해 UML의 제약사항(Constraint)을 확장하여 {appl-class}, {variable}, {extensible}, {static}, {dynamic}, {incomplete}, {for all new methods}, {optional} 8개의 새로운 요소를 제안했다. 설계 모델에서 구현을 위한 상세 설계로 변환할 수 있다. 하지만, 제안한 스테레오 타입의 의미가 추상적이기 때문에 구현을 위한 구체적인 상세설계로의 매핑이 어렵다.

3. 컴포넌트의 메타모델

본 장에서는 컴포넌트의 구성요소를 그림 3에서와 같이 UML을 이용하여 메타 모델로 정의한다. 컴포넌트는 클래스, 인터페이스, 가변성, 워크플로우로 이루어진다. 컴포넌트는 하나 이상의 클래스로 구성되며, 이 클래스

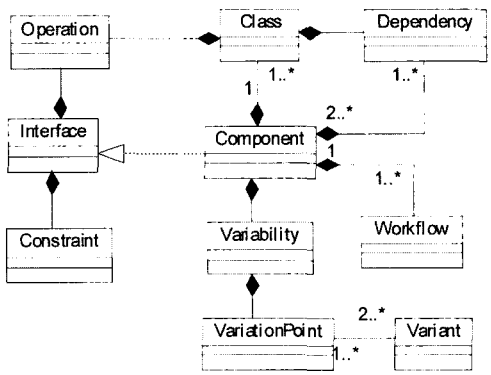


그림 3 컴포넌트의 메타 모델

들은 컴포넌트에서 선언된 인터페이스를 구현하기 위해 사용된다. 인터페이스를 통해서 컴포넌트의 내부 구현을 숨기며, 서비스 제공이 가능하다.

인터페이스는 Provided 인터페이스와 Customize 인터페이스, 그리고 Required 인터페이스로 구성된다[18]. Provided 인터페이스는 컴포넌트가 다른 컴포넌트 즉, 클라이언트에게 서비스를 제공하기 위한 인터페이스이다. Customize 인터페이스는 컴포넌트의 가변성을 설정하기 위한 인터페이스이다. Required 인터페이스는 컴포넌트가 필요로 하는 다른 컴포넌트의 인터페이스를 명세한다. 즉, 서로 다른 패밀리 멤버를 위해 해당 컴포넌트를 사용할 때 패밀리 멤버의 목적에 맞게 설정하기 위해 사용되는 인터페이스이다.

가변성이란 컴포넌트가 제공해야 하는 기능이 패밀리 멤버들 간에 서로 다른 기능이다[19]. 가변성은 가변점과 가변치(Variant)로 표현된다. 가변점이란 가변성이 발생하는 위치를 의미하며 주로 기능 단위로 나타난다. 가변치란 가변점에서 멤버가 실제로 갖게되는 값을 의미한다. 이러한 가변성이 컴포넌트에 적용되어야 각 멤버별 재사용성이 가능하다[20].

가변성의 종류는 속성 가변성, 로직 가변성, 워크플로우 가변성, 인터페이스 가변성, 영속성 가변성 5종류로 분류한다[21]. 속성 가변성은 프로덕트 패밀리의 멤버들 사이에 같은 속성을 갖지만, 그 타입 등이 패밀리 멤버마다 서로 다른 경우이다. 로직 가변성은 프로덕트 패밀리 멤버 사이에 같은 기능을 제공하지만, 그 내부 알고리즘이 다른 로직이다. 워크플로우 가변성은 그 내부 메시지 흐름의 순서가 서로 다른 워크플로우이다. 인터페이스 가변성은 컴포넌트 사용자마다 원하는 인터페이스의 모양이 다른 것이며, 영속성 가변성은 컴포넌트와 연동될 데이타베이스의 스키마가 패밀리 멤버마다 다른 경우이다.

가변점에 채워질 가변치의 종류가 분석 단계에서 모두 식별될 수 있는 경우는 가변성의 영역(Scope)이 Close 영역이고, 모두 식별될 수 없는 경우는 Open 영역이다. Close 영역은 가변치를 모두 식별하여 컴포넌트에 모두 포함시킨 후, 컴포넌트를 운영할 때, Customize 인터페이스에 의해 적절한 가변치가 선택된다. 너무 많은 종류의 Close 영역이나 Open영역은 Customize 인터페이스를 통해 적절한 가변치를 실행 타임에 플러그 인(Plug-In)한다.

4. 컴포넌트용 프로파일을 적용한 컴포넌트 개발 프로세스

본 장에서는 기존의 MDA개발 프로세스에 컴포넌트 PIM 설계를 위한 컴포넌트용 프로파일(UML Profile for Component)을 적용한 컴포넌트 개발 프로세스를 그림 4와 같이 제안한다. 요구사항 분석 단계에서는 요구사항 명세서를 분석하여 기능적 요구사항과 비기능적 요구사항을 도출한다.

PIM 설계 단계에서는 기존 컴포넌트 개발 프로세스에서의 컴포넌트의 개략 설계와 같은 업무를 수행한다. 컴포넌트 단위를 설계하고, 컴포넌트들간의 관계를 명세하며, 컴포넌트가 제공하는 인터페이스를 설계하고, 설계한 인터페이스를 실현화를 위한 객체들과 객체들간의 관계를 명세한다[22]. 고품질 컴포넌트를 위한 컴포넌트의 가변점과 가변치를 명세한다. 분석된 컴포넌트는 컴포넌트용 UML 프로파일을 사용하여 컴포넌트 플랫폼에 독립적인 PIM으로 설계된다. 본 논문에서는 컴포넌트 플랫폼 독립적인 설계를 컴포넌트용 PIM(Component Based PIM, CB-PIM)라고 정의한다. CB-PIM은 UML 2.0을 기반으로 설계하며, UML 2.0에 없는 표기방법은 컴포넌트용 UML 프로파일을 사용한다.

PSM 설계 단계에서는 자동 변환을 위한 MDA 도구에 의해 컴포넌트 개략 설계물인 CB-PIM으로 특정 플랫폼을 고려한 컴포넌트 상세 설계물인 PSM을 생산한다. 자동 변환 방법은 컴포넌트용 UML 프로파일을 이용하여 표기된 CB-PIM을 분석한 후, 특정 플랫폼을 위한 UML 프로파일에서 제공하는 표기법으로 매핑하여

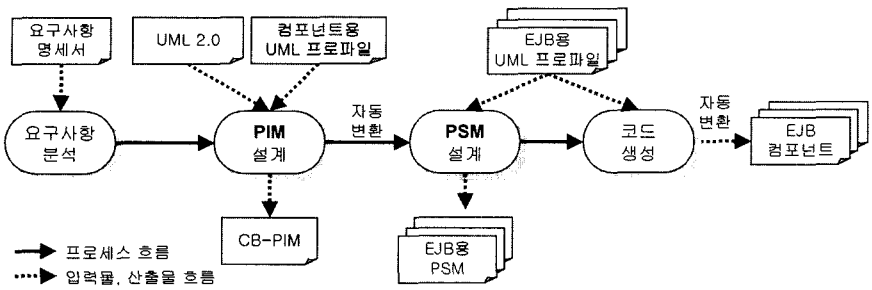


그림 4 컴포넌트용 프로파일을 적용한 컴포넌트 개발 프로세스

PSM을 생성한다.

코드 생성 단계에서는 생성된 PSM을 해당 UML 프로파일과 자동 변환 도구에 의해 최종 컴포넌트 산출물을 생성한다. 본 논문에서 제안한 PIM 수준의 UML Profile은 컴포넌트 플랫폼 종속적이지 않고 컴포넌트를 추상화할 수 있는 범용적인 UML 프로파일이다. 따라서, UML Profile for EJB, CORBA등을 준수한 다양한 PSM 수준의 모델로 자동 변환이 가능하여, 모델의 재사용성과 확장성이 좋다. 그러므로, 컴포넌트용 UML 프로파일을 적용한 CB PIM을 이용하여, EJB 컴포넌트를 만든 후, 사용자의 요구에 의해 컴포넌트의 플랫폼이 변경되면, CB-PIM을 다시 .NET 컴포넌트를 위한 PSM으로 자동 변환하여 .NET 컴포넌트 생성이 가능하다. 이런 과정을 거쳐 CB-PIM이 진화되고, 최종적으로 컴포넌트의 품질이 향상된다.

5. 컴포넌트 기반 PIM 명세를 위한 UML 프로파일

본 장에서는 컴포넌트 설계를 위한 UML 프로파일을 제안한다. 제안하는 UML 프로파일은 CB-PIM을 설계하기 위해 UML 2.0[23]을 사용하며, UML에서 직접 지원하지 않는 요소는 MOF를 확장해서 사용한다. 본 논문에서 제안하는 컴포넌트용 UML 프로파일은 MOF를 준수하므로, MOF를 준수한 UML 도구 및 MDA 도구에서는 제안할 명세기법을 사용하여 CB PIM 설계가 가능하다. 컴포넌트 설계를 위한 UML 프로파일 이외의 표기법은 표 1과 같은 EDOC를 위한 UML 프로파일을 병행 사용한다.

5.1 컴포넌트 명세를 위한 항목

CBD에서 컴포넌트는 관련된 객체들을 캡슐화하기 위한 기본적인 단위이다. 그러므로 PIM에서 컴포넌트를 구성하고 있는 객체를 명세해야 한다. UML 2.0에 의해 컴포넌트와 컴포넌트의 관계는 의존관계로 표현이 가능하고, 컴포넌트의 내부를 표현하기 위해 클래스 명세와 클래스들간의 관계는 클래스 다이어그램을 사용한다. 컴포넌트 내부(Internal Part)와 컴포넌트 외부 환경(External Environment)과의 상호작용의 표현은 포트(Port)를 사용하여 더 명확하게 표현한다[24]. 또한 컴포넌트 내부 객체들간의 워크플로우는 순차도와 협력도를 사용하여 설계 한다. 컴포넌트 명세를 위한 UML 프로파일은 표 2와 같다. '적용되는 UML 요소' 항목은 표기법이 적용될 수 있는 UML 요소를 나타낸다.

컴포넌트는 시스템 컴포넌트와 비즈니스 컴포넌트로 분류한다[25]. 시스템 컴포넌트는 관련된 컴포넌트들에게 메시지를 전달하고, 컴포넌트들간의 협력을 돕는다. 또한 시스템 컴포넌트는 클라이언트 프로그램들에게 서비스를 제공하기 위해 비즈니스 컴포넌트를 위한 인터페이스 역할을 하며, 클라이언트와 상호작용을 한다[26]. 이 시스템 컴포넌트는 비즈니스 컴포넌트를 위한 퍼사드(Façade)와 중계자(Mediator) 패턴[27] 역할을 수행한다. CB-PIM에서 시스템 컴포넌트는 «SysComponent» 스테레오 타입을 사용하여 표현한다.

비즈니스 컴포넌트는 비즈니스 로직을 수행하고, 데이타 저장과 관련된 비즈니스 데이타를 관리하는 객체들로 구성된다. 그러므로 비즈니스 컴포넌트들은 시스템 컴포넌트를 통해 사용자에게 서비스를 제공한다. CB-PIM에서 비즈니스 컴포넌트는 «BizComponent» 스테레오 타입을 사용하여 표현한다.

클래스의 속성값이 데이타 베이스나 파일 시스템 등에 영구적으로 관리가 되어야 하면, 해당 클래스는 «Persistence»를 사용한다. 클래스가 비즈니스 로직만을 제공하거나, 구조체와 같이 데이타의 전달을 위한 객체(Value Object)[27]처럼 상태값을 영구적으로 보관하지 않는 경우에는 그렇지 않은 클래스는 «Transient» 스테

표 2 컴포넌트 명세를 위한 UML 프로파일 요소

요 소	명세 표기법	적용되는 UML 요소	비 고
컴포넌트	«component»	컴포넌트	UML 2.0 호환
시스템 컴포넌트	«SysComponent»	컴포넌트	
비즈니스 컴포넌트	«BizComponent»	컴포넌트	
비영속적인 클래스	«Transient»	클래스	
영속적인 클래스	«Persistence»	클래스	Default
기본키	«UniqueId»	변수	
동기 메시지	«Sync»	메소드	Default
비동기 메시지	«Async»	메소드	
컴포넌트간 호출 관계	«use»	의존관계	UML 2.0 호환
클래스간 관계	상속, 복합, 연관, 의존관계 등	클래스	UML 2.0 호환
제약조건	{ }, pre:, post:, inv:	클래스, 메소드, 연관관계	OCL
알고리즘	문자로 표기	메소드	OCL, ASL

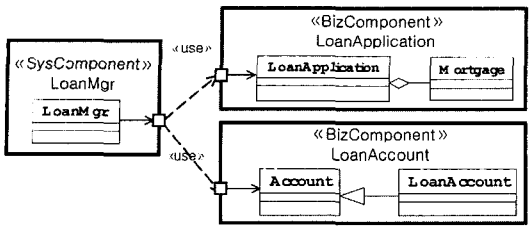


그림 5 컴포넌트 명세의 예제

레오 타입을 사용한다. 비동기 메시지를 사용하는 경우에는 «Async» 스테레오 타입을 클래스도, 순차도, 협력도의 메소드에 사용한다.

그림 5와 같이 LoanManagement 컴포넌트는 «SysComponent» 스테레오 타입에 의해 시스템 컴포넌트로 식별되며, LoanManagement 컴포넌트는 하나의 객체를 포함하고 있다. LoanApplication 컴포넌트는 «BizComponent» 스테레오 타입에 의해 비즈니스 컴포넌트로 명세 되고, 이 LoanApplication 컴포넌트는 LoanApplication과 Mortgage 클래스로 실체화된다.

5.2 인터페이스 명세를 위한 항목

인터페이스는 컴포넌트를 위한 규약을 제공한다. CBD에서 컴포넌트의 유지보수성과 대치성을 증가시키기 위해 컴포넌트의 구현으로부터 인터페이스를 분리한다. 그래서 CB-PIM에서의 컴포넌트뿐만 아니라 인터페이스를 명세하는 방법도 필요하다. 인터페이스는 Provided 인터페이스, Customize 인터페이스, Required 인터페이스 3종류의 인터페이스 타입으로 분류하며, 인터페이스는 시그네처(Signature)와 제약조건은 사전조건(Pre-condition), 사후조건(Post-condition)과 불변조건(Invariant)으로 정의된다. 인터페이스 명세를 위한 UML 프로파일은 표 3과 같다.

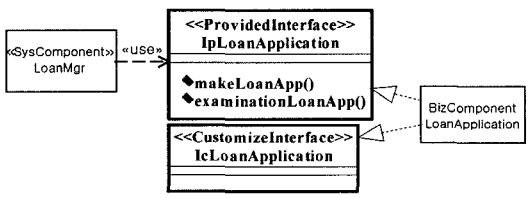
Provided 인터페이스는 컴포넌트에 의해 제공되는 서비스를 명세하고, 이것은 클라이언트 프로그램 또는 다른 컴포넌트들로부터 호출당한다. 이 Provided 인터페이스를 명세하기 위해 «ProvidedInterface» 스테레오 타입을 사용한다. 또한 다른 종류의 인터페이스와 쉽게 식별하기 위해 Provided 인터페이스의 이름은 'Ip'접두사를 사용하여 명명한다.

Customize 인터페이스는 컴포넌트의 행위를 고객의 용도에 맞게 가공하기 위한 메커니즘을 제공한다. 이러한 목적을 위해 Provided 인터페이스와는 분리하여 특별히 설계된 인터페이스를 사용한다. 우리는 이러한 목적의 인터페이스를 Customize 인터페이스라고 하며 «CustomizeInterface» 스테레오 타입으로 명세 한다. Customize 인터페이스는 가변점에 가변치를 할당하기 위해 사용되며, 이름은 'Ic'접두사를 사용한다.

Required 인터페이스는 컴포넌트에 의해 호출되는 외부의 컴포넌트 인터페이스를 명세 한다. Required 인터페이스는 «RequiredInterface» 스테레오 타입으로 명세된다. Required 인터페이스의 이름은 'Ir'접두사를 사용하여 명명한다. 도구에 의해 컴포넌트 조립시 Required 인터페이스와 Provided 인터페이스의 설계를 비교 분석하여, 도구에 의해 사용 가능한 컴포넌트인지 테스트 및 시뮬레이션도 가능하다.

사전조건, 사후조건, 불변조건을 명세하기 위한 제약사항을 명세하기 위해 OCL을 사용한다. CB-PIM에서 스테레오 타입, 꼬리표값으로만 표현 못하는 의미를 OCL을 사용하여 작성된 PIM은 구현하기 위한 보다 명확한 정보를 내포 가능하게 한다.

그림 6과 같이 인터페이스는 CB-PIM에서 표현될 수 있고, 그 인터페이스가 구체화되는 컴포넌트를 명세할 수 있다. LoanApplication 컴포넌트는 IpLoanApplication 인터페이스와 커스트마이제이션을 위한 IcLoanApplication 인터페이스를 실체화하는 것을 명세 한다. 또한 LoanMgr 시스템 컴포넌트는 IpLoanApplication 인터페이스의 서비스를 요구한다. 즉, IpLoanApplica-



.그림 6 인터페이스 명세의 예제

표 3 인터페이스 명세를 위한 UML 프로파일 요소

요 소	명세 표기법	적용되는 UML 요소	비 고
인터페이스	«Interface»	인터페이스	UML 2.0 호환
Provided 인터페이스	«ProvidedInterface»	인터페이스	UML 2.0 호환
Customize 인터페이스	«CustomizeInterface»	인터페이스	
Required 인터페이스	«RequiredInterface»,	인터페이스	UML 2.0 호환
시그네처	함수명(매개변수 :타입,...) :반환타입	오퍼레이션	UML 2.0 호환
제약조건	{ }, pre:, post:, inv:	클래스, 메소드, 연관관계	OCL
알고리즘	문자로 표기	메소드	OCL, ASL

tion 인터페이스는 LoanMgr 컴포넌트의 Required 인터페이스이다.

5.3 가변성 명세를 위한 항목

컴포넌트의 가변성은 컴포넌트의 재사용성과 확장성을 증가시킬 수 있는 장치이다. 가변성의 설계는 가변점과 그 가변점을 위한 가변치로 구성된다. 가변성의 종류는 속성 가변성, 로직 가변성, 워크플로우 가변성, 인터페이스 가변성, 영속성 가변성으로 분류한다. UML은 가변성을 표현하기 위한 장치를 가지고 있지 않다. 따라서 PIM 설계시 비표준 표기법을 사용하므로 MDA 도구에서 인식 할 수 없다. 가변성을 표현하기 위한 UML 프로파일은 표 4와 같다.

가변점은 표 4와 같이 5종류로 표현된다. 컴포넌트의 속성 가변치가 적용될 속성 가변점은 «VP-Attr»로 표현하고, «VP-Logic»은 로직 가변점을 묘사한다. 워크플로우 가변치가 적용될 워크플로우 가변점은 «VP-Workflow»로 표현하며, 인터페이스 가변점과 영속성 가변성은 각각 «VP-Interface»와 «VP-Persistency»로 표현한다.

가변치는 «Variant» 스테레오 타입을 사용한다. 가변성의 영역은 가변치가 모두 식별 가능하면, {vScope = "Close"} 꼬리표값을 사용하고, 가변치가 플러그 인 기법을 통해 컴포넌트 사용자에 의해 추가된다면, {vScope = "Open"} 꼬리표 값을 사용한다. {vpID = 값} 꼬리표값은 여러 가변점과 그 가변점에 채워질 가변치를 연결하기 위한 식별번호를 갖는다.

{varID = 값} 꼬리표값은 하나의 가변점이 여러 개의 가변치를 가지고 있으므로, 그 가변치를 식별하기 위한 식별번호이다. 가변치 식별 번호는 클래스, 순차도, 협력도등 모든 다이어그램에서 사용 가능하다. 예를 들면, 워크플로우 가변성을 갖는 메소드의 가변치를 표현하기 위해 해당 메소드의 내부 업무의 흐름을 여러 개의 순차도에 의해 표현 가능하다. 각각의 순차도에는 가변치 번호를 부여한다.

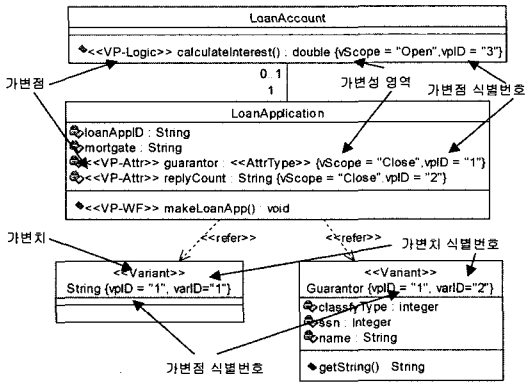


그림 7 가변성 명세의 예제

가변성을 CB-PIM에 적용한 예는 그림 7과 같다. LoanApplication 객체의 보증인(guarantor) 속성은 속성 가변성이며, 2개의 가변치를 가지고 있다. 하나는 보증인의 아이디만 문자열로 관리하는 가변치와 Guarantor 구조체로 관리되는 가변치가 있다. 이자 계산을 위한 Loan 컴포넌트의 calculateInterest( ) 메소드는 여러 종류의 이자계산 방식을 지원해야 하므로 Open 영역의 로직 가변성이다. 대출 신청서를 작성하기 위한 makeLoanApp( ) 메소드는 내부 워크플로우가 다르므로 워크플로우 가변점으로 설정된다.

5.4 배치(Deployment) 명세를 위한 항목

실행 가능한 단위인 컴포넌트는 기능적 요구 사항뿐만 아니라 컴포넌트가 미들웨어에 배치되고 운영되기 위한 요소가 필요하다. 컴포넌트 배치 정보의 명세는 올바른 통합과 사용환경을 보장하기 위해 명세 한다[28]. 컴포넌트를 운영하기 위한 실행, 배치, 트랜잭션, 보안등의 규칙들은 플랫폼마다 구현 방법이나 명세방법이 다르지만, 이러한 규칙들은 서로 다른 플랫폼의 컴포넌트에서도 지켜져야 한다.

표 4 가변성 명세를 위한 UML 프로파일 요소

요 소	명세 표기법	적용되는 UML 요소	비 고
가변점	«VP»	변수, 메소드	
속성 가변점	«VP-Attr»	변수, 유즈케이스	
로직 가변점	«VP Logic»	메소드, 유즈케이스	
워크플로우 가변점	«VP WF»	메소드, 유즈케이스	
인터페이스 가변점	«VP-Interface»	오퍼레이션	
영속성 가변점	«VP Persistency»	메소드	
가변치	«Variant»	클래스, 메소드	
가변성 영역	{vScope = 값}	메소드, 유즈케이스	Close, Open
가변점 식별 번호	{vpID = 값}	변수, 메소드	
가변치 식별 번호	{varID = 값}	클래스도, 순차도, 협력도, 활동도, 상태도	
제약조건	{ }, pre:, post:, inv:	클래스, 메소드, 연관관계	OCL
알고리즘	문자로 표기	메소드	OCL, ASL

컴포넌트 배치 명세에서 배치 속성은 컴포넌트를 컴포넌트 미들웨어에 설치하기 위한 정보를 정의한다. 실행환경 속성은 컴포넌트 미들웨어에서 컴포넌트 인스턴스가 실행될 때 갖는 특징 등을 정의한다. 트랜잭션 속성은 트랜잭션을 처리하는 방법을 설정하는 속성이다. 보안 속성은 컴포넌트를 사용하기 위한 보안 전략을 관리하는 속성이다. 컴포넌트 배치를 위한 항목은 표 5와 같다.

배치 환경을 명세하기 위해 «DeployProperty» 스테레오 타입을 사용한다. deployedName 속성은 컴포넌트 배치시 미들웨어에 의해 별칭으로 사용되기 이름을 정의한다. 컴포넌트가 서버에서 실행 될 때 컴포넌트 서버에서 다른 컴포넌트간의 정보 교환 메커니즘은 이 별칭을 사용하게 된다. 또한 artifactName 속성은 컴포넌트가 패키지 될 때 자동화 도구에 의해 자동으로 deployedName에 의해 패키지 된다.

실행 환경을 정의하기 위해 «RuntimeProperty» 스테레오 타입을 사용한다. 활동 정책에는 컴포넌트 또는 컴포넌트 인스턴스당 접근 클라이언트수의 한계, 컴포넌트당 인스턴스의 수, 인스턴스의 활성 시간(Activity time)의 제약, 비활성 종료시간(Inactivity timeout) 등을 명세 한다. 컴포넌트의 통합을 위해서는 컴포넌트의 기능적 특징뿐만 아니라 실제 클라이언트와 어플리케이션을 위한 아키텍처 실행 관점에서의 제약사항도 중요하다[29]. 따라서 어떻게 클라이언트가 컴포넌트를 사용하는지에 관한 명세를 위한 활동 정책(Activation Policy)이 필요하다.

트랜잭션 정책을 명세하기 위해 «TXProperty» 스테레오타입을 정의한다. 트랜잭션 정책을 명세하기 위한 속성 중 useTX가 false인 경우에는 트랜잭션에 관한 다른 속성은 모두 무시하고, 트랜잭션 메커니즘 구현 없이 컴포넌트 생성이 가능하다. TXAttrType은 TXAttrTypeKind 열거형 타입을 사용한다. 이 열거형 타입은 Enterprise Java Beans(EJB)에서 정의하고 있는 required, requiredsNew, supports, mandatory, notSupported, never 속성을 갖는다. 이 속성은 구현될 때 <trans-attribute> 속성에 사용된다. TXIsolation은 트랜잭션의 독립성 정책을 반영하고, 또한 트랜잭션 때문에 발생된 교착상태를 해결하기 위한 롤백 할 시간을 정의한다.

보안 정책을 정의하기 위해 «SecurityProperty» 스테레오 타입을 사용한다. userRoleName 속성은 해당 컴포넌트를 사용할 수 있는 사용자의 역할을 명세 한다. 일반적으로 Provided 인터페이스의 경우에는 모든 사용자의 역할에 사용을 허락하고, Customize 인터페이스의 경우에는 관리자 역할의 사용자에게만 허락할 것이다.

그림 8은 CB-PIM 수준으로 LoanMgr 컴포넌트의 실행 환경 및 조건을 명세한 예이다. 컴포넌트용 UML 프로파일에 의해 작성된 컴포넌트의 보안, 배치, 트랜잭션, 실행 환경 등의 정보는 UML Profile for EJB에 의해 XML 기반의 디플로이먼트 디스크립트의 <container-transaction>, <display-name>, <ejb-class>, <ejb-client-jar>, <ejb-entity-ref>, <ejb-jar id>, <ejb-name>, <method-permission>, <transaction-type>, <trans-attribute>, <use-caller-identity>, <security-role-ref>, <security-role> 속성을 자동 생성할 수 있다.

6. 평 가

기존 연구와 비교해 보면, Fontoura의 UML-F[16]는 객체지향 프레임워크에 기반을 두고, Exertier[17]는 컴포넌트 설계를 위해 시스템의 분리, 컴포넌트의 영역 설계, 컴포넌트의 내부 설계, 컴포넌트의 논리적 배치의 구성 항목을 제안했지만, 이에 관한 표현 방법은 제안하지 않았다. 본 논문은 컴포넌트 설계를 위한 4종류의 구성들을 모두 표현할 수 있는 UML 프로파일을 묘사했으며, 추가적으로 컴포넌트 가변성, 컴포넌트 실행을 위한 요소 등을 명세하였다.

본 논문에서 제시한 컴포넌트용 UML 프로파일은 컴

표 5 배치 명세를 위한 UML 프로파일 요소

요 소	명세 표기법	적용되는 UML 요소	비 고
배치 속성	«DeploymentProperty»	컴포넌트	
	deployedName, artifactName	컴포넌트	
실행 환경 속성	«RuntimeProperty»	컴포넌트	
	virtualClientsPerInstance,instancePerComponent, instanceTimeToLivecomponentTimeToLive,instanceInactivityTimeout	컴포넌트	
트랜잭션 속성	«TXProperty»	인터페이스, 컴포넌트,클래스, 메소드	
	useTX, TXAttrType,TXIsolation, TXTimeOut	인터페이스, 컴포넌트,클래스, 메소드	
보안 속성	«SecurityProperty»	인터페이스,컴포넌트, 클래스, 메소드	
	userRoleName	인터페이스,컴포넌트, 클래스, 메소드	

표 6 컴포넌트용 UML 프로파일의 CBD, MDA 기술의 지원 여부

○ : 지원

기 술항 목	CBD	MDA의 PIM	EDOC 프로파일	본 논문의 프로파일	비 고
컴포넌트 단위 명세	○		○	○	«SysComponent»외5종
Provided인터페이스	○	○	○	○	«ProvidedInterface»
Required 인터페이스	○			○	«RequiredInterface»
Customize 인터페이스	○			○	«CustomizeInterface»
가변점 명세	○			○	«VP Attr»외 5종
가변치 명세	○			○	«Variant»외 3종
컴포넌트 배치 명세	○			○	«TXProperty»외 3종
컴포넌트 워크플로우 명세	○	○	○	○	UML(순차도, 협력도)
모델 수준의 재사용		○	○	○	
컴포넌트 독립 모델 지원				○	
플랫폼 독립적 모델 지원		○	○	○	
모델간 변환 지원		○	○	○	

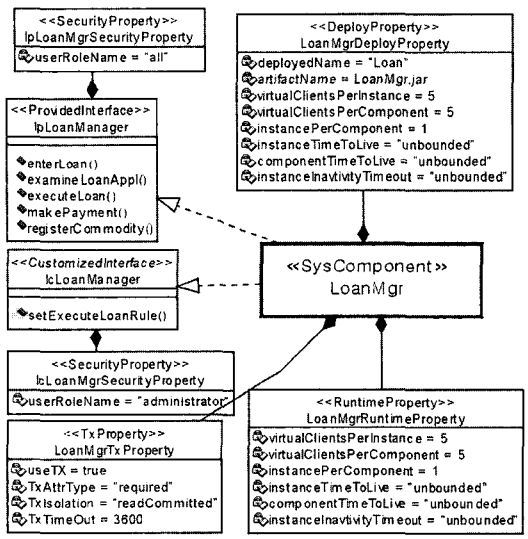


그림 8 컴포넌트의 실행 환경 및 조건 설계의 예

포넌트 기술과 MDA 기술의 PIM을 접목하여 컴포넌트용 PIM 작성이 가능하게 한다. 표 6과 같이 컴포넌트용 UML 프로파일을 CBD, MDA의 PIM 및 EDOC를 위한 UML 프로파일의 특징과 비교하여, 제안된 UML 프로파일이 컴포넌트를 PIM 수준으로 설계하는데 필요한 컴포넌트 내부 설계, 인터페이스, 가변성, 배치 명세 등의 CBD 기술을 모두 포함하여 PIM수준의 컴포넌트 설계가 가능하였다.

본 논문에서 제안한 UML 프로파일은 CBD를 MDA의 자동화 기술을 이용하여 효율적인 컴포넌트 생성이 가능하게 한다. 컴포넌트용으로 설계된 PIM은 어플리케이션에서 요구하는 플랫폼을 위한 PSM으로 자동 변환된 후 최종 컴포넌트를 자동 생산할 수 있게 하였다. 그러므로, 고품질의 컴포넌트를 자동 변환 기법에 의해 여러 플랫폼의 컴포넌트를 대량 생산 가능하게 하였다.

7. 결 론

CBD의 특징은 도메인 컴포넌트를 통해 큰 단위의 재사용이 가능하다. 또한, 여러 컴포넌트 사용자 마다 서로 다른 요구사항을 지원하기 위해 커스트마이제이션을 지원하고, 인터페이스와 구현을 분리함으로써 유지보수성을 증가시킬 수 있는 개발 방법이다.

MDA는 도메인을 분석하여 특정 어플리케이션 플랫폼 종속적이지 않은 PIM 설계를 만들다. 또한, PIM을 재사용하여 여러 어플리케이션 플랫폼 종속적인 PSM들을 설계하여 코드를 자동 생성하므로 생산성을 향상시킨다.

본 논문에서는 CBD와 MDA기술을 접목하기 위해 컴포넌트 정보의 재사용 가능한 CB-PIM을 제안하였고, CB-PIM에 컴포넌트 요소인 컴포넌트 내부 객체 및 컴포넌트간의 관계, 인터페이스, 가변성, 컴포넌트 배치관련 정보를 설계하기 위해 CBD용 UML 프로파일을 제안하였다. 제안된 CB-PIM과 컴포넌트용 UML 프로파일을 이용하여 컴포넌트를 자동으로 작성할 수 있다. 또한, 요구사항이 변경되어 플랫폼이 변경된다면, MDA 기술을 이용하여 CB-PIM의 수정 없이 CB-PIM을 PIM으로 자동 변환 후에 새로운 플랫폼의 컴포넌트를 자동화하여 생산할 수 있다. PIM 설계시 제안된 컴포넌트용 UML 프로파일을 적용함으로써, CBD와 MDA의 장점을 모두 사용하여 높은 개발 생산성, 이식성, 상호운용성, 및 유지보수성을 증가시킬 수 있다.

참 고 문 헌

[1] OMG, *MDA Guide Version 1.0.1*, omg/2003-06-01,

June 2003.
[2] Heineman, G. and Councill, W., *Component-Based Software Engineering*, Addison Wesley, 2001.
[3] Frankel, D., *Model Driven Architecture™: Applying MDA™ to Enterprise Computing*, Wiley, 2003.
[4] Flater, D., "Impact of Model-Driven Architecutre," *In Proceedings of the 35th Hawaii International Conference on System Sciences*, January 2002.
[5] Kleppe, A., Warmer, J. and Bast, W., *MDA Explained*, Addison-Wesley, 2003.
[6] JCP, *UML™ Profile For EJB_Draft*, Java Community Process. 2001.
[7] OMG, *UML™ Profile for CORBA Specification V1.0*, Nov. 2000.
[8] OMG, *UML Profile for EDOC V1.0*, http://www.omg.org/technology/documents/formal/edoc.htm, 2004.
[9] OMG, *UML Profile for Enterprise Collaboration Architecture (ECA) V1.0*, 2004.
[10] OMG, *Metamodel and UML Profile for Java and EJB V1.0*, 2004.
[11] OMG, *Flow Composition Model (FCM) V1.0*, 2004.
[12] OMG, *UML Profile for Patterns V1.0*, 2004.
[13] OMG, *UML Profile for ECA V1.0*, 2004.
[14] OMG, *UML Profile for Meta Object Facility V1.0*, 2004.
[15] OMG, *UML Profile for Relationships V1.0*, 2004.
[16] Fontoura, M., Pree, W., and Rumpe, B., "UML-F: A Modeling Language for Object-Oriented Frameworks," *Proceedings of 14th European Conference on Object Oriented Programming (ECOOP 2000), Lecture Notes in Computer Science 1850*, 2000.
[17] Exertier, D., Lnaglois, B., and Roux, X., "PIM Definition and Description," *Proceedings of 1st European Workshop, Model-Driven Architecture with Emphasis on Industrial Applications(MDA-IA 2004)*, 2004.
[18] 허진선, 김수동, "컴포넌트 참조 모델의 기술적 비교 평가," 한국정보과학회, 제31권, 제6호, 2004.
[19] Atkinson, C., et al., *Component-based Product Line Engineering with UML*, Addison-Wesley, 2001.
[20] Geyer, L. and Becker, M., "On the Influence of Variabilities on the Application-Engineering Process of a Product Family," *Proceedings of Software Product Line Conference (SPLC) 2002, Lecture Notes in Computer Science 2379*, 2002.
[21] Kim, S., Her, J., and Chang, S., "A Formal View of Variability in Component-Based Development," *Journal of Information and Software Technology (IST)*, To Appear, 2005.
[22] Choi, S., Chang, S., and Kim, S., "A Systematic Methodology for Developing Component Frameworks," *Proceedings of 7th Fundamental Approaches to Software Engineering (FASE'04) Conference, Lecture Notes in Computer Science 2984*, 2004.
[23] OMG, *Unified Modeling Language: Superstructure version 2.0*, ptc/03-08-02, 2003.
[24] Rumbaugh, J., Jacobson, I., and Booch, G., *The Unified Modeling Language Reference Manual, Second Edition*, Addison-Wesley, 2004.
[25] Cheesman, J. and Daniels, J., *UML Components*, Addison-Wesley, 2001.
[26] 민현기, 김수동, "EJB 기반의 효율적인 설계 패턴 및 엔터프라이즈 아키텍처 설계 기법", 한국정보과학회, 제30권, 제11호, 2003.
[27] Roman, E., *Mastering Enterprise JavaBeans™ and the Java™2 Platform*, Enterprise Edition, WILEY, 1999.
[28] Bachman, F. and Bass, L., *Volume II:Technical Concepts of Component-Based Software Engineering*, CMU/SEI-2000-TR-008, May 2000.
[29] Deprince, W. and Hofmeister, C., "Usage Polices for Components," *Proceedings of the 6th ICSE Workshop on CBSE*, 2003.

민 현 기

1999년 건양대학교 전자계산학과 공학사 2001년 숭실대학교 컴퓨터학과 공학석사 2005년 숭실대학교 컴퓨터학과 공학박사 2005년 3월~현재 숭실대학교 정보과학대학 전임연구원. 관심분야는 컴포넌트 기반 개발(CBD), 제품계열 공학(PLE), 모델 기반 아키텍처(MDA)

김 수 동

1984년 Northeast Missouri State University 전산학 학사. 1988년/1991년 The University of Iowa 전산학 석사/박사. 1991년~1993년 한국통신 연구개발단 선임연구원. 1994년~1995년 현대전자 소프트웨어연구소 책임연구원. 1995년 9월~현재 숭실대학교 컴퓨터학부 부교수. 관심분야는 객체지향 S/W공학, 컴포넌트 기반 개발(CBD), 제품계열 공학(PLE), 모델 기반 아키텍처(MDA)