사용 관점 중심의 컴포넌트 모델링

김 태 웅[†] · 김 경 민[††] · 김 태 공[†††]

요 약

컴포넌트 기반 개발은 시스템을 이해하고 분석하기 위한 컴포넌트 모델링 방법이 중요한 부분을 차지하고 재사용성을 높이는 방안으로써 받아들여지고 있다. 컴포넌트는 일반적으로 컴포넌트를 개발하는 사람과 그것을 조립하여 사용하는 사람의 관점이 다르며 이에 따른 내용과 목적이 다르기 때문에 대규모의 컴포넌트 시스템을 개발하는데 있어서 그 역할에 따라 두 가지 유형으로 구분하는 것이 중요하다. 이것이 현실화되기 위해서는 명확한 컴포넌트와 인터페이스 명세가 필요하며 조립자와 개발자 간의 서로 다른 관점에서의 컴포넌트 모델이 필요하다. 이에 본 논문에서는 조립자 관점과 생성자 관점이라는 서로 다른 역할에 따라 두 가지 유형의 컴포넌트 모델을 제안하며 이를 위해 UML을 확장한다. 또한 제안한 모델을 작성, 관리하며 모델간의 자동변환을 위한 툴을 개발하여 적용해 봄으로써 그 효용성을 검증한다.

키워드 : 컴포넌트기반개발방법론, 컴포넌트 모델링, UML, 모델링언어, 모델 자동변환

Component Modeling Focusing on View-point of Component Use

Kim Tae Woong[†] · Kim Kyung Min[††] · Kim Tae Gong[†††]

ABSTRACT

In component based development, component modeling for understanding and analyzing is the important part and is used to improve reusability. Generally, components are need to be divided into two types according to their usages, where the developer and assembler are usually different. To make a good component model, a complete component and interface specification for those components are needed. And the component model needs to adopt two different views of developer and assembler. In this paper, we suggest two different views of component model that is related to views from developer and assembler, and we expand UML. Also we validate the efficiency of the suggested model by developing and applying a tool for building, managing and automatic transformation.

Key Words : CBD(Component Based Development), Component Modeling, UML(Unified Modeling Language), Modeling Language, Automatic Model Transformation

1. 서 론

소프트웨어 컴포넌트란 하나 이상의 기능을 갖는 독립적인 소프트웨어이며, 조립을 통해 응용프로그램을 작성할 수 있는 부품 형태의 소프트웨어를 말한다[1]. 컴포넌트는 시스템 구축 시에 구성요소의 모듈화를 촉진시키며 인터페이스를 통하여 다른 컴포넌트의 조립이 가능하게 하며 개발 시 병행 개발이 가능하도록 한다. 응용프로그램 개발 시 이미 작성되어 있는 소프트웨어들의 조합을 통해 응용프로그램을 작성할 수 있다면 개발자의 노고와 비용의 절감을 가져와 높은 효율을 가져 올 수 있다[2]. 즉, 경험이 많은 프로그램 개발자에 의해 만들어진 컴포넌트들을 재사용함으로써 신뢰성, 안정성 및 유지보수성이 매우 뛰어난 소프트웨어를 개발할 수 있는 것이다. 이러한 컴포넌트 개발을 위해서는 시스템을 이해하고 분석하기 위한 컴포넌트 모델링 방법이 중요한 부분을 차지하고 있다. 컴포넌트 모델링은 대상을 종합적으로 이해하고 주어진 조건이나 입력에 대한 대상의 반응을 예측할 목적으로 비교적 엄격한 규칙에 따라 구현된 표현을 의미한다[3]. 컴포넌트 모델링 방법은 개발 초기 단계의 부정확한 산출과 불충분한 모델 구축의 문제를 해결할 수 있는 방법으로 간주되고 있으며, 개발 노력과 유지보수 비용을 줄일 수 있다.

이러한 컴포넌트 기반의 모델에는 UML을 기반으로 하는 RUP, Catalysis, Advisor (CBD96), MaRMI-III 등 컴포넌트 기반 개발 방법론에서 제공하는 모델들이 있다. 그러나 이러한 기존의 방법론을 이용한 모델링과 소프트웨어의 설계는 컴포넌트의 기본적인 특성인 인터페이스에 대한 표현이 단순하고 제한된 표현 방법으로 그 뜻을 나타내기 때문에 여러 가지로 해석이 가능할 수 있는 모호한 점이 있다. 그리고 내부 디자인의 규모가 객체지향 방법론의 중심이 되는

† 정 회 원 : 인제대학교 컴퓨터공학부 유비쿼터스 지능형 홈 인력 양성사업단 계약교수
†† 준 회 원 : 인제대학교 일반 대학원 전산학과(박사 수료)
††† 정 회 원 : 인제대학교 컴퓨터공학부 교수
논문접수 : 2006년 8월 22일, 심사완료 : 2007년 2월 15일

객체 혹은 클래스와 컴포넌트 사이에 큰 차이점을 보이게 되므로 정확한 컴포넌트의 설계 및 표현에 한계가 있다. 또한 컴포넌트 기반 개발과정은 크게 컴포넌트 조립에 의한 개발과 컴포넌트 생성에 의한 개발로 이루어지며, 이에 따라 컴포넌트 조합(composition)과 재사용성을 강조하기 때문에 전통적인 소프트웨어 방법론으로는 이런 특성을 적절하게 반영하기 어렵게 된다. 즉 기존 모델들에서는 컴포넌트 기반 개발 프로세스의 각 작업에 대하여 체계적이면서 구체적인 지침들을 충분히 제시하지 못하고 있다[4]. 그러므로 컴포넌트 기반의 개발에서는 컴포넌트를 표현하기 위해 그 특성에 적합한 표현방법이 필요하다고 할 수 있다. 컴포넌트는 일반적으로 컴포넌트를 개발하는 사람과 그것을 조립하여 사용하는 사람이 다르며 이에 따른 내용과 목적이 다르기 때문에 대규모의 컴포넌트 시스템을 개발하는데 있어서는 그 역할에 따라 두 가지 유형으로 구분하는 것이 중요하다.

이에 본 논문에서는 효율적인 컴포넌트 기반의 개발을 위해 조립자 관점과 생성자 관점이라는 서로 다른 역할에 따라 두 가지 유형의 컴포넌트 모델을 제한한다. 또한 모델을 생성, 저장 관리하고 두 모델간의 상호변환을 위한 자동변환 툴을 개발하여 그 효용성을 검증하고자 한다. 조립자 관점의 컴포넌트 모델에서는 호출하여 조립할 수 있는 컴포넌트 인터페이스 정보를 표현하고 생성자 관점의 컴포넌트 모델에서는 컴포넌트 전체 상호작용 관계와 서비스의 내부 비즈니스 로직에 관해 표현한다.

본 논문의 구성은 2장에서 기존 방법론에서 제공하는 컴포넌트 관련 모델들에 관해 연구하고 문제점을 분석한다. 3장에서는 문제점 해결을 위한 방안으로서 사용 관점에 따른 컴포넌트 모델을 제안한다. 4장에서는 컴포넌트 모델을 생성, 저장하고[5, 6] 두 모델간의 상호변환을 위한 자동변환 툴을 개발하여 적용하고 그 효용성을 검증한다. 끝으로 5장에서는 향후 연구과제를 포함하여 결론을 맺는다.

2. 관련연구

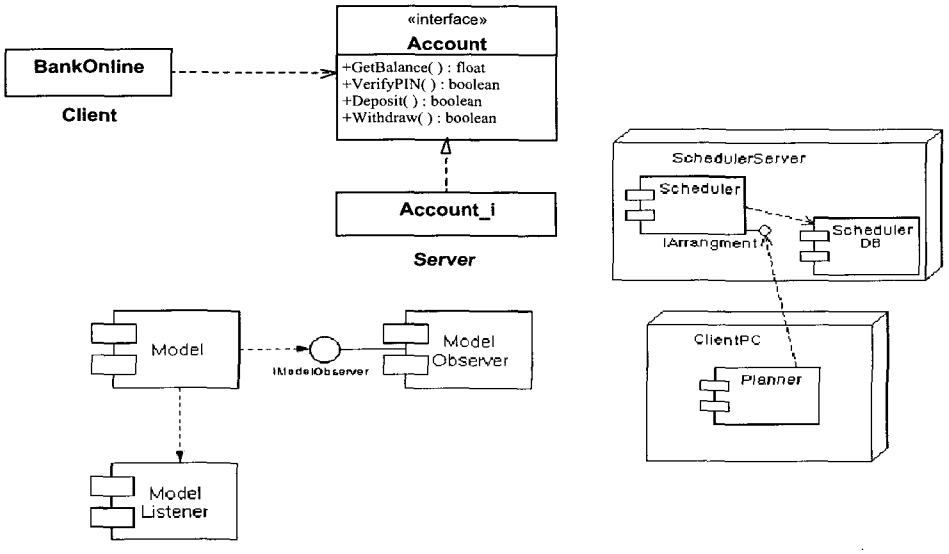
본 장에서는 본 논문의 사용 관점 중심의 컴포넌트 모델을 도출하기 위하여 기존의 개발방법론에서 제안하고 있는 컴포넌트 관련 모델과 그 문제점에 대해 살펴본다.

2.1 RUP에서의 컴포넌트 모델

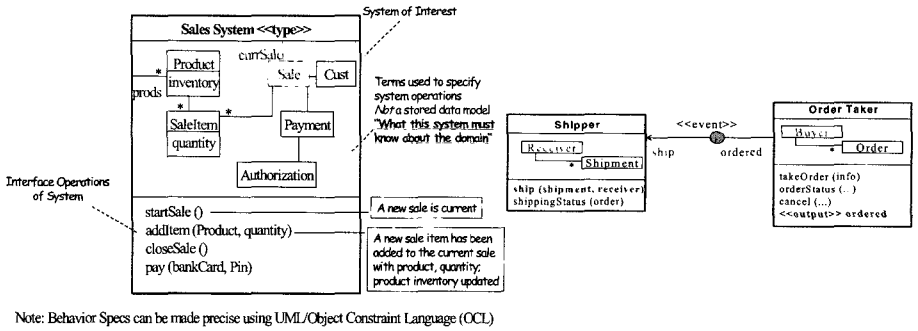
RUP[7]에서는 컴포넌트 개념을 전체 개발 공정 동안 표현하기 위해 UML에서 제공하는 컴포넌트 관련 모델을 그대로 수용하고 있으며, 그림 1과 같은 구성요소로 표현한다[8]. UML이 소프트웨어의 설계를 위한 표준 모델링 언어지만 모든 상황에 대한 의미를 표현하기에는 충분하지 않다. 그리고 객체지향 중심의 모델링 언어이기 때문에 컴포넌트의 특성인 인터페이스에 대한 표현이 단순하고 제한된 표현방법을 사용하여 정확한 컴포넌트의 표현에 한계가 있다. 또한 컴포넌트 개발은 컴포넌트 조립에 의한 개발과 컴포넌트 생성에 의한 개발로 이루어지는 반면 RUP의 컴포넌트 관련 모델에서는 이러한 구분 없이 표현하고 있다.

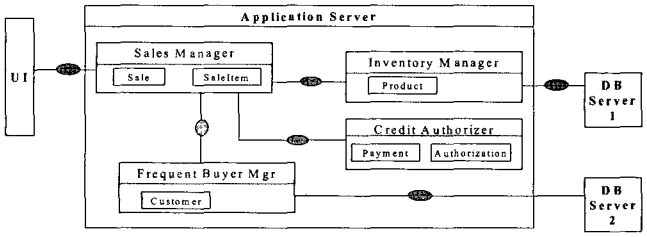
2.2 Catalysis에서의 컴포넌트 모델

Catalysis에서는 컴포넌트를 상호작용을 하는 객체들의 집합으로서 단순히 소프트웨어의 집합일 뿐 아니라 하나의 비즈니스 행위가 될 수 있다고 정의한다[8]. 그리고 컴포넌트라는 단위를 type이란 용어를 사용하여 정의하고 있으며[9] 객체에 대한 클래스 또한 타입이란 용어를 사용하며 그림 2와 같은 구성요소로 표현한다[10]. Catalysis 또한 현재의 컴포넌트 개발 방법론과 마찬가지로 추상적이고 일반적인 틀만을 제시하고 있기 때문에 실제로 소프트웨어를 개발하는 것과는 많은 차이가 있다. 또한 여러 모델들이 합쳐져 컴포넌트 개념을 기술하고 있기 때문에 컴포넌트 구현을 위한 내부 정보를 얻기 위해서는 컴포넌트 모델에서 역으로 방법론 각 단계들에서 제시하는 모델들을 추적하여 찾아야 하는 어려움이 있다.

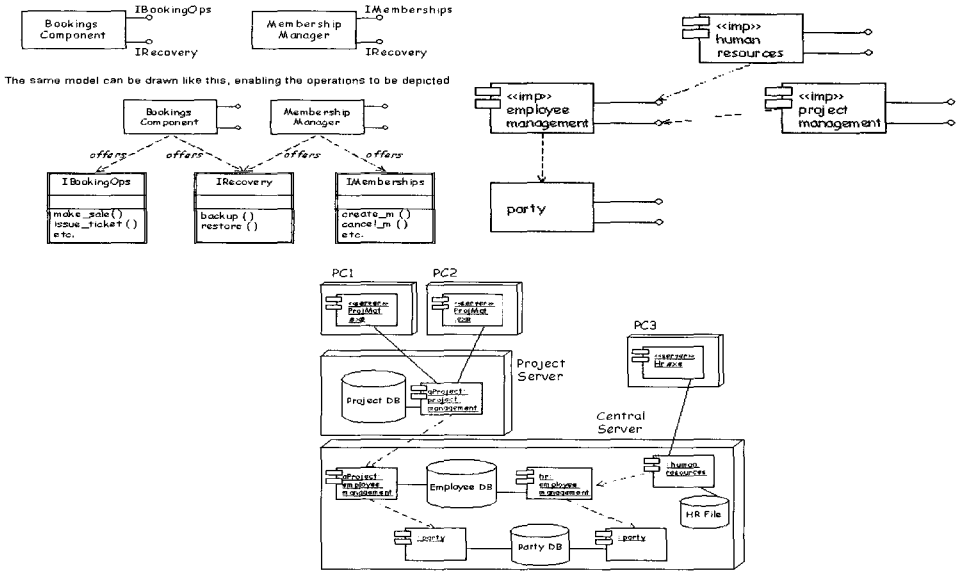


(그림 1) RUP에서 컴포넌트를 나타내기 위한 구성요소





(그림 2) Catalysis에서 컴포넌트를 나타내기 위한 구성요소



(그림 3) Advisor에서 컴포넌트를 나태내기 위한 구성요소

2.3 Advisor(CBD96)에서의 컴포넌트 모델

Advisor(CBD96)[11, 12]에서는 컴포넌트를 그 관점에 따라 명세, 구현, 실행물로 나누어 각자의 관심 영역을 명확히 구분하고, 개발의 초기 단계부터 인터페이스의 역할을 강조하고 있다. 또한 컴포넌트를 모델링 하는 과정에서 흔히 접하게 되는 다양한 문제들을 해결하기 위해 애플리케이션 개발트랙과 컴포넌트 공급트랙으로 구성되며, 각각 컴포넌트 기반 개발(CBD)과 단일 컴포넌트 개발(CD)을 지원하는 실질적인 기법들을 제시한다[13]. 그림 3은 컴포넌트를 이루는 구성요소를 표현하고 있다. 그러나 지금까지의 다른 방법론들과 마찬가지로 누구나 쉽게 모델을 기반으로 하여 개발하려는 조립자나 생성자에게 구체적이며 실질적인 컴포넌트 모델을 제시하지 못하고 있다.

2.4 MaRMI-III에서의 컴포넌트 모델

MaRMI-III[8, 14, 15]에서는 컴포넌트를 시스템의 재사용 및 대체 가능한 최소단위의 부품으로 정의한다. 이와 같은 컴포넌트 개념을 MaRMI-III 전체 개발 공정 동안 표현하기 위해 컴포넌트 명세서, 컴포넌트 인터페이스 정의서, 컴포넌트 협력도, 비즈니스 컴포넌트 모형 기술서, 컴포넌트 설계서 등으로 나타낸다. 개발의 초기부터 아키텍처를 강조하고 있지만, 그 개념이 모호하여 컴포넌트 조립자가 이러한 강점을 충분히 활용하기 어렵다. MaRMI-III 또한 각 개발 단계에서 컴포넌트 개념을 기술하고 있기 때문에 컴포넌트 구현을 위한 내부 정보를 얻기 위해서는 역으로 방법론 각 단계들에서 제시하는 모델들을 추적하여 찾아야 하는 어려움이 있다.

이처럼 RUP, Catalysis, Advisor, MaRMI-III 방법론에서

제공하는 컴포넌트 관련 모델들은 UML을 기반으로 모델들을 표현하고 있으며 개발 공정의 여러 단계들에서 컴포넌트 개념을 나타내고 있다. 따라서 객체 지향 중심인 UML의 일반적인 단점을 가지게 되어, 내부 디자인의 규모가 객체지향 방법론의 중심이 되는 객체 혹은 클래스와 컴포넌트 사이에 큰 차이점을 보이게 되므로 정확한 컴포넌트의 설계 및 표현에 한계가 있다. 또한 컴포넌트의 기본적인 특성인 인터페이스에 대한 표현이 단순하고, 제한된 표현방법으로 그 뜻을 나타내기 때문에 여러 가지로 해석이 가능할 수 있는 모호한 점이 있다. 이들 방법론들에서는 개발 방법 각 단계마다의 여러 모델들로 분산되어 컴포넌트 개념을 기술하고 있기 때문에 컴포넌트 구현을 위한 내부 정보를 얻기 위해서는 컴포넌트 모델에서 역으로 방법론 각 단계들에서 제시하는 모델들을 추적하여 찾아야 한다. 이것은 컴포넌트 모델이 방법론의 한 단계로서 포함하고 있는 정보들이 추상적이기 때문이다. 또한 이와 같은 모델 추적은 방법론에 대한 기반 지식이 없는 개발자에게는 힘든 작업이며, 개발자가 모든 방법론을 숙지해야 한다.

컴포넌트 개발은 크게 컴포넌트 조립에 의한 개발과 컴포넌트 생성에 의한 개발로 이루어지며, 기존 모델에서는 개발되는 방법에 따른 조립과 생성을 위한 컴포넌트 모델의 구분 없이 표현한다. 때문에 컴포넌트 모델 정보의 전부가 컴포넌트를 조립하는 사람과 관련을 갖는 것은 아니며, 실제 이 모델을 기반으로 구현하려는 사람에게는 부족한 정보들이 많게 된다.

3. 사용 관점 중심의 컴포넌트 모델

3.1 조립자와 생성자 관점에서 필요한 정보

본 논문에서는 컴포넌트가 사용되는 역할에 따라 컴포넌트를 조립하여 개발하는 조립자 관점과 직접 컴포넌트를 구현하여 개발하는 생성자 관점이라는 두 가지 분리된 유형으로 컴포넌트 모델을 제안한다. 그리고 모델 추적이라는 힘든 작업을 해결 하고자 방법론 과정 중 한 단계에서의 모델이 아닌 독립된 컴포넌트 모델을 제시하여, 제안하는 모델들이 조립에 의한 컴포넌트 개발과 생성에 의한 컴포넌트 개발에 실질적으로 사용될 수 있도록 한다.

이를 위하여 우선 컴포넌트 조립자를 위한 정보와 생성자를 위한 정보를 표 1과 같이 분류하여 정의한다. 조립자를 위한 정보는 컴포넌트에서 제공하는 서비스와 직접 호출하여 사용할 수 있는 컴포넌트 인터페이스 정보를 나타내기 위한 정보를 나타낸다. 생성자를 위한 정보는 개발하려는 컴포넌트의 전체적인 구조와 컴포넌트들 간의 상호작용 정보 그리고 해당 컴포넌트 인터페이스에 대한 내부 상세 정보로서 실제 컴포넌트의 인터페이스를 구현하기 위한 오퍼레이션들과 속성들, 그들 타입간의 관계 정보를 나타낸다.

3.2 조립자 관점의 컴포넌트 모델

컴포넌트 조립(Component Assembly)은 컴포넌트 기반 소프트웨어 개발의 궁극적 목표로서 소프트웨어를 구성하는 단위 컴포넌트들의 조립을 통하여 개발비용이 절감되고 유지보수성이 뛰어난 소프트웨어를 개발할 수 있게 된다. 컴포넌트간의 조립을 위해서 컴포넌트는 인터페이스를 기반으로 결합되며, 이 인터페이스는 정보 은닉(Information Hiding) 원리에 따라 컴포넌트를 자유롭게 대체하고 독립적으로 사용할 수 있도록 지원한다.

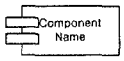
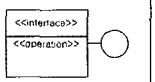
조립자 관점의 컴포넌트 모델에서는 컴포넌트에서 제공하는 서비스를 직접 호출하여 사용할 수 있도록 컴포넌트 인터페이스 정보를 표현하기 위해 표 2와 같이 UML을 확장하며 정의된 모델은 그림 4와 같다.

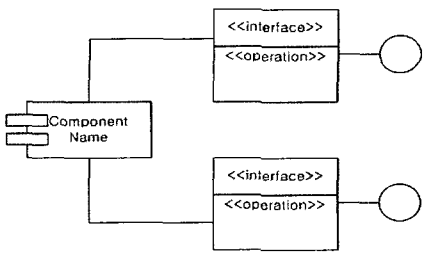
이 모델은 해당 컴포넌트가 어떤 컴포넌트와 관련 되어 수행되는지 알 필요 없이 단순히 컴포넌트를 호출해서 사용 가능 하도록 조립자가 알아야 하는 컴포넌트 이름과 인터페이스 정보만을 나타낸다. 이것은 자세한 비즈니스 로직은 숨긴 채 컴포넌트 인터페이스를 통한 서비스를 가능하게 한다.

〈표 1〉 사용관점에 따라 필요한 컴포넌트 정보

항 목	생성자	조립자
컴포넌트 이름	○	○
컴포넌트 간의 관계	○	×
인터페이스 이름	○	○
인터페이스의 오퍼레이션 이름	○	○
인터페이스의 오퍼레이션 인자타입	○	○
인터페이스의 오퍼레이션 리턴타입	○	○
인터페이스의 속성 타입	○	×
인터페이스의 외부 인터페이스와의 관계	○	×
인터페이스의 내부 타입(객체) 이름	○	×
타입 간의 관계	○	×
타입의 오퍼레이션 이름	○	×
타입의 오퍼레이션 인자타입	○	×
타입의 오퍼레이션 리턴타입	○	×
타입의 속성 타입	○	×

〈표 2〉 조립자 관점의 컴포넌트 모델을 위한 UML확장 표기법

UML 확장	포함 정보	의 미
	컴포넌트 이름	사용할 컴포넌트
	인터페이스 이름 오퍼레이션 이름 인자타입 리턴타입	컴포넌트 사용 시 필요한 인터페이스의 오퍼레이션 목록으로 해당 컴포넌트에서 제공하는 서비스



(그림 4) 조립자 관점의 컴포넌트 모델

3.3 생성자 관점의 컴포넌트 모델

생성자 관점의 컴포넌트 모델은 순수 구현을 위한 개발자들에게 필요한 많은 정보들을 나타낸다. 컴포넌트의 인터페이스 구현물이 자신의 책임을 완수하기 위해 다른 컴포넌트와 어떤 상호작용을 하는지, 컴포넌트에서 제공하는 서비스의 내부 비즈니스 로직이 어떠하며 내부 로직들 간의 관계가 어떠한지, 내부적으로 사용되는 정보들은 어떠한지에 관해서 표현한다. 이처럼 실제 생성자에게 필요한 모든 정보를 포함하기 위해 컴포넌트 상호작용 모델과 인터페이스 내부 모델 두 단계로 구분하며, 이 두 단계의 모델에 구체적으로 나타냄으로써 실제 구현 시 모델의 이용 효율을 높인다.

(1) 컴포넌트 상호작용 모델

컴포넌트 상호작용 모델은 생성자를 위한 1단계 모델로서 개발하려는 컴포넌트의 전체적인 구조와 컴포넌트들 간의 상호작용 정보를 나타낸다. 이를 위해 표 3과 같이 UML을 확장하며 정의된 모델은 그림 5와 같다.

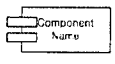
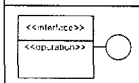
(2) 인터페이스 내부 모델

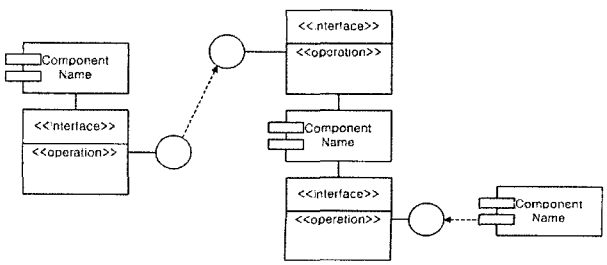
생성자를 위한 2단계 모델인 인터페이스 내부 모델은 해당 컴포넌트의 하나의 인터페이스에 대한 내부 상세 모델로서 실제 컴포넌트의 인터페이스를 구현하기 위한 오퍼레이션들과 속성들, 그들 타입간의 관계 정보를 나타낸다. 이를 위해 표 4와 같이 UML을 확장하며 정의된 모델은 그림 6과 같다.

3.4 기존 컴포넌트 모델과의 비교

컴포넌트 모델은 그 모델을 기반으로 하여 소프트웨어를 개발하려는 사람들에게 쉽게 접근가능하며 구체적이고 실질적인 모델이 되어야 한다. 그러나 기존의 컴포넌트 모델은 여러 모델들로 분산되어 컴포넌트 개념을 기술하고 있으며, 그 정보들 또한 추상적이다. 본 논문에서 제안한 컴포넌트 모델은 컴포넌트를 설계하고 직접 개발하는 생성자 관점의 두 개의 모델과, 컴포넌트를 조립하여 소프트웨어를 개발하는 조립자 관점의 단일화된 모델을 제안하였다. 조립자 관점의 모델은 컴포넌트가 제공하는 인터페이스 정보만으로 컴포넌트를 조립하여 소프트웨어를 개발하므로 컴포넌트 내부 비즈니스를 숨길 수 있다는 장점을 가진다. 이는 컴포넌트 내부 설계상의 변화에 컴포넌트 조립자는 쉽게 대응할 수 있으며 소프트웨어의 변경이 수반되지 않는다는 것을 의미한다.

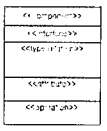
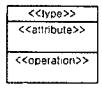
〈표 3〉 컴포넌트 상호작용 모델을 위한 UML 확장 표기법

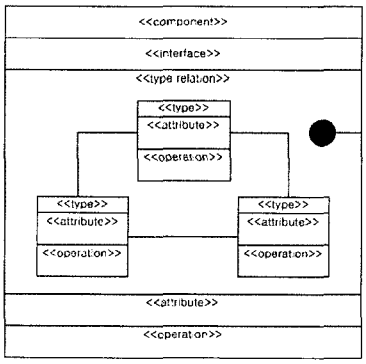
UML 확장	포함 정보	의 미
	컴포넌트 이름	사용할 컴포넌트
	인터페이스 이름 오퍼레이션의 이름 인자타입/리턴타입	컴포넌트 사용 시 필요한 인터페이스의 오퍼레이션 목록으로 해당 컴포넌트에서 제공하는 서비스
		화살표의 출발지 컴포넌트에서 목적지의 컴포넌트를 필요로 함을 의미



(그림 5) 컴포넌트 상호 작용 모델

〈표 4〉 인터페이스 내부 모델을 위한 UML 확장 표기법

UML 확장	포함 정보	의 미
	컴포넌트 이름/인터페이스 이름/속성정보/ 오퍼레이션 이름/인자타입/리턴타입/내부 타입 관계도	해당 컴포넌트의 하나의 인터페이스에 대한 상세 정보를 나타내는 기본 틀로서 컴포넌트 생성자에게 필요한 인터페이스 내부 정보와 로직 간의 타입 관계도를 포함함.
	타입 이름/타입에 대한 속성정보/오퍼레이션의 이름/인자타입/리턴타입	특정 인터페이스 내에서 내부적으로 처리되는 로직에 관한 정보로서 클래스보다 높은 추상화 단계인 타입을 의미함
	컴포넌트 이름/인터페이스 이름/반복 표시자(1, *)	관련 있는 외부 인터페이스와의 관계를 지정함. 동일 컴포넌트 내의 다른 인터페이스이거나 다른 컴포넌트의 인터페이스일 수 있음
	반복 표시자(1, *)	타입 관계도에서 타입간의 연관관계가 있음을 의미



(그림 6) 인터페이스 내부 모델넌트 상호 작용 모델

〈표 5〉 기존 컴포넌트 모델과의 비교

컴포넌트 모델	특 징
기존의 컴포넌트 모델	컴포넌트 모델의 분산 컴포넌트 내부 비즈니스 로직의 분산 모델 역추적을 통한 컴포넌트 구현 컴포넌트 조립을 위한 모델 부재
제안한 컴포넌트 모델	생성자와 조립자 관점의 모델 독립된 생성자 관점의 컴포넌트 모델 정의 단일화되고 독립된 모델로 컴포넌트 구현 컴포넌트 조립자를 위한 모델 정의 컴포넌트 비즈니스 로직 변경에 쉽게 대응

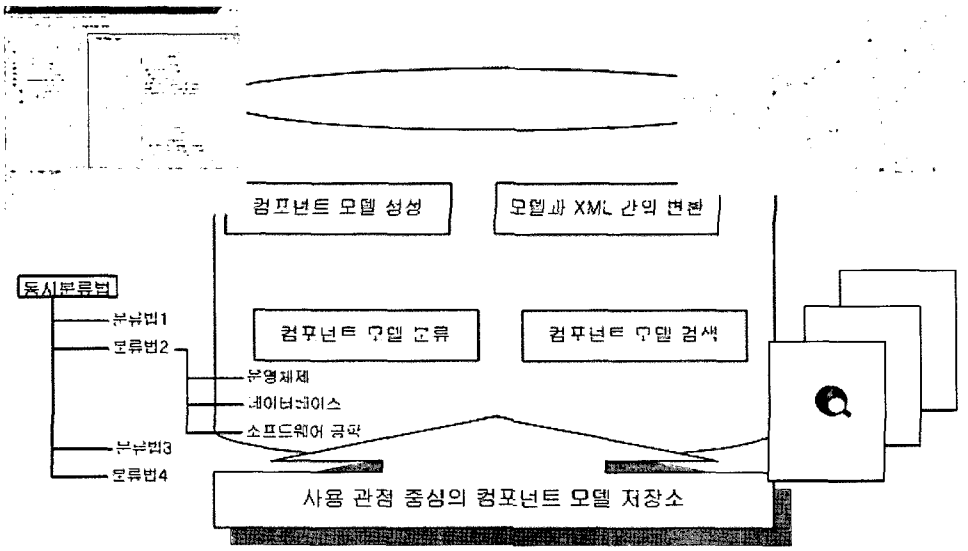
4. 적용사례

본 장에서는 제안한 사용 관점에 따른 컴포넌트 모델을 생성하고, 저장 관리하며 모델간의 상호변환을 위한 자동변환 툴을 자바언어를 사용하여 개발하고 적용해 본다(그림 7). 이를 위하여 본 논문에서 제안한 확장된 UML을 이용하여 컴포넌트 생성자 관점의 모델을 생성하고, 생성된 모델은 XML로 변환하여 저장되며, XML문서를 분석하고 파싱하여 컴포넌트 조립자 관점의 모델을 자동 생성한다. 그림 8과 그림 9는 각각 생성자 관점과 조립자 관점 모델에 대한 XML스키마를 표현하고 있다. XML문서와 모델간의 상호변환은 DOM파서와 JHotDraw[16]를 이용하여 개발한다. JHotDraw에서 그려지는 모든 다이어그램들은 그들의 정보를 직렬화(serialization)하여 본 논문에서 제안한 XML스키마 구조에 따라 XML문서로 저장한다. 그림 10은 이와 같은 과정을 수행하는 소스코드의 일부분이다.

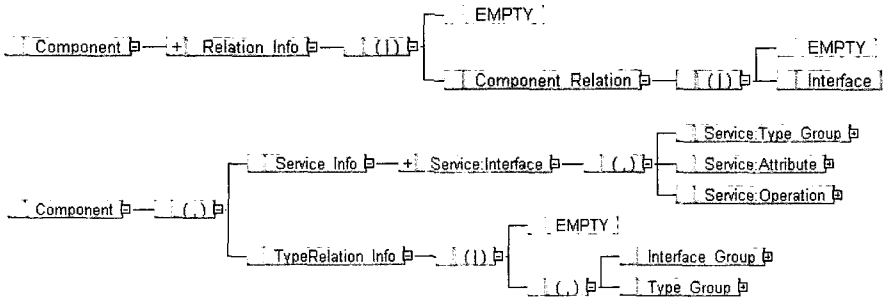
제안한 컴포넌트 모델을 전자상거래 시스템의 상품구매시스템 중 구매컴포넌트를 선정하여 실제 적용해 보며 다음과 같은 단계로 진행한다.

1단계 : 구현하고자 하는 컴포넌트의 생성자 관점의 상호작용 모델 작성 (그림 11)
2단계 : 구현하고자 하는 컴포넌트의 생성자 관점의 인터페이스 모델 작성 (그림 12)
3단계 : 그림 11, 12로부터 자동 생성되어진 컴포넌트 조립자 관점의 모델 (그림 13)

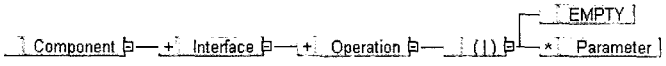
구매 컴포넌트는 고객이 상품을 열람하고 주문하는 과정을 표현한 컴포넌트로서 본 논문에선 제안한 생성자 관점의 모델 중 상호작용 모델로 표현하면 그림 11과 같다. 이는 컴포넌트 조립에 필요한 인터페이스들, 인터페이스의 종류 그리고 컴포넌트 상호작용에 대해 표현하고 있으며, 고객컴포넌트(CustomerMgr)와 상품컴포넌트(ProductMgr)가 구매컴포넌트(PurchaseMgr)와 각각 어떻게 상호작용하고 있는지를 보



(그림 7) 컴포넌트 모델 저장소의 전체 구성도



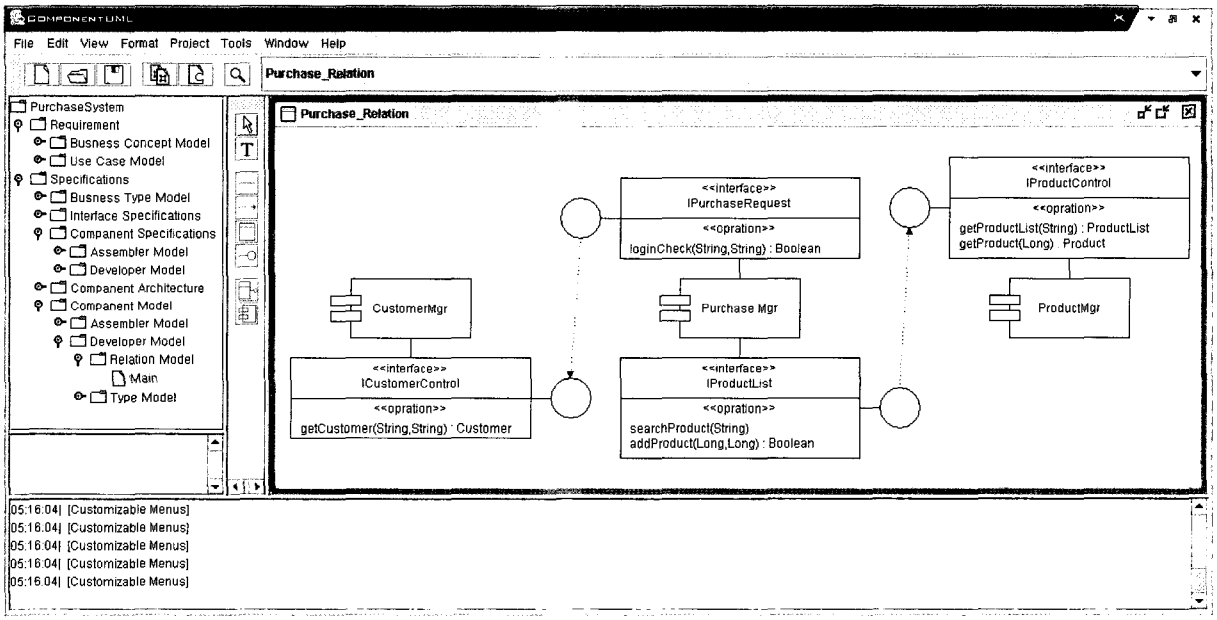
(그림 8) 컴포넌트 상호작용모델(위), 인터페이스 내부 모델(아래)



(그림 9) 컴포넌트 인터페이스 모델

```
m_XML = new XMLCreate("Component");
...
if (m_XML.appendElement("Service_Info")) {
   if (vt.size() > 0) {
      /* Interface */
      CInterfaceModel iModel = null;
      for(int t=0 ; t<vt.size() ; t++) {
         iModel = (CInterfaceModel)vt.get(t);

         m_XML.appendElement("Service:Interface");
         m_XML.appendAttribute("ID", iModel.getInterfaceID());
         m_XML.appendAttribute("Name", iModel.getName());

         m_XML.appendElement("Service:Type_Group");

         Vector vt2 = m_objControl[2].getObjRefType(iModel.getInterfaceID());

         if(vt2.size() > 0) {
            /* Type */
            CTypeModel tModel = null;
            for(int w=0 ; w<vt2.size() ; w++) {
               tModel = (CTypeModel)vt2.get(w);

               m_XML.appendElement("Service:Type");
               m_XML.appendAttribute("ID", tModel.getTypeID());
               m_XML.appendAttribute("Name", tModel.getName());
               m_XML.appendElement("Service:Attribute_Group");
```

(그림 10) 모델을 XML문서로 변환하는 자바 소스코드

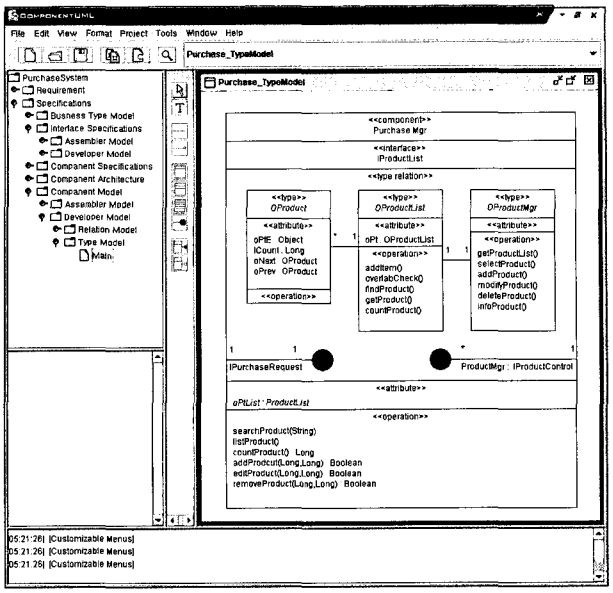


(그림 11) 생성자 관점의 컴포넌트 상호작용 모델

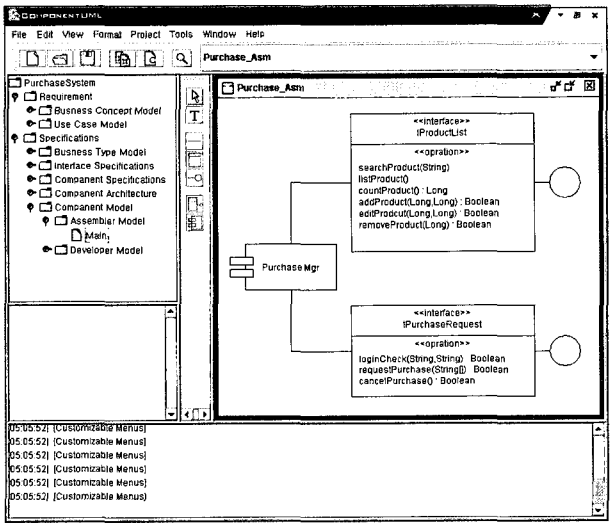
여주고 있다. 그림 11은 생성자 관점의 컴포넌트 상호작용 모델로서 주변의 어떤 컴포넌트와 상호작용하고 있는지를 정확히 파악할 수 있고, 인터페이스에 포함된 오퍼레이션으로 컴포넌트의 조립이 어떻게 이루어지는지 명시하고 있다. 이는 하나의 컴포넌트가 다른 컴포넌트의 어떠한 인터페이스를 필요로 하는지와 어떠한 인터페이스를 제공하는지를 결정한다. 이러한 모델은 해당 컴포넌트가 시스템의 어디에 위치해 있으며 시스템의 전체 구성을 파악하는데 도움이 된다.

그림 12는 그림 11에 있는 구매컴포넌트(PurchaseMgr)에 대한 인터페이스 내부모델로서 본 장에서는 구매컴포넌트(PurchaseMgr)가 가지는 2개의 인터페이스(IPurchageRequest, IProductList) 중 IProductList 인터페이스에 대한 인터페이스 내부 모델이다. 이는 실제 구현에 필요한 구체적인 정보로서 다른 컴포넌트에서 사용이 가능한 오퍼레이션(operation)들과 속성(attribute)들, 내부 로직을 처리하는 타입(type)들과 그들 간의 타입 관계(type relation)를 포함한다. 그리고 컴포넌트들간의 상호작용을 위한 오퍼레이션뿐만 아니라 해당 컴포넌트가 제공되어야 하는 모든 인터페이스에 대한 오퍼레이션을 표현한다. 따라서 그림 12에서는 그림 11에서 표현된 컴포넌트 상호작용을 위한 오퍼레이션에 컴포넌트가 서비스해야 하는 추가적인 오퍼레이션도 표현되어 있다. 이는 컴포넌트를 직접 개발하는 생성자에게 아주 유용한 모델이며 컴포넌트의 내부가 어떻게 구성되어 기능을 수행하는지 알 수 있다.

그림 13은 생성자 관점의 컴포넌트 모델 정보를 기반으로 하여 자동으로 변환된 구매컴포넌트(PurchaseMgr)의 조립자 관점의 모델이다. 본 적용사례에서 선정된 구매컴포넌트



(그림 12) 생성자 관점의 인터페이스 내부 모델



(그림 13) 조립자 관점의 컴포넌트 모델

(PurchaseMgr)는 2개의 인터페이스를 가지고 있으며, 각각의 인터페이스가 제공하는 오퍼레이션들의 목록으로 구성되어 있다. 이러한 오퍼레이션은 구매컴포넌트(PurchaseMgr)가 제공하는 서비스에 대한 오퍼레이션과 다른 컴포넌트들과의 상호작용에 의하여 제공되는 오퍼레이션으로 구분된다. 그러나 조립자 관점에서는 이 2가지의 오퍼레이션을 구분할 필요가 없으며 어떻게 동작하는지 알 필요도 없다. 예로, 그림 13에 있는 IProductList 인터페이스에 있는 searchProduct(String) 오퍼레이션은 구매컴포넌트(PurchaseMgr)가 제공하는 서비스이지만 상품컴포넌트(ProductMgr)와 결합되어 제공되는 서비스이다. 하지만 이미 개발된 구매컴포넌트(PurchaseMgr)를 조립하여 시스템을 구축하고자 하는 조립자 관점에서는 상품컴포넌트(ProductMgr)와의 상호작용을 모른채 searchProduct(String) 오퍼레이션을 이용하여 조립한다.

조립자 관점의 컴포넌트 모델은 조립자 관점에서 필요한 컴포넌트가 제공하는 서비스에만 초점을 두었으며, 컴포넌트 내부의 자칫 복잡해지기 쉬운 컴포넌트 모델을 생략하여 조립에만 집중할 수 있도록 하였다.

이와 같이 생성자 관점의 컴포넌트 모델을 2가지로 구분함으로써 컴포넌트의 상호작용 모델과 인터페이스 내부 모델을 하나로 표현할 경우의 단점인 컴포넌트 모델의 복잡도나 시스템의 가독성 문제를 해결할 수 있다. 또한 컴포넌트 개발자는 단일한 컴포넌트를 개발하기 위해서는 인터페이스 내부 모델만으로 컴포넌트를 개발할 수 있다. 조립자 관점의 모델은 컴포넌트가 제공하는 기능, 즉 서비스에 초점을 두어 조립자 관점에서 필요한 정보들만으로 구성함으로써 모델을 단순화 하였다.

5. 결론 및 향후연구

본 연구에서는 RUP, Catalysis, Advisor, MaRMI-III 개발 방법론에서 나타내는 기존 컴포넌트 관련 모델들의 문제점을 분석하여, 이에 대한 해결 방안으로서 사용 관점 중심의 컴포넌트 모델을 제안하였다. 그리고 본 연구에서 개발한 컴포넌트 모델 저장소의 모델생성과 자동변환 과정을 이용하여 적용해 보았다.

기존 방법론들에서는 개발 방법 각 단계마다 여러 모델들로 분산되어 컴포넌트 개념을 기술하고 있기 때문에 컴포넌트 내부 정보를 얻기 위해서는 방법론 각 단계들에서 제시하는 모델들을 역 추적하여 찾아야 한다. 이것은 컴포넌트 모델이 방법론의 한 단계로서 포함하고 있는 정보들이 추상적이기 때문이며, 모델 역 추적은 방법론에 대한 기반 지식이 없는 개발자에게는 힘든 작업이다. 또한 컴포넌트 개발은 크게 컴포넌트 조립에 의한 개발과 컴포넌트 생성에 의한 개발로 이루어지는 반면, 기존 모델에서는 개발되는 방법에 따른 조립과 생성을 위한 컴포넌트 모델의 구분 없이 표현하기 때문에 그 컴포넌트 모델 정보의 전부가 컴포넌트를 조립하는 사람과 관련을 갖는 것은 아니며, 또한 실제 이 모델을 기반으로 구현하려는 사람에게는 부족한 정보들이 많게 되었다.

이에 본 논문에서 컴포넌트가 사용되는 역할에 따라 컴포넌트를 조립하여 개발하는 조립자 관점과 직접 컴포넌트를 구현하여 개발하는 생성자 관점의 이 두 가지 분리된 유형으로 컴포넌트 모델을 제안했다. 그리고 모델 추적이라는 힘든 작업을 해결 하고자 방법론에서 제공하는 하나의 단계에서의 모델이 아닌 독립된 컴포넌트 모델을 제안하여, 이 모델만을 이용하여 조립에 의한 컴포넌트 개발과 생성에 의한 컴포넌트 개발에 실질적으로 사용될 수 있도록 하였다. 이를 통해 조립자는 컴포넌트 내부의 자세한 비즈니스 로직은 숨긴 채 컴포넌트 인터페이스를 통한 서비스가 가능하게 되었으며, 생성자는 필요한 모든 정보를 두 단계의 모델로 구체적으로 나타냄으로써 실제 구현시 모델의 이용 효율을 높였다. 이렇게 분리된 두 가지 유형으로 정의함으로써 생성자 계약조건에 대한 변경이 조립자 계약의 변경을 수반하지 않게 되므로 조립자에게는 어떠한 영향도 주지 않기 때문에 컴포넌트 변경에 쉽게 대응할 수 있다. 또한 생성자 관점에서 생성된 컴포넌트 모델은 컴포넌트 조립자의 다양한 요구에 맞도록 자동 변환되므로 시스템 개발 비용이 절감되는 효과를 가져 올 수 있다.

본 논문에서 제안하는 컴포넌트 모델에서는 모델이 사용되는 관점에 따라 컴포넌트의 정보들을 적합하게 구분 하였으나 생성자 관점에서의 컴포넌트 식별과 추출 방법에 관한 사항이 미비하다. 향후 현재 개발되어 있는 컴포넌트 모델 저장소 및 변환 시스템에 컴포넌트 식별과 추출방법을 추가하여 연구가 계속 진행되어야 하겠다.

참 고 문 헌

[1] Dedmond F. D'Souza, Alan C. Wills,"Object, Component and Frameworks With UML", ADDISON-WESLEY, 1998.

[2] Pathbot, "소프트웨어 컴포넌트 소개",http://my.dreamwiz.com/coup/cp/intr02.html.

[3] 김창완, 객체지향 모델링과 구현, 대림, 1998.

[4] 김수동, "컴포넌트 정의 및 관련 기술 동향", 소프트웨어 공학회지 제12권 제3호 pp.5-18, 1999.

[5] 김태웅, 김경민, 김태공, "인터페이스 명세기반 컴포넌트 저장소 모델", 한국정보과학회 춘계학술발표 논문집, 제29권 제2호, pp.109-111, 2002.

[6] 김경민, 김태웅, "XML기반 인터페이스 명세 중심의 컴포넌트 저장소 모델", 한국정보처리학회 추계학술발표 논문집, 제9권 제2호, pp.2083-2086, 2002.

[7] Kruchten, P. B.,"Rational Unified Process, The: An Introduction, Second Edition", Addison-Wesley, 2000.

[8] 전상욱, 김인규, 김정윤, 윤경아, 배두환, "CBD 방법론 비교 분석", 한국정보처리학회지 Vol.10 No.03 pp.30-39, 2003.

[9] W. Kozaczynski, "Composite Nature of Component", In Proceedings of the 1999 International Workshop on Component Based Software Engineering, pp.73-77, May 1999.

[10] D'Souza, D. F., and A. C. Wills,"Object, Components, and Frameworks with UML: The Catalysis Approach", Addison-Wesley, 1998.

[11] Sterling, "The CBD96 Standard Versin 2.1", Sterling Software, July 1998.

[12] Harmon, p.,"Visual Modeling Tools, Case Vendors, and Component Methods", Component Development Strategies, pp.5-18, 1999.

[13] Dodd, J., et al., "Advisor 2.04", Sterling Software, 1999.

[14] "MaRMI 방법론", 한국 전자 통신 연구원, http://www.component.or.kr

[15] "MaRMI-III 1.0", ETRI, 2001, http://www.component.or.kr

[16] Erich Gamma, Thomas Eggenschwiler, "JHotDraw as Open-Source Project", JHotDraw, October 2004.

김 태 웅

e-mail : twkim@cs.inje.ac.kr

1994년 인제대학교 전산학과(이학사)

1998년 인제대학교 전산학과(이학석사)

2002년 인제대학교 전산학과(박사과정 수료)

1999년 ~ 2006년 동의과학대학 겸임교수

2007년 ~ 현재 인제대학교 컴퓨터 공학부 유비쿼터스 지능형 홈 인력 양성사업단 계약교수

관심분야 : 소프트웨어 공학, 영역지향 프로그래밍, 소프트웨어 개발 방법론, 모델 리팩토링

김 경 민

e-mail : kmkim@cs.inje.ac.kr

2002년 인제대학교 컴퓨터공학부 졸업(학사)

2004년 인제대학교 일반 대학원 전산학과 졸업(이학석사)

2006년 인제대학교 일반 대학원 전산학과(박사 수료)

관심분야 : Model Transformation, Code Generation, Refactoring, AOSD

김 태 공

e-mail : ktg@cs.inje.ac.kr

1983년 서울대학교 계산통계학과(학사)

1985년 서울대학교대학원 계산통계학과 전산과학전공(이학석사)

1994년 서울대학교대학원 계산통계학과 전산과학전공(이학박사)

2003년 ~ 2004년 The University of Texas at Dallas 방문 교수

1990년 ~ 현재 인제대학교 컴퓨터공학부 교수

관심분야: 소프트웨어 공학, Model Engineering, AOSD, Generative Programming