DOI: 10.3745/KIPSTD.2009.16-D.3.381

2+1 View 통합 메타모델 기반 PIM/PSM 컴포넌트 모델링 기법

송 치 양[†] · 조 은 숙[††]

요 약

소프트웨어 모델링 과정에서 생성되는 산출물들에 대한 모델의 재사용성을 향상시키기 위한 기법으로, MDA 모델 기반의 개발방법이 적용되고 있다. 아직도, UML을 이용한 MDA 기반의 개발에 대한 계층적이고 체계적인 기법은 미약한 상태이다. 이로 인해, MDA 메타모델에 기반한 일관성 있고 재사용성이 높은 MDA 모델링이 제대로 실현되지 못하고 있다. 본 논문은 이를 해결하기 위해, 2+1 View 통합 메타모델을 통한 MDA(PIM/PSM) 컴포넌트 모델링 기법을 제시한다. 먼저, 개발 프로세스 view와 MVC View를 표현할 수 있는 meta-architecture view 모델을 정의한다. 정의된 meta-architecture view 계층의 메타 레벨에서, 개발 프로세스 view와 MVC view별로 MDA 기반의 계층적 통합 메타모델이 제시된다. 이 메타모델은 UML 모델과 GUI 모델이 갖는 모델링 요소들을 PIM과 PSM에 의해 계층적으로 표현한다. 제시한 메타모델을 ISMS 시스템에 적용하여 MDA 기반의 컴포넌트 모델링 사례를 제시한다. 이를 통해, 개발단계별 및 MVC 방식에 의해 일관성 있고 계층적인 MDA 컴포넌트 모델을 구축할 수 있다. 따라서, 모델에 대한 독립성과 재사용성의 향상을 기대할 수 있다.

키워드 : MDA, PIM/PSM, MVC, 메타모델, UML

A PIM/PSM Component Modeling Technique Based on 2+1 View Integrated Metamodel

CheeYang Song[†] · EunSook Cho[††]

ABSTRACT

As a technique to enhance reusability for the created artifacts in software modeling process, the model driven method such like MDA has been applied. Unfortunately, the hierarchical and systematic MDA based development technique using UML is poor yet. This causes the problem that the MDA modeling with high consistency and reusability based on MDA metamodel is not being realized. To solve this problem, this paper proposes a MDA (PIM/PSM) component modeling technique using 2+1 view integrated metamodel. At first, the meta-architecture view model which can represents development process view and MVC view is defined. Then, the hierarchical integrated metamodels per view are addressed separately for modeling process and MVC at metamodel level on the hierarchy of the defined meta-architecture view model. These metamodels are defined hierarchically by layering the modeling elements in PIM and PSM pattern for UML models and GUI models. Appling the proposed metamodel to an ISMS application system, it is shown as a component modeling case study based on MDA. Through this approach, we are able to perform a component model with consistency and hierarchy corresponding to development process and MVC way. Accordingly, this may improve more independence and reusability of model.

Keywords : MDA, PIM/PSM, MVC, Metamodel, UML

1. 서 론

서비스들간의 융합화에 힘입어 소프트웨어는 복잡화, 대형화되어가고 그 수요가 폭주하는 실정이다. 이를 대처하기 위한 소프트웨어 생산기술로 모델 중심의 모듈성과 독립성을 갖춘 컴포넌트 구축을 통한 재사용에 대한 연구가 활발하다[1]. 이러한 모델링 방법이 MDA(Model Driven Architecture)[2] 개발방식이다. MDA는 구현 플랫폼의 모델 반영여부 등 관심의 분리(separation of concern)의 원칙에 입각하여, 개발 프로세스(단계)에 따라 PIM(Platform Independent Model)[3] 모델과 PSM(Platform Specific Model) 모델 방식으로 분리된 모델의 생성을 지향한다. 또한, 어플리케이션 모델링은 3개 영역계층의 MVC(Model View Controller)[4]

※ 이 논문은 경북대학교(지원 2009년도) 학술연구지원금에 의해 연구되었음.
† 정 회 원 : 경북대학교 소프트웨어공학과 조교수
†† 정 회 원 : 서일대학 소프트웨어과 조교수
논문접수 : 2009년 2월 19일
수 정 일 : 1차 2009년 3월 27일
심사완료 : 2009년 4월 15일

디자인 패턴에 의해 독립된 모델화를 통해서MDA 개발을 할 수 있다. 그래서, 이러한 개발 프로세스와 MVC가 결합된 MDA 기반의 메타모델 사용한 MDA 컴포넌트 모델링 방법이 필요하다.

UML(Unified Modeling Language)[5]은 4+1 아키텍쳐 view에 의해 각 모델별 모델링 요소로서 메타모델을 정의하고 공통 모델링 요소로 그룹화하여 명세하고 있다. UML은 프로세스 개념을 갖고 있지 않는 모델링을 위한 표현언어로서, 개발 과정에서 필요한 모델들과 그 모델링 요소들을 정의하고 있다. 한편, UML의 클래스 모델에서 스테레오타입을 통해서 MVC 방식으로 표현할 수 있다. [6]에서 MVC 기반의 PIM 및 PSM 패턴방식에 의한 적용사례를 보여주고 있다. 그러나, UML 모델의 모델링 요소를 기반으로 정의된 메타모델을 제공하고 있지 않으며, PIM/PSM 모델화 접근이 명확하지 않다. 또한, [7]에서, UML을 사용한 메타모델 기반의 계층적 모델링 방법을 제시하였다. 그러나, MVC 디자인 패턴이 반영되지 않았고, MDA 기반의 컴포넌트 모델링을 지원하지 않는다. [8]에서 UML의 컴포넌트 모델을 이용하여 MDA/PIM으로 명세하기 위한 메타모델 및 컴포넌트 명세를 위한 UML 프로파일을 제시했다. 이것은 UML의 컴포넌트 모델을 사용한 PIM 모델의 명세에 대해서만 보여주고 있다. 결국, UML을 이용한 MDA 기반의 개발에 대한 계층적이고 체계적인 기법이 미약한 실정이다. 그래서, UML 모델을 사용해서 프로세스 관점과MVC 관점에 의한 통합 메타모델 기반의 체계화된 모델링 기법이 필요하다.

본 논문은 UML 모델을 대상으로 개발 프로세스 및 MVC 패턴에 의한 2+1 View 메타모델을 통한 MDA(PIM/PSM) 컴포넌트 모델링 기법을 제시한다. UML은 개발과정에서 필요한 모델에 대한 무엇(“what”)을 정의한 반면, 본 논문은 UML을 어떻게(‘how”) MDA 기반 개발 프로세스에서 적용할 것인가에 대해 확장된 메타모델을 통해 해결하는 기법을 제시한다. 기본 개념은 정형화된 메타모델에 프로세스 개념을 부여하는 것이다. 즉, UML을 사용하여 MDA 기반으로 소프트웨어를 모델링할 때, 개발의 진행단계에 따른 시간적인 개발 프로세스 관점과 구조적인 어플리케이션의 아키텍쳐 관점(MVC)에서 접근하는 것이다. 개발 프로세스 관점은 요구분석, 기본 설계(PIM), 상세 설계(PSM) 및 구현의 개발단계에 따라 모델링하는 것이다. 기본설계 단계에서 PIM 모델을 생성하고, 상세설계 단계에서 PSM 모델을 계층적으로 구축함에 의해서 MDA를 지향한다. 한편, 어플리케이션의 아키텍쳐 관점은 MVC패턴에 의해 시스템을 GUI(Graphic User Interface Model)/ 비즈니스 로직/ DB(Data Base) 모델로 분리하여 모델링을 수행하는 것이다. 또한 이 모델들은 개발 프로세스의 상세화 수준에 따라 PIM 모델과 PSM 모델들로 각각 모델링을 통해 MDA로 접근한다. 따라서, MDA 개발방식에 따라 PIM 모델과 PSM 모델별로 컴포넌트 모델이 설계된다. 그래서, 독립화된 PIM 및 PSM 패턴의 컴포넌트 모델을 생성함으로서 모델링 산출물의 재사용성을 제고시킬 수 있다. 이를 제공하기 위해, 본 논문은 개발 프로세스 View와 MVC View를 통합하는 2+1 meta-architecture view 모델을 정의한다. 정의된 meta-architecture view 모델의 메타레벨에서 개발 프로세스와 MVC 개발방식을 위한 UML 기반의 계층적 통합 메타모델을 정의한다. 즉, 개발단계별과 MVC별로 MDA 개발방식에 따라 PIM 모델과 PSM 모델을 생성하기 위해, UML 모델의 모델링 요소들로 구성된 계층적 메타모델을 정의한다. 적용사례로서, ISMS(Internet Shopping Mall System) 시스템을 대상으로 개발의 진척 수준에 따라 정의된 메타모델내 모델링 요소들을 사용해서 MDA 기반의 컴포넌트 모델링을 보인다. 이를 통해, 개발단계별 및 MVC 방식에 의한 체계성 있는 컴포넌트 모델링을 수행할 수 있으며, 모델의 독립성과 재사용성을 향상시킬 수 있을 것이다. 추후, 모델간 매핑(mapping) 규칙이 정의된다면, CASE Tool을 통한 모델들간의 consistency checking이 가능할 것이다.

본 논문의 구성은 다음과 같다. 2장에서 UML의 메타모델과 메타모델 기반의 모델링 프로세스에 대한 기존 연구들을 살펴본다. 3장에서는 2+1 view 모델의 접근 원리 및 UML 모델 사용의 계층적 피라미드 구조를 기술한다. 4장은 계층적 피라미드 구조에 기반하여 MDA 지향의 개발 프로세스 view와 MVC view에 의한 메타모델을 제시한다. 5장은 제시 메타모델의 모델링 요소를 사용해서 ISMS 응용시스템을 대상으로 적용사례를 보인다. 6장은 기존 연구와 대비하여 정량적 및 비 정량적 비교평가를 기술하고 7장에서 결론을 맺는다.

2. 관련 연구

MDA 기반의 모델링을 위한 PIM/PSM과 MVC 접근의 현존 연구에 대해 UML의 메타모델과 이 메타모델에 기반한 모델링 프로세스에 초점을 두고 분석한다.

2.1 UML 메타모델

UML[5]은 그 구성 모델들에 대해 4+1 view 아키텍쳐에 근간을 두고 구조적인 아키텍쳐로서 정의하고 있다. 즉, UML을 구성하는 개별 모델들을 각각 메타모델로 정의하고 이들을 그룹화한 메타모델로 명세하고 있다. UML은 본질적으로 모델링의 표현언어로서 모델이 무엇(“what”)인가를 정의하고 있어 어떻게(“how”) 개발과정에 적용되는가에 대한 프로세스 개념을 갖고 있지 않다. 따라서, UML의 메타모델은 개발의 프로세스나 MVC 패턴에 의한 모델링 요소를 계층화하여 정의하고 있지 않다. 본 논문에서 MDA 기반 개발기법은 UML을 사용하여 모델 중심으로 개발 모델의 상세화 수준별(PIM/PSM) 그리고 어플리케이션의 아키텍쳐 패턴별(MVC)로 모델링을 수행하는 것으로 정의한다. 즉, UML의 어떤 동일 모델에 대해 PIM/PSM 및 MVC 형태로 모델이 독립화되어 산출물을 생성하는 것을 의미한다. 따라서, 기존 UML 메타모델을 이용해서 MDA 기반의 모델링을 수행할 수 없기에 확장된 메타모델이 필요하다. 본 논문에서는 UML 메타모델에 프로세스 개념과 MVC 개념을 메타모델에 표현하고

이 메타모델에 기반해서 MDA 기반 모델링이 이루어지도록 하는 것이다.

2.2 메타모델 기반 MDA 모델링 프로세스

MDA 기반의 모델링은 UML 모델들에 대해 PIM/PSM과 MVC 패턴에 의해 메타모델이 정의되어 있고 계층적 패턴에 의해 정의된 메타모델의 모델링 요소들을 사용한 기법이 요구된다. 대표적 상용 방법론인 UP(Unified Process) 방법론[9]은 전형적인 UML에 기반을 둔 계층적인 모델링 방법이다. 그러나, 기존 UML 메타모델이 MDA 기반으로 수행되는 메타모델을 제공하지 않으므로서 PIM/PSM 및 MVC에 의한 독립 패턴화된 모델링이 어렵다. [6]에서 MVC 패턴방식에 의한 실제적인 모델링 적용사례를 보여주고 있다. 이 방법은 스테레오타입을 사용해 MVC 패턴을 표현하고, PIM/PSM의 계층적 모델링을 지원한다. 그러나, UML 모델들에 대한 정형화된 메타모델이 정의되어 있지 않고 MVC와 PIM/PSM의 매끄러운 결합 모델링이 구체적이고 명확하지 않다. [7]에서, UML 모델들을 가지고 객체 수준에서 개념적/ 명세적/ 상세적 모델링 단계에 의한 계층적 메타모델과 이에 기반한 모델링 프로세스를 제시했다. 그러나, 객체 수준의 모델링을 지원하는 메타모델로서, 재사용성이 높은 컴포넌트 수준의 모델링을 지원하는 메타모델을 제시하고 있지 않다. 따라서 이것은 컴포넌트 모델링 기법을 지원하지 않는다. 또한, PIM/PSM과 MVC의 패러다임에 의한 메타모델과 모델링 프로세스를 제공하지 않으며, UML Specification V2.0을 수용하지 않는다. 한편, [8]에서 MDA/PIM으로 명세하기 위해 UML의 컴포넌트 모델에 대해 메타모델과 UML 프로파일을 제시하고, 이에 통한 컴포넌트 명세 기법을 제안했다. 이것은 UML의 컴포넌트 모델에 대해 PIM 모델 수준의 명세에 대해서만 보여주고 있다. 그래서, 시스템의 정적 및 행위 모델링에 필요한 다른 UML 모델들인 클래스 모델, 상호작용 모델, 활동 모델 등을 포함하고 있지 않다. 따라서, UML 모델들을 가지고 개발 프로세스 및 MVC 관점별로 구분하여 정의된 통합 메타모델 기반의 모델링 기법을 지원하지 않는다.

결국, 개발 프로세스와 MVC 패턴의 개념이 포함된 MDA 기반 개발방법을 제공하기 위한 정형화된 메타모델의 부재에 기인하여 개발 단계별 접근이 용이치 않고, 모듈성이 높은 모델이 생성되지 않으므로서, 모델링 산출물의 재사용성을 저하시킨다.

3. 2+1 View 기반 MDA 접근 모델

본 장에서는 개발 프로세스와 어플리케이션 3 계층에 의한 2+1 View 메타 아키텍쳐의 전체적인 구조를 살펴보고 이에 기반한 각 View별 피라미드 구조를 기술한다.

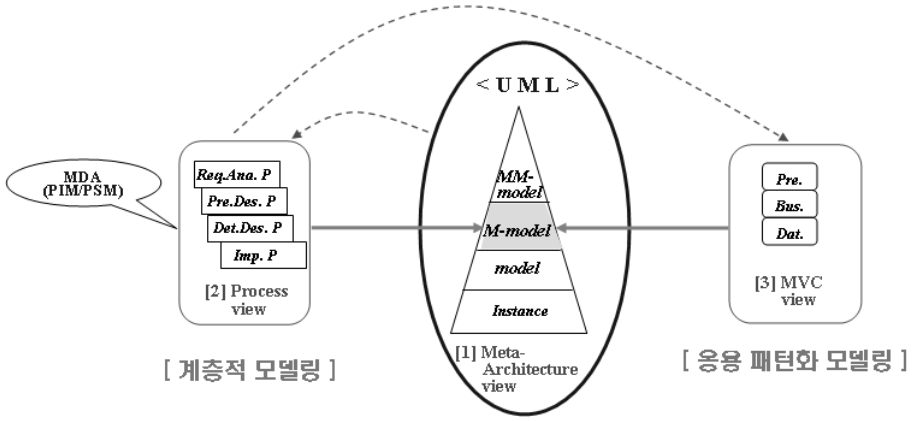
3.1 2+1 View MDA 접근 모형

UML은 모델링 언어로서 어플리케이션의 개발에 필요한 모델들이 무엇인가를 정의하고 있다. 그래서, 어떻게 개발 프로세스에 따라 UML 모델이 적용되는지 제공하고 있지 않다. 본 논문의 목적은 UML을 MDA(PIM/PSM) 기반의 개발에 어떻게 적용할 것인가를 해결하기 위해 메타모델을 통한 접근을 제시하는 것이다. 이것은 UML 모델들을 대상으로 개발단계에 의한 프로세스와 어플리케이션의 아키텍쳐 패턴인 MVC를 반영해서 계층화된 메타모델을 통해서 해결한다. 이를 통해, 구축 모델간의 일관성 있는 생성 및 구축 모델의 재사용성의 제고하고자 한다. 제시 방법의 설계 원리는 다음과 같다.

- **2+1 view에 의한 계층적 통합 메타모델 정의**
  - MDA 기반 2+1 View meta-architecture 모델 정의
    - meta-architecture view : MDA 모델 구축을 위한 인프라 아키텍쳐 정의
      meta-architecture view에 의한 4계층 구조(MM (Meta-Meta)-model, M(Meta)-model 등)상에서 메타레벨에서 정의되는 process view와 MVC view 정의
    - process view : 개발 프로세스상의 단계별 구축 대상 모델 및 모델링 요소를 정의(UML 모델 대상)
    - MVC view : 어플리케이션 아키텍처의 3계층 MVC 패턴별 구축 대상 모델 및 모델링 요소를 정의 (UML 모델 및 GUI 모델 대상)
  - 각 view별 피라미드 구조와 계층적 통합 메타모델 정의
    - 피라미드 구조는 모델 대상의 뷰(view)별 계층화 구성
    - 계층적 통합 메타모델은 피라미드 구조에 뷰별로 정의한 각 모델들이 갖는 모델링 요소들로 계층화 정의 (모델 및 모델링 요소의 계층화를 통한 메타모델내 프로세스 개념 주입)

- **정형적 메타모델 사용의 MDA 컴포넌트 모델링**
  - Process view 기반 PIM 및 PSM 컴포넌트 모델 구축
    Process view 기반 계층적 통합 메타모델에서 정의한 개발 단계별 모델링 요소를 사용하여 PIM과 PSM 컴포넌트 모델 구축
  - MVC view 기반 MVC 컴포넌트 모델 구축
    MVC view 기반 계층적 통합 메타모델에서 정의한 MVC별 모델링 요소를 사용하여 MVC 컴포넌트 모델 구축

이를 구축하기 위한 2+1 view 기반 MDA 접근 모델의 원칙 및 특징은 (그림 1)에서 보여주듯이, MDA 기반의 ***PIM/PSM*** 모델의 계층적 접근과 ***MVC*** 디자인 패턴을 적용한다. 개념적 측면에서, 모델 기반으로 MDA와 MVC를 사용하고, ***메타모델***을 사용하여 구축 대상 모델 및 모델링 요소를 정의하고, 이 메타모델 구성시 모델링 요소를 개발



(그림 1) 2+1 view meta-architecture 모델

단계에 따라 계층화하여 ***프로세스*** 개념을 부여한다. 여기서 메타모델은 UML 모델과 GUI 모델의 추상적 구문(abstract syntax)을 정의하기 위해 사용한다. 또한, 모델링 성숙도(모델명세의 표현 혹은 묘사 정도) 기반으로 ***계층성***(Hierarchy)과 추상화(Abstraction)의 개념을 적용한다. 계층성은 모델 및 모델링 요소들이 개발단계 및 MVC에 의해 계층 구조로 조직화하는데 사용한다. 계층화를 통해 수준별 모델의 명세가 가능하게 되어, 생성된 모델의 재사용성을 높일 수 있다. 또한, 개발단계별 및 MVC별 모델과 모델링 요소의 식별과 적용이 용이해진다.

(그림 1)에서, 2+1 view meta-architecture 접근 모델은 한 개의 meta-architecture view와 두개의 metamodel modeling view로 구성한다. 전자인 meta-architecture view는 메타 아키텍쳐 차원의 계층적 인프라 topology를 표현한다. 후자는 개발 단계를 나타내는 process view와 어플리케이션 아키텍쳐를 의미하는 MVC view로 구성된다. 이 두 개 view별로 4장에서 통합 메타모델로 정의된다. 통합 메타모델은 process view와 MVC view에 의해 구축에 필요한 UML 모델/GUI 모델 그리고 이 모델이 가진 모델링 요소를 가지고 정의한다. Process view는 meta-architecture view의 M2 레벨에서 정의되며 S/W 생명주기에 의한 단계적 개발 관점이다. MVC view는 meta-architecture view의 M2 레벨에서 정의되며 MVC 패턴에 대한 architecture framework의 관점이다. 이후, 2+1 view를 구성하는 meta-architecture view, process view 및 MVC view에 대해 각각 계층적 메타 피라미드 구조와 메타모델을 구축한다[10]. (그림 1)에서 Req.Ana. P는 requirement analysis phase(요구분석 단계), Pre.Des. P는 preliminary design phase(기본설계 단계), Det.Des. P는 detail design phase(상세설계 단계), Imp. P는 implementation phase (구현 단계), pre.는 presentation(GUI 표현 계층), Bus.는 business(비즈니스 로직 계층), Dat.는 data(데이터 계층)를 각각 나타낸다.

한편, 메타모델에 계층적인 프로세스 개념을 제공하기 위해, 피라미드 구조상에 계층화되는 ***모델***과 계층적 통합 메타모델상에 계층화되는 ***모델링 요소***에 대한 분할 기준 및 그들간 관계의 생성 규칙이 필요하다. 예로, 개발 프로세스 뷰에 의한 메타모델의 구축은 개발 생명주기의 구체화에 따라 일치하는 확장적 모델링 요소들이 추가됨에 의해 계층적으로 분할되어 구성되어야 한다. 이때, 어떤 특정 단계에서 사용되는 모델 구성 및 그 모델링 요소들은 이 단계에 부합토록 모델링 수준, 모델의 용도, 모델 및 모델링 요소들간의 입출력 관계 등을 고려해서 분할 기준이 정립되어야 한다.

먼저, 모델(UML 모델 및 GUI 모델)과 모델링 요소의 계층화를 위한 분할 기준은 다음과 같다.

- 모델 및 이 모델의 모델링 요소 자체가 지닌 본질적 용도 또는 추상화 수준의 적용
  예로, 프로세스 관점에서class model의 경우, 이 모델의 모델링 작업은 구현 플랫폼의 반영여부 및 모델 명세의 상세화에 따라 기본설계 단계와 상세설계 단계에 거쳐서 점진적으로 구축한다. 그래서 모델 대상의 피라미드 구성시, 이 두 단계에 걸쳐 class 모델을 배치((그림 3)에서 Class_M과 C# Class_M/Java Class_M) 한다. 메타모델 구축은 피라미드 구조상의 각 단계내 포함된 모델들을 대상으로 각 모델별 필수적이고 범용적인 모델링 요소들로 계층화하여 정의한다.
- 개발 프로세스에 따라 모델들간 및 모델링 요소들간의 모델링 진행 수준 동일화
  모델링 작업이 UML/GUI 모델간 그리고 UML 모델들간에 연계 모델링이 이루어져야 하므로, 이들과의 상호 산출물 입출력 관계, 모델링 대상의 크기(granularity) 그리고 전체적 추상화 수준에 부합한 모델 및 모델링 요소들로 구성한다.
- MVC 패턴을 모델링하기 위해 적합한 모델 및 모델링 요소의 구성 및 분할
  예로, “V” 패턴을 모델링하기 위해 GUI 모델이 피라미드 구조에 필요하고, 구현환경의 반영에 따라 메타모델 정의시 모델링 요소를 PIM/PSM에 의해 분할한다.

실제적으로 다양한 모델들간 그리고 모델링 요소들간에

상호관계를 갖고 있다. 따라서, 일관성 있고 체계성이 있는 모델들간/모델링 요소들간의 관계들을 추출하기 위해 공통적 적용의 생성 규칙이 필요한데, [7]에 제시한 규칙에 준거하여 생성한다. 이러한 계층적 분할 기준과 메타모델의 생성 규칙에 따라, 피라미드 구조와 메타모델을 정의한다.

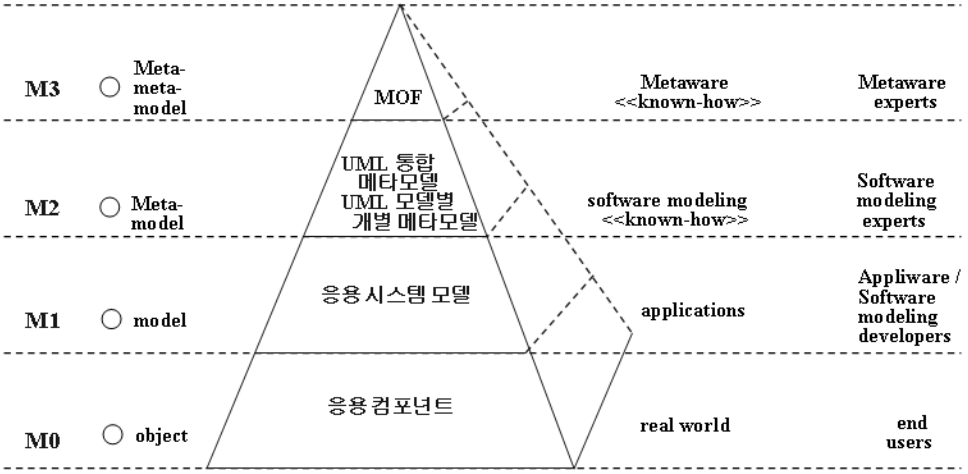
3.2 2+1 View별 피라미드 구조

2+1 view meta-architecture모델을 구성하기 위한 근간이 되는 meta-architecture view에 대한 메타 피라미드는 [10]에서 제시한 것을 적용, 4계층 아키텍처로 구성하여 (그림 2)에서 보여준다. 각 view별 피라미드 구조가 필요한 이유는 개발 단계별 및 MVC별로 계층화하여 PIM과 PSM 모델을 생성하기 위함이다. 즉, 모델링 산출물의 추상화 수준에 따라 피라미드 계층 구조에 부합토록 UML 모델 혹은 GUI 모델을 프레임워크 형태로 정의할 수 있기 때문이다. 이를 통해, 피라미드 구조상에 계층별로 포함된 모델들을 대상으로 4장에서 뷰 별로 모델의 모델링 요소를 가지고 계층적 메타모델을 구축할 수 있다.

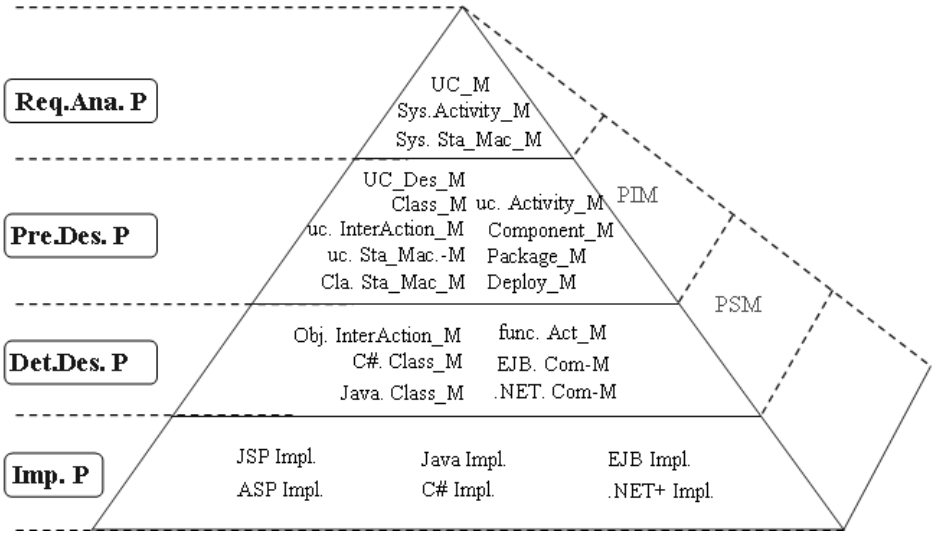
(그림 2)에서, Meta architecture view는 독립적이고 이질적인 어플리케이션 및 컴포넌트간의 상호작용 아키텍처를 표현한다. M1 레벨은 프로그램들이 실행되는 레벨이며, appliware는 특정 도메인 종속된 응용을 나타낸다. M2는 어플리케이션의 생산을 관리하기 위해 사용하며, metaware는 특정 도메인에 독립적임을 의미한다. M3는 메타 모델들이 어떻게 기술되어야 하고 관리되어야 하는지를 정의한다. 메타모델 구성요소의 예로서, M1 레벨에 속하는 개체는 응용시스템별로 UML의 개별 모델들이 설계와 구현 플랫폼에 의해 생성하는 것들로 구성한다.

다음으로 process view (즉, development phase view)는 MDA 기반의 개발 단계(프로세스)에 의해 생성되는 UML 모델들을 대상으로 피라미드 구조를 정의한다.

(그림 3)에 보여주듯이, 개발 단계는 요구사항의 분석 및 정의를 수행하는 requirement analysis(요구분석) 단계, 시스템의 PIM 모델(구현 플랫폼 미반영)을 생성하는 preliminary design(기본 설계)단계, 시스템의 PSM 모델(구현 플랫폼 반영)을 구축하는 detailed design(상세설계) 단계, 그리고 시



(그림 2) Meta-architecture view의 메타 피라미드 구조

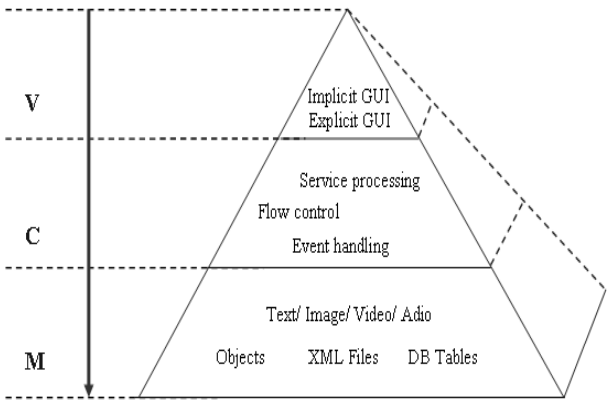


(그림 3) Process view의 피라미드 구조

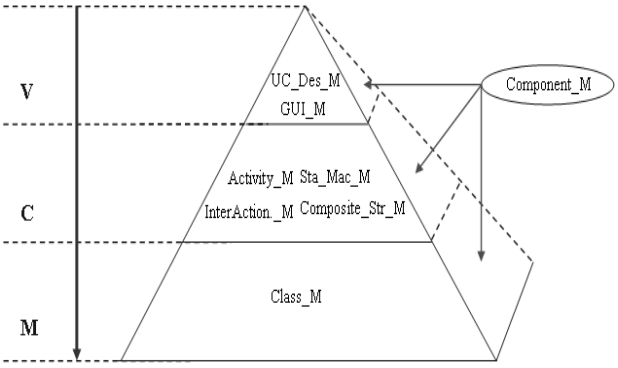
스템을 구현하는 implementation(구현) 단계로 구성한다. 각 개발 단계는 모델링의 추상화 수준에 따라서 구축해야 하는 UML 모델들을 가지고 process view의 피라미드 구조를 정의한다. 각 개발단계에 의해 피라미드상에 포함된 개체들(모델)은 단원 4.2에서 process view 기반 통합 메타모델을 정의하기 위한 기반 대상모델이 된다. 예로, 상세설계(Det.Des.P) 단계에서 구축해야 할 모델들은 Obj. Interaction_M(Object 수준의 Interaction Model)외 5개 UML 모델들이 되는 것이다. 여기서, 개발단계에 따른 계층적 모델링을 위해, UML 모델중 어떤 모델은 각 개발단계에 한번만 포함되는 경우와 여러 단계에 속한 경우가 있다. 동일 모델이 다수 개발단계에 포함된 경우, 개발 단계에 따라 상이한 시스템의 크기를 대상으로 동일한 표기법을 가지고 모델링한다. 가령, activity model의 경우, 응용의 모델링 대상 크기에 따라 다양하게 적용할 수 있는데, 본 논문에서는 요구분석/ 기본설계/ 상세설계의 개발 단계에 걸쳐서, 각각 "시스템" 크기를 대상으로 모델링하는 "Sys. Activity_M", 유스케이스 크기 수준의 "uc. Activity_M", 그리고 독립적 단위(기능) 모듈 수준 크기의 "func. Activity_M"로 개체들을 구성, 구축한다. (그림 3)에서, Req.Ana. P, Pre.Des. P 등은 (그림 1)에 나타낸 약어의 설명과 동일하다.

이어서, MVC view는 개발 단계에 따라 3 계층 MVC 디자인 패턴에 의한 관점을 의미한다. MVC view에 의한 피라미드 구조는 묘사의 내용에 의한 추상적 피라미드와 UML 모델들로 구성되는 상세적 피라미드로 구성한다. 먼저, 추상적 피라미드 구조는 (그림 4)에서 보여준다. (그림 4)에서 MVC 아키텍쳐 패턴의 계층 구조는 presentation view, business view 및 data view[1, 6]로 구성한다. 패턴별 구성 개체에 대한 예로서, "V" 계층을 구성하는 개체들로 implicit GUI(Window, pop-up 등)와 explicit GUI(Table, graph 등)가 있다[11].

다음으로, 상세적 피라미드 구조는 (그림 5)에 나타내었다. 특별히, 상세적 피라미드 구조를 구성하는 모델로서 UML 모델 외에 "V" 모델링을 위해 GUI 모델이 필요하기에 이를 포함한다. (그림 5)에서, "Component_M"은 MVC의 각 계층에 연관하고 있다. 이것은 UML 모델 및 GUI 모델을 사용



(그림 4) MVC view의 추상적 피라미드 구조



(그림 5) MVC view의 상세적 피라미드 구조

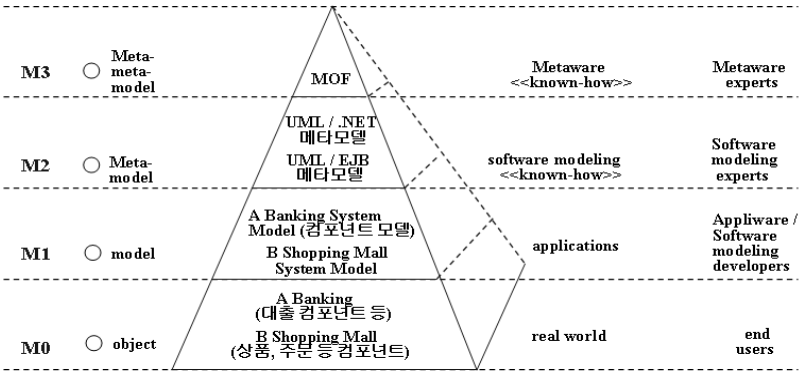
해서 M, V 및 C별로 컴포넌트 모델을 구축하는 것을 나타낸다. 예로서, "V"컴포넌트 모델의 생성과정은 다음과 같다. 먼저, 유즈케이스 명세서에 GUI 스케치를 기술한다. 초기 스케치를 통해 생성된 초기 화면들을 중심으로 GUI 클래스들을 추출하여 PIM 수준의 GUI 클래스 모델을 생성한다. 다음으로, 상호 연관된 GUI 클래스들을 모아서 GUI 컴포넌트 모델로 생성한다. 유사하게, "M"컴포넌트 모델과 "C" 컴포넌트 모델의 생성과정도 해당 UML 모델을 사용하여 구축할 수 있다((그림 8) 참조). UML 모델중에 패키지 모델 및 배치모델 등은 process view 기반 피라미드 구조에서 포함하고 있고 복잡성을 줄이기 위해 포함하지 않았다.

4. 2+1 View 기반 계층적 통합 메타모델

제 3장에서 2+1 view 기반의 각 view별 피라미드 구조를 정의하였고 본 장에서는 피라미드 구조에 정의한 모델들을 가지고 계층적 통합 메타모델을 구축한다. 즉, 각 view별 피라미드 구조상에 정의된 계층적 UML 모델과 GUI 모델을 대상으로 각 모델이 포함하고 있는 모델링 요소들을 가지고 이들을 계층화하여 PIM/PSM와 MVC 패턴에 의해 상호 관계를 정립함으로서 통합 메타모델을 정의한다. 이렇게 정의된 통합 메타모델의 모델링 요소를 가지고 개발 프로세스에 따라 계층적으로 모델링을 수행할 수 있으므로 이것이 메타모델에 프로세스 개념이 주입된 것이다.

4.1 Meta-architecture view 기반 메타모델

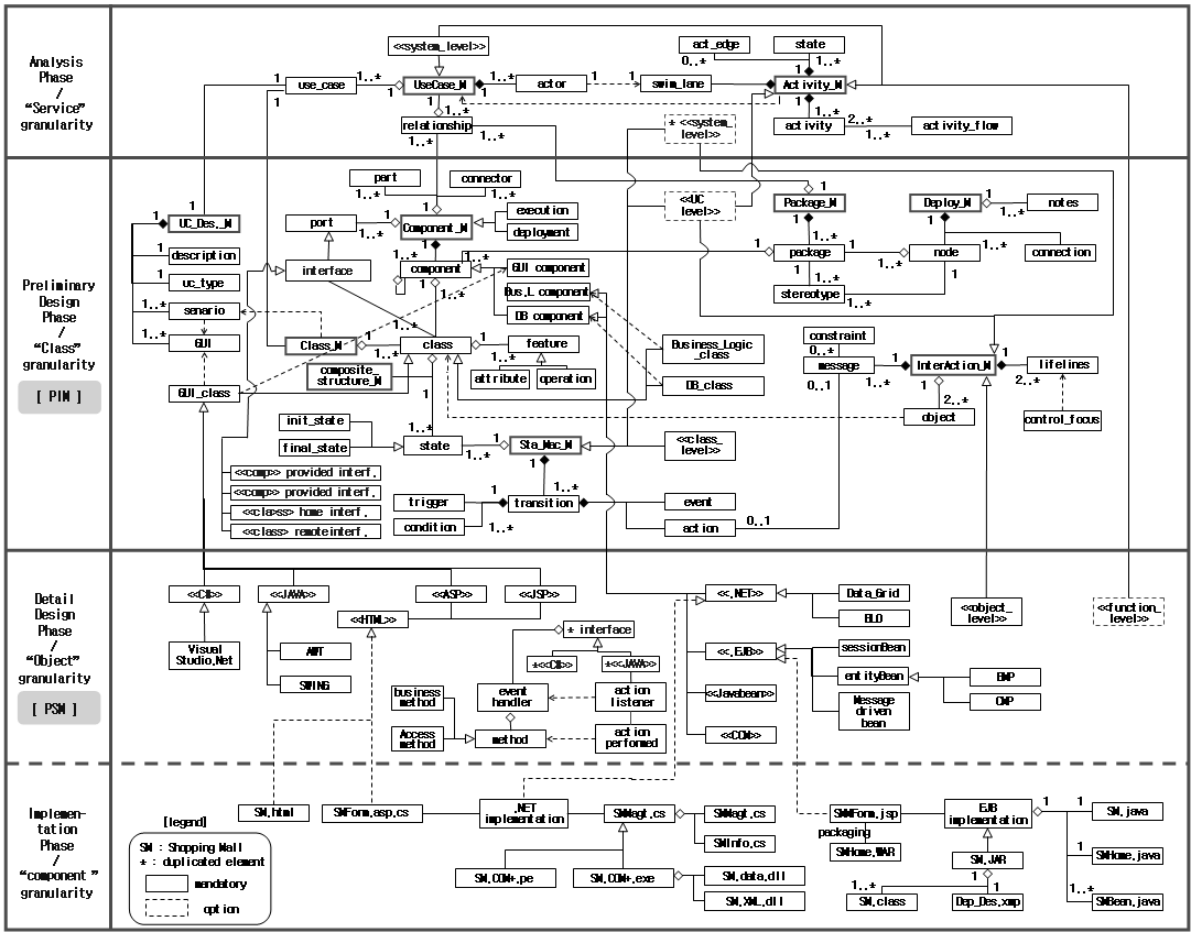
meta-architecture view 기반 피라미드 구조인 (그림 2)에 대한 인스턴스로서 meta-architecture view에 대한 메타모델을 정의한 것이 (그림 6)이다. 이 view에 의한 메타모델은 메타레벨 수준에서의 메타모델을 정의되므로 다른 view 별 메타모델 구성과 달리, UML 모델이 개체의 대상으로 포함되지 않는다. UML 모델이 개체의 대상이 되는 경우는 이 meta-architecture view에 기반하는 process view와 MVC view의 메타모델이다. 계층 모델간의 관계 예로서, M2의 "UML / .NET/ EJB 메타모델"(EJB: Enterprise Java Bean)을 사용해서 M1의 "A Banking System Model"을 생성함을 의미한다.



(그림 6) meta-architecture view 기반 계층적 메타모델

4.2 Process view 기반 계층적 통합 메타모델

process view 기반 피라미드 구조(그림 3)에서, 4개의 개발 단계별로 계층화된 UML 모델들을 대상으로, PIM/PSM의 MDA 기준에 부합하도록, 각 모델이 가진 모델링 요소들을 가지고, process view 기반 계층적 통합 메타모델을 정의한다. 메타모델의 표현은 아키텍처의 구문적(syntactic)인 구조를 잘 표현해주는 클래스 모델로 나타낸다. process view 기반 계층적 통합 메타모델은 (그림 7)에서 보여준다. 이 메타모델은 [7]에서 제시한 것을 확장 및 수정하였다. 즉, MDA 기반 PIM 및 PSM 모델로 계층화하고, 객체 모델링에서 컴포넌트 모델링으로 확장하여 제시한다. 또한, UML V2.0으로 현행화하고, 구현의 단계를 추가하고, GUI 모델링 요소를 추가하고, 그리고 UML 모델별 다양한 granularity level을 반영한다. (그림 7)에서 개발 프로세스에 의한 단계는 모델링의 추상화 수준에 따라 요구분석 단계, 기본설계 단계, 상세설계 단계와 구현 단계로 구성한다. 각 단계별 모델링 대상의 granularity는 각각 service/ class/ object/ component이다. 계층적 통합 메타모델의 구축과정은 모델의 분할 기준

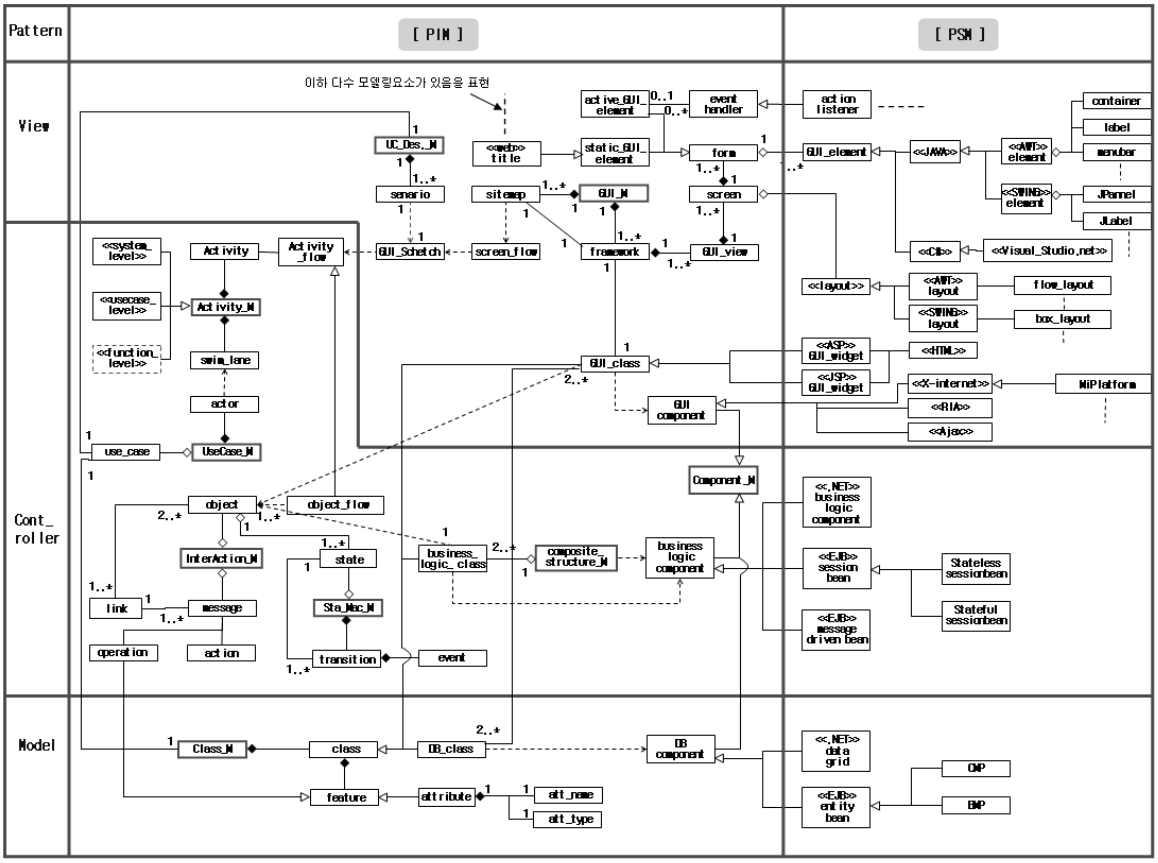


(그림 7) Process view 기반 계층적 통합 메타모델

및 메타모델의 생성규칙에 따라서, (그림 3)의 계층별 분할된 모델들을 대상으로 모델별 모델링 요소들을 추출하고, 모델링 요소들간 관계를 추출하고, 개발 단계에 의해 모델링 요소를 계층화한다. 이후에, 이들 간의 구조와 관계는 상호 매핑하는 모델링 요소들을 가지고 연관을 나타내고, 다중성(multiplicity)을 포함해서 클래스 모델로 표현한다. (그림 3)의 피라미드 구조상에서 요구분석 단계를 위한 구축 모델로 “UC_M”, “Sys. Activity_M” 그리고 “Sys. Sta_Mac_M”로 정의하고 있다. 여기서, “Sys. Activity_M”모델은 모델링의 “시스템” 크기를 대상으로 활동 모델이 가진 모든 모델링 요소를 가지고 구축한다. 따라서, 활동 모델의 모델링 요소로서 액티비티, 액티비티 흐름, 스윔레인 등으로 이루어지는데, 이들을 각각 “activity”, “activity_flow”, “swim_lane”의 모델링 요소로 명명하여 추출한다. 다음에 “Activity_M”(활동 모델을 나타내는 요소)에 이들 모델링 요소를 관계시킨다. 즉, “Activity_M”과 “activity”간에 관계 표현은 활동 모델은 액티비티가 필수적 요소이고, 다수의 액티비티들을 가질 수 있으므로, 복합(composition)과 1:1..*의 관계로 표현한다. “system_level”과 “Activity_M”간에는 상속관계로 표현한다. 한편, PIM 모델을 생성하기 위한 기본설계 단계는 클래스 모델 등을 대상으로 이 수준에 부합한 모델링 요소들로 추출하여 메타모델을 구축한다. 즉, PIM 모델 생성을 위한 UML 모델링 요소들의 구성은 구현 플랫폼에 독립적인 모델링 요소들로 이루어진다. 반면 PSM 모델은 구현 환경에 종속적인 모델링 요소들로 구성한다. 구현 플랫폼에 종속적인 PSM 모델 생성을 위해, .NET과 EJB[12]를 대상으로 모델링 요소들을 정의하였으며, 이들 플랫폼에 부합하는 GUI 관련 구현 언어들인 ASP와 JSP를 대상으로 하였다. 구현 단계에 대해서도 .NET과 EJB 플랫폼에 종속하여 관련 구현 실행파일들을 모델링 요소로 정의하였다. 단계간 메핑의 예로서, PSM 모델링 요소(예, “<<.NET>>”)와 구현 모델링 요소(예, “<<.NET implementation>>”)간의 관계는 realization으로 표현한다. 아울러, 이 통합 메타모델에 컴포넌트 모델을 포함하고 있으므로 컴포넌트 형태로 PIM과 PSM 모델을 각각 구축할 수 있다. (그림 7)에서 모델이 가진 모델링 요소들 중에 필수요소와 선택요소를 구분하기 위해 클래스 표현을 실선 사각형과 점선 사각형으로 각각 표현한다.

4.3 MVC view 기반 계층적 통합 메타모델

MVC view 기반 피라미드 상세화 구조(그림 5)에서, MVC 패턴에 의해 구성된 UML/GUI 모델들을 대상으로 그 모델링 요소들을 가지고 PIM과 PSM 방식으로 분할하여 MVC view 기반 계층적 통합 메타모델을 정의한다. 메타모델의 구축과정은 단원 4.2의 process view 기반 메타모델 정의와 동일하며, 단지 MVC 와 PIM/PSM 그리고 구축 대상 모델 및 모델링 요소들이 상이한 것뿐이다. (그림 8)에서,



(그림 8) MVC view 기반 계층적 통합 메타모델

"View"는 GUI 모델링을 위한 것으로서 "UC_Des_M"과 "GUI_M" 모델들이 가진 모델링 요소들로 메타모델을 정의한다. "GUI_M" (GUI Model)은 [13]에서 제시한 specific UI metamodel과 concrete UI metamodel의 모델링 요소들중 핵심 요소들을 추출하여 정의하고, 새로이 GUI component를 생성하기 위한 모델링 요소들을 추가하였다. GUI 모델링의 PSM 모델은 JAVA를 대상으로 AWT와 SWING을 모델링 요소로서 추출하고 스테레오타입을 이용해서 표현한다. 이것을 나타낸 것이 "<<JAVA>>"와 "<<AWT>> element"모델링 요소이다. 또한, 구현 플랫폼의 .NET과 EJB 환경을 반영하여 ASP와 JSP를 대상으로 GUI 관련된 모델링 요소들을 추출하고, 정의한다. "Controller"는 business logic을 모델링하는 것으로서, "Activity_M", "Sta_Mac_M", "InterAction_M"과 "Composite_Str_M" 모델의 모델링 요소들로 메타모델을 정의한다. "Model"는 data 모델링을 위한 것으로서 "Class_M"에 관계된 모델링 요소들로 메타모델을 표현한다. "View"(예, GUI component), "Controller"(예, Component_M)과 "Model"(예, DB component)들간의 관계는 상호 관련성이 있는 모델링 요소들로 매핑한다. 따라서, MVC 통합 메타모델을 통해서 M/ V/ C별로 GUI 컴포넌트, 비즈니스 로직 컴포넌트 그리고 DB 컴포넌트를 구축할 수 있다. 또한, M/ V/ C별 컴포넌트는 PIM 및 PSM 패턴의 계층적 모델링 수준에 의해 component 모델을 생성할 수 있다.

결국, 제시한 process view와 MVC view 기반 통합 메타모델에 준거하여 개발단계별로 해당 모델의 모델링 요소를 가지고 어플리케이션의 모델링 작업을 용이하게 할 수 있다. 아울러, MDA 방식으로 PIM과 PSM 방식으로 독립적 모듈화하여 모델을 구축함으로서 재사용성을 제고시킬 수 있다.

5. ISMS 적용사례

제시한 2+1 기반 통합 메타모델을 사용한 계층적 컴포넌트 모델링을 입증하기 위해, 사례 적용으로 인터넷 쇼핑몰 시스템(ISMS)을 대상으로 디자인한다. 적용 방법은 4장에서 제시한 개발 단계별 process view 기반 통합 메타모델과 MVC view 기반 통합 메타모델에 정의한 계층적 모델링 요소들을 사용해서 개발 프로세스별/ MVC별 모델링을 수행한다. 통합 메타모델에 기반하여 실제로 모델링된 결과 산출물은 다음 <표 1>과 같다. use case model, class model, sequence model 그리고 component model을 모델링하였다. 모델링된 산출물 중에, 핵심 모델인 class model과 component model을 발췌하여 사례로서 기술한다. 이들 모델은 process view기반 통합 메타모델의 해당 모델링 요소를 사용하여 기본설계 단계에서 PIM 모델을 생성하고, 상세설계 단계에서 PSM 모델을 구축한다. 동시에 MVC view 기반 통합 메타모델의 해당 모델링 요소를 사용해서 PIM 모델과 PSM 모델에 MVC를 가미, 표현해서 작성한다.

〈표 1〉 ISMS 사례 적용통한 구축 모델 산출물

구 분	구축 모델 산출물	
	Process View 통합 메타모델 적용	Architecture View 통합 메타모델 적용
분석 단계	Use case model	
기본 설계 단계 + MVC	Use case Description PIM Class Model PIM Interaction Model (Sequence Model) PIM Component Model	
상세 설계 단계 + MVC	PSM Class Model PSM Interaction Model (Sequence Model) PSM Component Model * 대상 구현환경 : Java, J2EE	
구현 단계		

5.1 PIM 모델의 구축

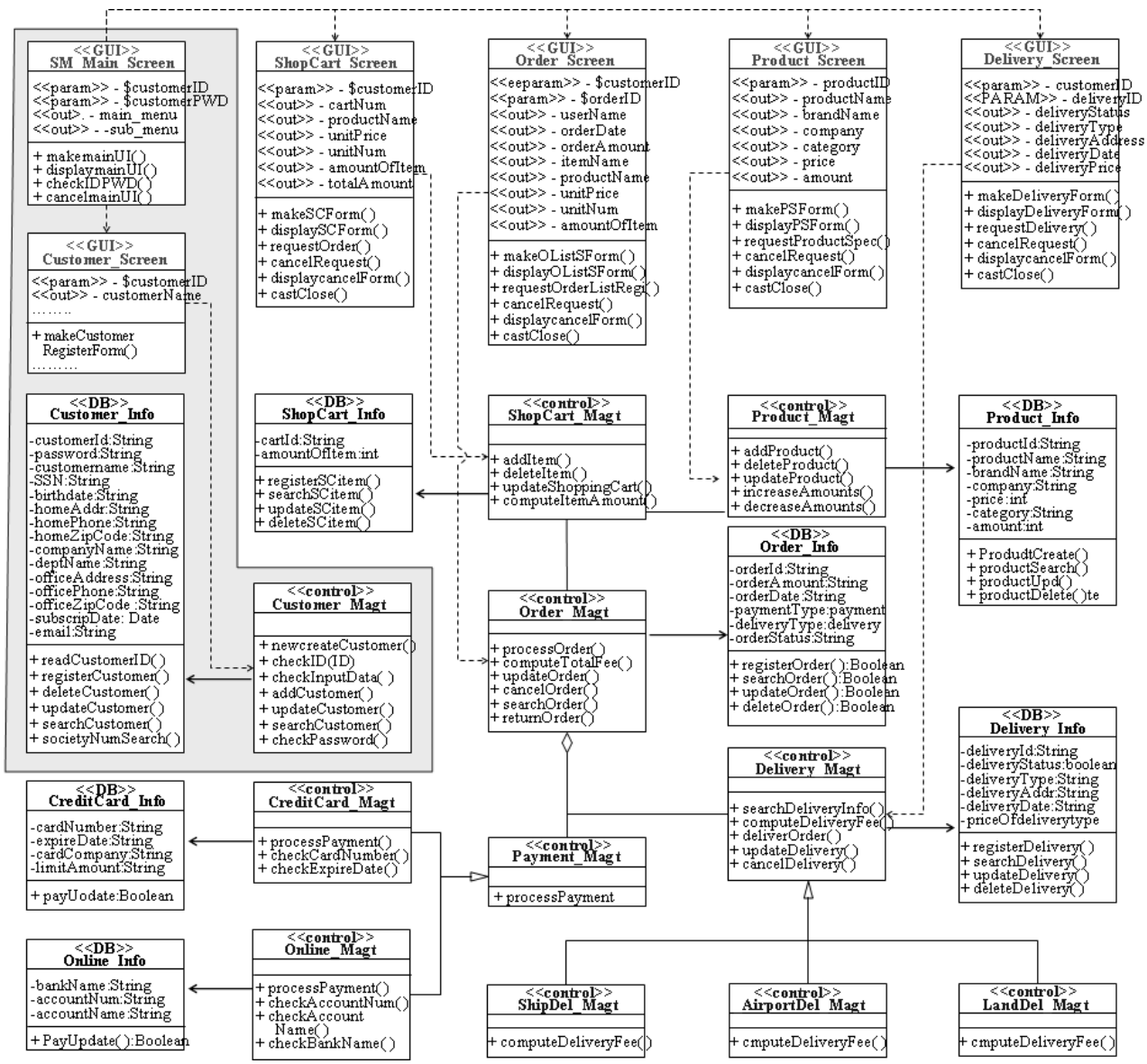
ISMS에 대한 개략 설계로서 PIM 수준의 PIM Class model과 PIM Component model을 생성한다. 생성 방법은 (그림 7)과 (그림 8)에 정의한 PIM 부문에 해당하는 Class metamodel과 component metamodel의 모델링 요소들을 사용해서 구축한다.

따라서, class model의 생성은 process view에 의한 (그림 7)의 메타모델에서 정의한 "class", "attribute", "operation", "visibility", "class", 이들간 relationship의 모델링 요소들 그리고 MVC view에 의한 (그림 8)의 메타모델에서 정의한 "GUI_class", "Business_logic_class" 및 "DB_class"를 사용해서 모델링한다. 생성된 PIM class model은 (그림 9)에서 보여준다. (그림 9)에서 보여주듯이, 개발단계와 MVC에 의해 PIM 수준으로 모델 명세의 추상화 수준을 명확히 표현할 수 있다.

PIM component model의 작성은 (그림 7)과 (그림 8)의 계층적 메타모델내 PIM 수준의 모델링 요소인 "component", "interface" 및 "class"들의 모델링 요소를 사용해서 표현한다. 이 PIM component model은 ISMS에 대해서 GUI_component, Bus.L component 그리고 DB component들로 구성된다. 예시로서, GUI_component 부문에 대해 3개로 구성된 컴포넌트를 모델링한 것이 (그림 10)이다. 각 컴포넌트는 MVC 패턴에 의해 표현한다.

5.2 PSM 모델의 구축

PIM 모델과 달리, 구현환경 혹은 플랫폼에 종속적인 PSM 모델의 생성은 (그림 7)과 (그림 8)에 정의한 PSM 부문의 모델링 요소를 사용해서 PSM class model과 PSM component model을 구축한다. 적용 사례를 위한 구현 플랫폼 및 언어로서 각각 EJB와 JSP를 대상으로 하였다. PSM class model의 생성은 PIM class model을 기반으로 구현환경에 부합토록 추가적 모델링 요소로서, "interface"(home interface /remote interface) 추가, "entityBean" / "sessionBean" / "JSP"를 사용하여 클래스의 스테레오타입 명칭 변경, 클래스내 메소드 추가(예, entityBean에 "access method" 추가, sessionBean에 "business method" 추가), 제어 클래스의

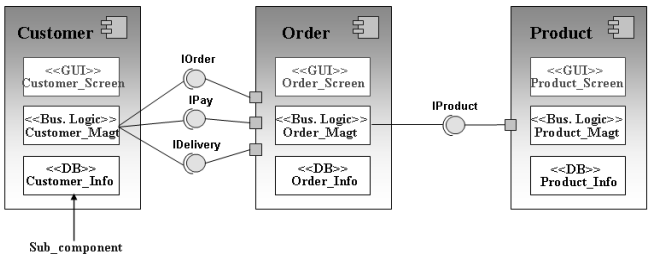


(그림 9) PIM Class model

오퍼레이션에 파라메터 추가, 그리고 helper class 추가(예, customer Data) 등이다. (그림 11)은 PSM class model 적용 예로서, (그림 9)상에서 회색박스로 표현된 "고객관리" 부문에 대해 MVC를 반영하여 PSM 모델을 모델링한 것을 보여준다.

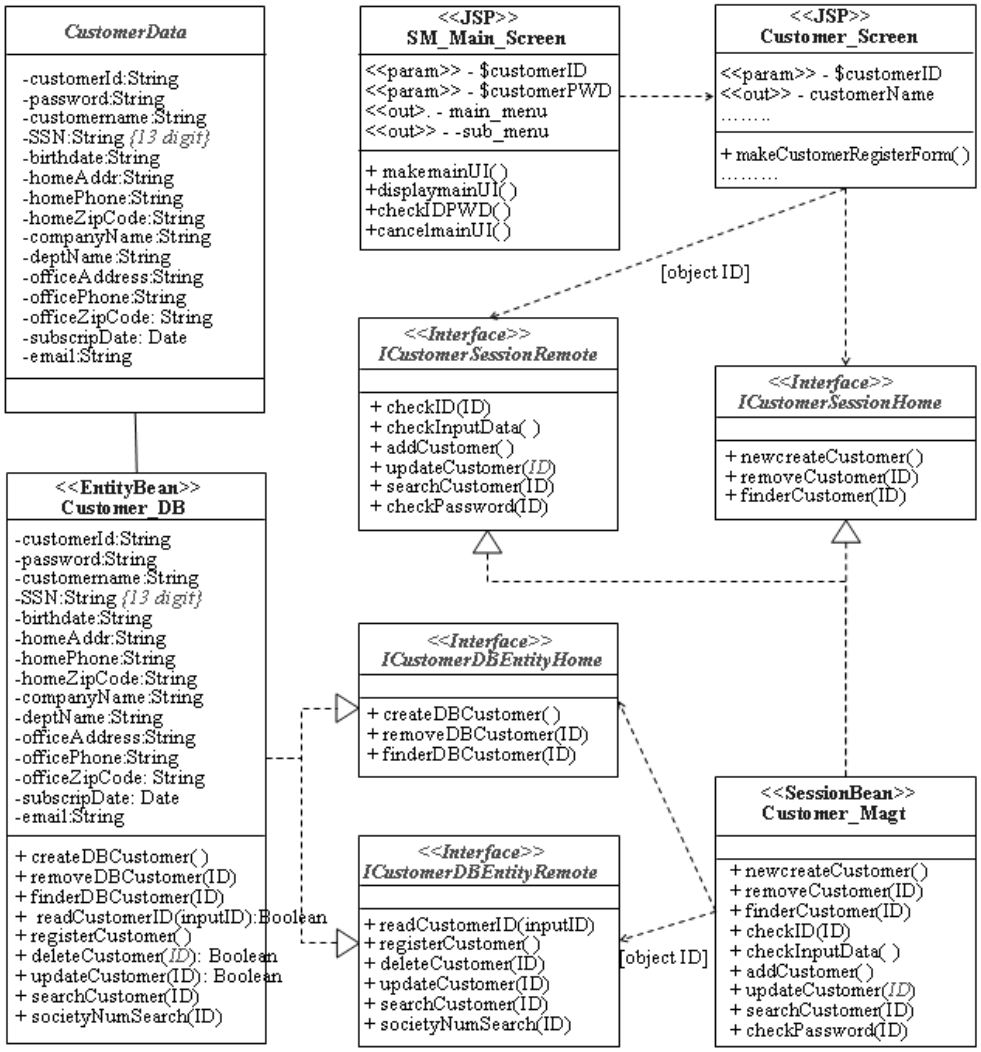
한편, PSM component model의 구축은 PIM component model을 기반으로 구현 플랫폼에 부합토록 "<<EJB>> session bean", "<<EJB>> entitybean"등의 모델링 요소를 사용한다. 그외, 인터페이스 추가, 컴포넌트내 클래스들로 실체화 등의 추가적 모델링을 통해 작성한다. PSM component model의 구축 예로 (그림 12)에서 보여주듯이, (그림 10)의 PIM 모델을 기반으로 MVC의 3계층에 의해 interface와 class와의 관계를 각각 <<implement>>와 <<realization>>으로 나타내고, artifact를 추가하여 표현하였다.

구축 사례에서 기술하지 않은 interaction model에 대해

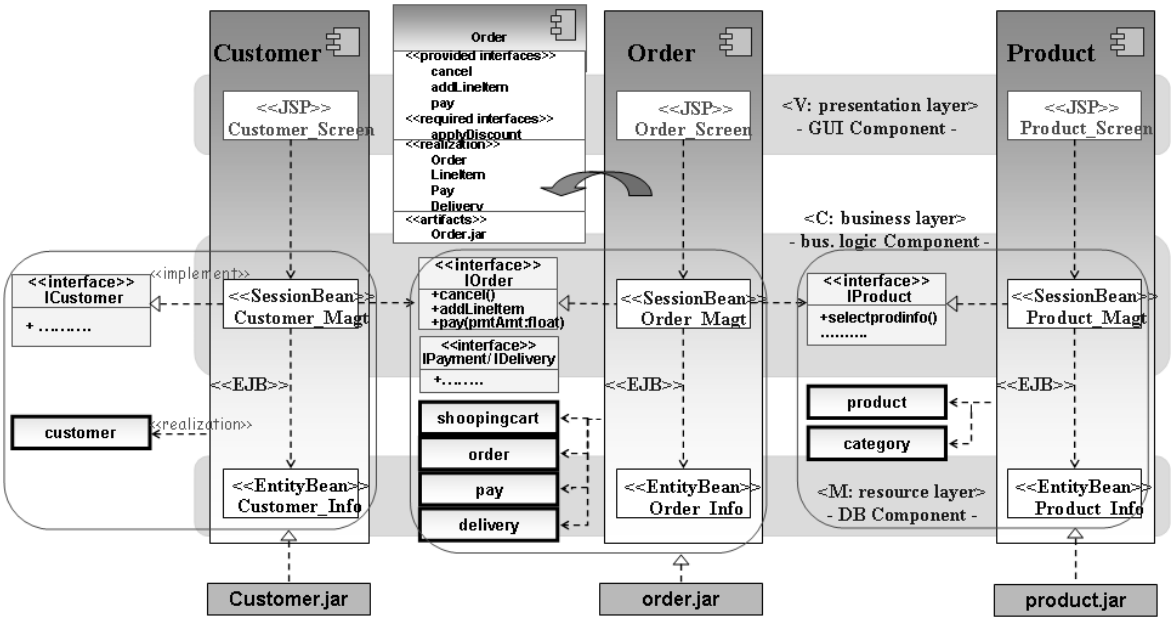


(그림 10) PIM Component model

PIM 모델과 PSM 모델간의 차이점을 살펴보자. PIM Interaction model에 PSM Interaction model 생성에서 추가되는 모델링 요소는 홈 인터페이스와 리모트 인터페이스 표기, 실제 해당 객체 대응의 클래스 오퍼레이션을 메시지로 표현, 스테레오타입, 옵션, 파라메터 타입 그리고 제약사항 등이다.



(그림 11) PSM Class model



(그림 12) PSM Component model

6. 비교 평가

ISMS 시스템 모델링의 적용사례를 통해, 제시한 개발단계별 메타모델과 MVC 메타모델의 모델링 요소를 사용하여 MDA 기반의 계층적 PIM 모델과 PSM 모델의 생성이 용이하고 쉽게 컴포넌트 모델링이 수행됨을 보였다.

6.1 기존 연구와의 비 정량적 비교평가

기존 연구와의 평가를 메타모델 아키텍쳐 측면, 메타모델 기반 모델링 측면 그리고 생성 모델의 품질 측면에서 비교한다. 먼저, 메타모델 아키텍쳐 측면에서 비교한 것이 <표 2>이다.

<표 2>에서, [5]에 대해 "UML Metamodel", [7]에 대해 "Layered metamodel" 그리고 제시 방법은 "2+1 view meta-model"로 약칭한다. UML 메타모델은 전체 UML은 아키텍쳐 및 개별 메타모델은 우수하나, 계층적 메타모델화 및 MVC 패턴화 등이 미약하다. Layered metamodeld은 계층적 개별 메타모델은 우수하나, 구현 단계 및 MVC 패턴을 지원하지 않는다.

<표 3>은 메타모델 기반 모델링 측면의 비교를 보여준다. [6]에 대해 "CBD_Bible modeling" 및 [9]에 대해 "UP modeling"으로 명명한다. <표 3>에서 보여주듯이, 기존 방법론들은 MVC 기반의 컴포넌트 모델링 및 PIM/PSM 기반의 모델링이 충분치 못함을 알 수 있다. 즉, 기존 MVC 모델링은 MVC 모델링을 위한 메타모델을 정의하지 않았으며, 단지 객체 수준에서의 MVC 모델링을 제공한다. 반면, 본 논문은 MVC를 위한 메타모델을 정의하고, 이 메타모델에 기반한 객체와 컴포넌트 수준에서의 모델링을 제공한다. <표 3>에서 가중치 합계의 산정은 각 비교 항목의 만점은 1로 하여 범례 기준에 따라 평가 항목의 점수를 합산하여 계산한다. 충족 백분율의 산정은 만점 가중치 점수4 대비 얻은 가중치 점수를 백분율로 계산한다. 예로 2+1 view modeling의 경우, "++"가 3개이고 "+"가 1개 이므로 총 가중치 합계 점수는 3.75가 된다. 이를 백분율로 산정하면 94%(4 : 3.75 = 100% : x)가 된다.

다음은 각 방법별 생성 모델의 품질 측면에서 비교한 것이 <표 4>이다. 평가척도로서, 이해성은 메타모델을 제공한다면, 모델의 이해가 쉽고 개발단계별 명확한 구분 모델화가 가능하기에 이해성이 높은 것으로 평가한다. 일관성과 추적성은 MVC를 제공한다면, 요구기능이 GUI에서 Business logic으로 또한 DB로의 일관성이 높은 것으로 측정한다. 재사용성은 M/V/C별의 구축 모델과 개발 수준에 의한 PIM/PSM 모델의 분리된 모델 구축을 제공한다면 재사용성이 높은 것으로 평가한다.

〈표 2〉 메타모델 아키텍쳐 측면 비교

구 분	UML Metamodel	Layered Metamodel	2+1 view Metamodel
전체적/조직적 아키텍쳐 메타모델	++	+	+
계층적 개별 메타모델	-	++	++
계층적 통합 메타모델	-	+	++
전 개발단계 지원수준	0	+	+
구현 단계 지원 메타모델	-	0	+
MVC 패턴 지원	-	-	+
메타모델 확장성	+	+	+
가중치합계 (7 만점)	3.25	4.75	5.75

[범례] 가중치(백분율)/ 지원수준 , ++: 1.0/ very strong, +: 0.75/ strong, 0: 0.50/ average, -: 0.25/ weak, --: 0.0/ very weak

〈표 3〉 메타모델 기반 모델링 측면 비교

구 분 (가중치)	CBD_Bible modeling	UP modeling	Layered modeling	2+ 1 view modeling
MVC 기반 Class 모델링 (1)	++	++	0	++
MVC 기반 컴포넌트 모델링 (1)	0	-	-	+
PIM/PSM 기반 모델링(MDA) (1)	+	0	+	++
개발단계별 모델링 (1)	++	++	++	++
가중치 합계 (4) / 충족 백분율(100%)	3.25 (81%)	2.75 (69%)	2.50 (63%)	3.75 (94%)

[범례] 가중치(백분율)/ 지원수준 , ++: 1.0/ very strong, +: 0.75/ strong, 0: 0.50/ average, -: 0.25/ weak, --: 0.0/ very weak

〈표 4〉 구축 모델의 품질 측면 비교

구 분	CBD_Bible mothod	UP method	Layered method	2+1 view method
모델의 이해성	+	+	++	++
모델간 일관성/ 추적성	+	+	+	+
모델의 재사용성	++	+	+	++
가중치 합계 (3) / 충족 백분율(100%)	2.50 (83%)	2.25 (75%)	2.50 (83%)	2.75 (92%)

6.2 재사용성의 정량적 비교평가

재사용성에 대한 기존 방법들과의 정량적 평가로서 <표 4>의 평가척도중 "재사용성"에 대해 분석한다. 재사용 가능 모델 수를 산정하는 방식은 다음과 같다.

- 산정 방법
  - 재사용 가능 구축 모델 수 = PIM/PSM 모델수(개발단계수) × MVC 모델수 × UML 모델수
  - 개발과정에서 생성되는 재사용 모델 수들을 누적하여 얻을 수 있다.
  - 단, 개발 단계수는 PIM과 PSM 모델 생성의 2개 단계를 대상으로 한다.
- 산정 수식

$$\bigcup_{(R)}^{Num} = \sum_{j=PIM}^{PSM} (MDA\ model) \left(\sum_{k=M}^{C} (MVC\ model) \left(\sum_{l=\ ClassM}^{CompM} (UML\ model) \right) \right)$$

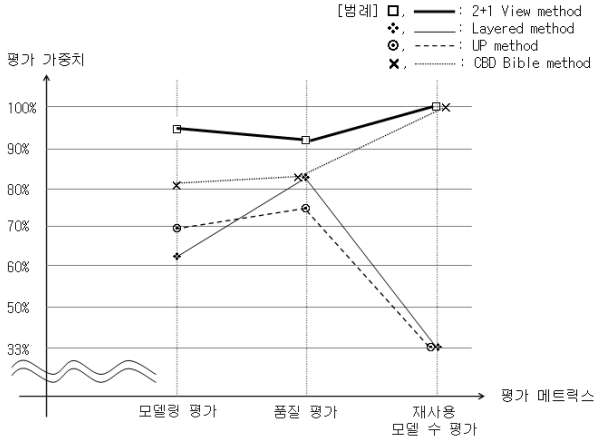
상기의 산정 수식에 의거, 각 모델링 방법들과의 재사용 모델 수를 비교한 것을 <표 5>에서 보여준다. 여기서, UML 대상 모델은 class model, component model 및 sequence model이다.

〈표 5〉 재사용 가능 모델 수의 분석

구 분	CBD_Bible method	UP method	Layered method	2+1 view method
PIM/ PSM 수	2	0	0	2
MVC 수	3	2	2	3
사용 UML 모델 수	3	3	3	3
재사용 가능 모델 수 / 충족 백분율 [100%]	18 [100%]	6 [33%]	6 [33%]	18 [100%]

<표 5>에서, 제시한 2+1 view 기반 컴포넌트 모델링 방법은 UML 모델, PIM/PSM 모델 그리고 MVC 모델 모두를 제공함으로서 구축되는 재사용 모델의 수를 극대화할 수 있다.

지금까지 정량적 및 비 정량적 평가의 전체에 대해서 즉, <표 3>(모델링 평가)/ <표 4>(품질평가)/ <표 5>(모델의 재사용 수)의 비교결과를 가지고, 이를 총괄적으로 시각화하여 그래프로 표현한 것이 (그림 13)이다.



(그림 13) 기존 연구 대비 비교분석 총괄결과

7. 결 론

본 논문은 모델링 산출물의 재사용성을 향상시키기 위해, MDA(PIM/PSM) 기반의 2+1 view 메타모델에 의한 컴포넌트 모델링 기법을 제시하였다. 개념적 원리로서, 모델 지향 모델링, 계층적 모델링 그리고 프로세스를 가미한 메타모델 기반 모델링 접근을 사용했다. 그래서, 4개 개발단계의 프로세스와 MVC 패턴에 의해 UML 모델과 GUI 모델을 가지고 각각 계층적 통합 메타모델을 정의했다. 이렇게 정의된 메타모델내 모델의 모델링 요소를 사용해서 ISMS 용용시스템의 클래스 모델링과 컴포넌트 모델링에 적용하였다. 또한, 메타모델 구조 측면, 메타모델 기반 모델링 측면, 생성 모델의 품질 측면의 비정량적 평가와 그리고 모델의 재사용수에 의한 정량적 평가를 통해 제시 기법의 효과성을 입증하였다.

기대효과로서, UML 메타모델에 개발 프로세스와 MVC에 의한 통합 메타모델을 제시함으로서 UML 적용성을 위한 view의 확장이 가능해졌다. 또한, MVC 패턴에 의거 각 어플리케이션 계층별 GUI/ Business logic/ DB로 분할된 컴포넌트 모델링이 가능해졌다. 특히, PIM/PSM과 MVC의 MDA 개발방식에 따른 독립적 모듈성이 강한 모델을 할 수 있으므로 재사용성을 향상시킬 수 있다. 또한, 제시 메타모델내 모델링 요소에 기반하여 모델링하기 때문에 UML 모델의 사용이 용이하고 일관성 있고 모델을 구축할 수 있을 것이다.

향후 연구로서, 제시된 메타모델의 UML 표준으로의 체화 방법, SOA(Service Oriented Architecture) 기반의 메타모델 기반의 모델링 방법, 그리고 PIM 모델에서 PSM 모델로의 변환 알고리즘을 정립하여 이를 Case tool 지원을 통한 정형적 모델 체킹이 필요할 것이다.

참 고 문 헌

[1] 임윤선, 김명, 정승남, 정안모, “컴포넌트 재사용을 지원하는 컴포넌트 모델 및 프레임워크”, 정보과학회 논문지, 제34권, 제12호, Dec., 2007.

[2] Object Management Group, MDA Guide Version 1.0.1, 2003. http://www.omg.org/docs/omg/03-06-01.pdf

[3] T. Schattkowsky and M. Lohmann, “UML Model mappings for platform independent user interface design”, Springer-Verlag Berlin Heidelberg, LNCS 3844, pp.201-209, 2006.

[4] J. Sadd, “DEFINING THE OPENEDGE® REFERENCE ARCHITECTURE - PRESENTATION: MODEL-VIEW-CONTROLLER PATTERN”, PROGRAM SOFTWARE, 2006, http://www.progress.com

[5] Object Management Group, Unified Modeling Language: Infrastructure V2.1.1, 2007, http://www.omg.org/docs/formal/07-02-4.pdf

[6] 채흥석, 객체지향 CBD 개발 Bible, 한빛미디어출판사, 2003.

[7] C.Y. Song and D.K. Baik, “A Layered Metamodel for Hierarchical Modeling UML”, International Journal of Software Engineering and Knowledge Engineering, Vol.13, No.2, pp. 191-214, 2003.

[8] 민현기, 김수동, “컴포넌트 설계를 MDA/PIM으로 명세하기 위한 UML 프로파일”, 정보과학회 논문지, 제32권, 제3호, Mar., 2005.

[9] P. Kruchten. “The Rational Unified Process - An Introduction”, Addison-Wesley, Reading, Mass., &c., 2nd edition, 2000.

[10] 조은숙, 박수희, 장준호, “수요지향 교과과정 개발을 위한 3차원 기반의 메타모델 설계기법”, 한국컴퓨터교육학회 논문지, 제8권, 제6호, Nov., 2005.

[11] 이현주, 최병주, 이정원, “서비스 지향 아키텍처를 위한 컴포넌트 기반 시스템의 서비스 식별”, 정보과학회 논문지, 제35권, 제2호, Feb., 2008.

[12] E.S. Cho, S.D. Kim, and S.Y. Rhew, “A Domain Analysis and Modeling Methodology for Component Development”, International Journal of Software Engineering and Knowledge Engineering, Vol.14, No.2, 2004.

[13] 송치양, 조은숙, 김철진, “메타모델 기반 사용자 인터페이스 계층적 모델링 프로세스”, 멀티미디어학회 논문지, 제11권, 제4호, Apr., 2008.

송 치 양

e-mail : cysong@knu.ac.kr
1985년 한남대학교 전산학과(학사)
1987년 중앙대학교 전산학과(이학석사)
2003년 고려대학교 컴퓨터학과(이학박사)
1990년~2005년 한국통신 중앙연구소 책임연구원
2005년~2007년 상주대학교 소프트웨어공학과 조교수
2008년~현 재 경북대학교 소프트웨어공학과 조교수
관심분야 : UML 모델링 기술, 소프트웨어 개발방법, IP-TV 서비스

조 은 숙

e-mail : escho@seoil.ac.kr
1993년 동의대학교 전산통계학과(학사)
1996년 숭실대학교 컴퓨터학과(석사)
2000년 숭실대학교 컴퓨터학과(박사)
2002년~2003년 한국전자통신연구원 초빙연구원
2000년~2004년 동덕여자대학교 정보학부 전임강사
2005년~현 재 서일대학 소프트웨어과 조교수
관심분야 : 컴포넌트 기반 소프트웨어 개발, 임베디드 소프트웨어개발, 유비쿼터스 컴퓨팅