한국산학기술학회논문지
Vol. 9, No. 6, pp. 1678-1689, 2008

객체지향 환경에서 소프트웨어 생산성 향상을 위한 프레임워크 모델

김영규[1], 양해술[2], 최형진[3*]

Framework Model for Software Productivity Enhancement In Object-Oriented Environment

Young-Gyu Kim[1], Hae-Sool Yang[2] and Hyung-Jin Choi[3*]

요 약 최근 소프트웨어 개발에 적용하기 시작한 객체지향 방법(OOM:Object-Oriented Method)은 독립적인 소프트웨어의 재사용을 통한 개발 비용과 시간의 단축을 강조하고 있다. 그러나 개발 기술에 대한 지식 부족과 확장성 및 성능을 배제한 설계로 많은 문제점이 나타나고 있다. 따라서 본 논문에서는 소프트웨어 개발 생명주기에서 소프트웨어 생산성 향상을 위한 효율적인 객체지향 모델링 방법을 제안하고자 한다. 제안 방법은 Use Case 모델링, 분석 모델링, 그리고 설계 모델링 방법들을 포함하고 있다. 제안한 프레임워크 모델은 다음과 같은 특징이 있다. 첫째, 효율적인 객체지향 표준 개발모델 제안, 둘째, 소프트웨어 개발 및 유지보수 비용의 절감, 셋째, 신기술 적용에 따른 프로젝트의 불확실한 문제 해결 등이다.

Abstract Recently, OOM(Object-Oriented Method) access method which begins to apply to software development is emphasizing development cost and time reduction through independent software reuse. But because of planning to the exclusion of the lack of knowledge and expansion or performance for development technology many problems are coming out. Therefore, this thesis suggests Effective Object-Oriented Modeling methods considering the plan for higher productivity enhancement of software in the software development life cycle. Effective Object-Oriented Modeling method includes, methods, Use Case modeling methods, Analysis Modeling methods, Design Modeling methods and others. This thesis suggests Framework model method read in characteristics as follows. First, suggestion about effective object-oriented standard model development, Second, retrenchment of software development or maintenance cost, Third, solution about uncertain problems of projects in line with the application of new technology.

Key Words : Object-Oriented Method, Use Case Modeling, Analysis Modeling, Design Modeling

1. 서론

최근에는 객체지향 방법론 및 CBD(Component Based Development) 방법을 적용한 대규모 정보 시스템 개발 프로젝트가 증가하고 있으며, 이러한 프로젝트에 적용되는 방법론은 기존의 사상, 기법, 도구와는 전혀 다른 접근 방법을 요구하는 새로운 개발 방법론이라 할 수 있다. 그러므로 프로젝트의 성격 및 관리에 있어 다음과 같은 중대한 변화를 초래한다[1][2][3].

첫째, 소프트웨어 개발 및 유지보수 비용이 증가된다. 애플리케이션의 복잡성의 증가에 비해 소프트웨어 개발에 사용되는 CASE 도구는 상대적으로 빈약한 발전으로 개발 및 유지보수의 비용이 증가되었다. 둘째, 신기술의 적용에 따른 프로젝트의 불확실성이 증대된다. 즉, 전혀 다른 새로운 접근 방법을 사용하므로 프로젝트 수행과 관련한 위험과 불확실성이 크게 증대될 가능성이 항상 존재한다. 셋째, 투자규모의 대형화에 따른 프로젝트의 효율적 관리가 필요하다. 객체지향 방법론을 적용한 시스템 개발 프로젝트는 기존의 방식으로는 감당하기 어려운

[1]한림성심대학 인터넷비즈니스과
[2]호서대학교 벤처전문대학원
[3]강원대학교 컴퓨터과학과
*교신저자: 최형진(choihj@kangwon.ac.kr)
접수일 08년 11월 10일 수정일 (1차 08년 12월 01일, 2차 08년 12월 12일) 게재확정일 08년 12월 16일

새로운 내용과 기능이 필요하므로 프로젝트 관리에 어려움이 따른다. 또한 이러한 프로젝트는 대규모이므로 보다 체계적이고 효율적 관리의 필요성이 요구되어진다.

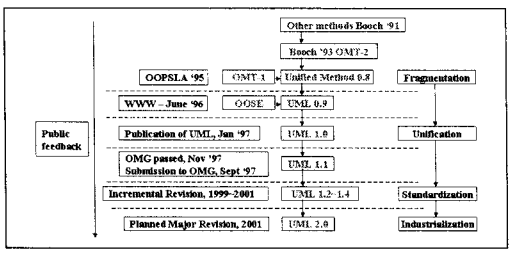
따라서 본 논문에서는 객체지향 환경의 소프트웨어 개발 생명주기에서 소프트웨어 생산성 향상을 위한 계획, 사용자 요구 분석, 설계, 품질관리 방법을 고려한 효율적인 객체지향 모델링 프레임워크 모델을 다음과 같이 제안한다. 첫째, Use Case 모델링 방법, 둘째, 분석 모델링 방법, 셋째, 설계 모델링 방법 등이다.

2. 관련 연구

이 장에서는 관련 연구로서 UML, CBD 방법론, 설계 패턴에 대하여 기술한다[1][4][5].

2.1 UML

UML(Unified Modeling Language)은 시각적인 언어로서, 객체지향 시스템을 분석하고 설계하는 사람들이 소프트웨어 시스템의 가공물을 가시화, 구축, 문서화하고, 그 시스템을 사용하는 조직의 업무를 모델링하는 방법을 제공한다.



[그림 1] UML의 역사

Rational Software사와 OMG에서는 객체지향 도해 표기법 요소들과 다른 표기법의 새로운 측면들을 엮어서 소프트웨어 개발 업계에 표준 모델링 언어를 발표하였다.

그러나 UML은 [그림 1]과 같이 아직도 표준으로서 진화중이며, 현재의 버전 2.0도 계속 변화할 것이다[5][6][7].

2.2 CBD 방법론

CBD 방법은 재사용 가능한 소프트웨어 모듈을 제작하여 기계 부품과 같이 이를 조립하여 보다 복잡한 소프트웨어를 만드는 방식을 말한다. 즉, CBD는 컴포넌트라는 소프트웨어의 복잡성과 생산성 문제를 해결하고자 하는 일종의 새로운 개발 패러다임을 제시한 것이다. 대다수의 개발 프로젝트는 공통된 기능을 하는 모듈이 있으며, 재사용으로 부가가치를 획득할 수도 있기 때문이다. CBD 방법의 특징은 다음과 같다. 첫째, 소프트웨어 전반의 개발 라이프 사이클에 대한 모든 측면과 절차를 컴포넌트를 기반으로 접근하는 개발 방법론이다. 둘째, 컴포넌트 자체를 개발하며, Java나 C++같은 객체지향 언어를 사용하여 컴포넌트 특성을 가진 시스템을 개발한다. 셋째, 개발된 컴포넌트를 기반으로 시스템을 개발하며, EJB, .Net 플랫폼 기반에서 각종 상용 컴포넌트를 이용하여 시스템을 개발한다. 넷째, 재사용 가능, 유지보수 용이, 개발확장 용이 등의 장점이 있다[8][9].

2.3 설계 패턴

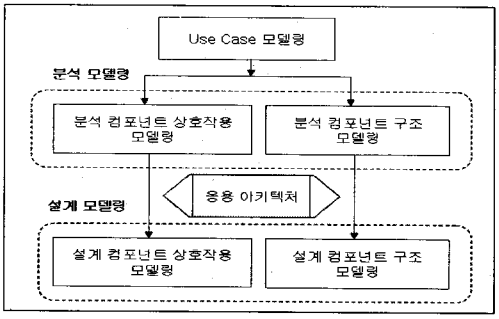
소프트웨어를 개발할 때에 패턴을 적용하는 효과는 먼저 기존의 성공적인 설계와 구조를 쉽게 재사용할 수 있다. 또한 검증된 기술을 패턴으로 표현함으로써 새로운 시스템의 개발자들에게 패턴을 좀 더 활용할 수 있는 여러 가지 결정들을 선택할 수 있도록 도움을 준다. 클래스와 객체 상호작용 및 그들이 담고 있는 의미의 명시적인 명세를 제공함으로써 기존의 시스템들의 문서화와 유지보수를 증가할 수도 있다. 또한 소프트웨어 개발의 공통적인 어휘를 제공함으로써 개발자가 빠르고 정확한 설계를 할 수 있도록 도와주고, 소프트웨어 개발팀 내부나 다른 팀과의 분명한 의사전달을 가능하게 하며 새로운 개발자들에게는 교육의 역할도 하게 된다[4]. 최근에는 객체지향 방법론과 CBD 방법을 적용한 대규모 정보 시스템 개발 프로젝트가 증가하고 있다. 이러한 프로젝트에 적용되는 방법론은 기존의 사상, 기법, 도구와는 전혀 다른 접근방법을 요구하는 새로운 개발방법론이라 할 수 있다.

3. 효율적인 객체지향 프레임워크 모델

이 장에서는 객체지향 환경에서 소프트웨어 생산성 향상을 위한 효율적인 객체지향 Use Case 모델링 방법을 제시한다[10][11][12].

3.1 프레임워크 모델의 요구사항

본 논문에서는 다음 [그림 2]와 같은 객체지향 환경의 소프트웨어 개발 생명주기에서 생산성 향상을 위한 사용자 요구분석에 따른 Use Case 모델링 방법, 분석 모델링 방법, 설계 모델링 방법을 제안한다.



[그림 2] 모델링 진행 절차

3.2 Use Case 모델링 방법

3.2.1 모델링 진행 절차

본 논문에서 제안하는 분석 및 설계 방법은 [그림 2]와 같다. 즉 Use Case 모델링, 분석 모델링으로 진행한다. 또한 모델링의 대상에 따라 애플리케이션을 위한 객체지향 모델링과 관계형 데이터베이스에서 데이터를 효율적으로 저장관리하기 위한 데이터 모델링을 두 부분으로 나누어서 진행한다.

3.2.2 Use Case 모델링

Use Case 모델링은 시스템을 사용하는 사용자 및 연계가 필요한 외부 시스템과 시스템이 제공하는 기능 사이의 상호작용을 모델링하는 것이다. Use Case 모델링을 통해 나오는 산출물은 [표 1]과 같다.

[표 1] Use Case 모델링 산출물 목록

산출물 명	설 명
Use Case 다이어그램	Actor 상호간의 관계, Actor와 시스템이 제공하는 기능사이의 관계 및 기능 상호간의 관계를 표현한 다이어그램
Actor 카탈로그	Actor의 역할을 정의한 문서
Use Case 명세서	시스템이 제공하는 기능의 세부적인 처리흐름을 기술한 문서

(1) Actor 도출

Actor는 시스템 외부에서 시스템과 상호 연관을 가진 사람이나 사물로서 시스템의 상태를 변경할 수 있는 모든 것이다. 사용자, 외부 하드웨어, 타 시스템 등이 Actor의 종류가 된다. 시스템과 인터페이스가 필요한 외부 개체는 시스템을 직접 사용하는 업무 담당자뿐만 아니라, 개발하는 시스템과 인터페이스가 필요한 타 시스템을 포함한다. Actor를 도출할 때에는 [표 2]와 같은 구성 요소를 고려해야 한다.

[표 2] Use Case 모델의 구성 요소

모델 구성 요소	설 명
	시스템 외부에서 시스템을 사용하는 사용자, 인터페이스가 필요한 타 시스템, 시스템 내부에 있으나 시스템을 구동시키기 위한 이벤트를 발생시키는 주체이다.
	시스템이 제공해야 하는 업무적인 기능을 의미한다. Use Case를 도출하는데 중요한 사항은 데이터베이스에 데이터를 입력, 수정, 삭제, 조회하는 단위로 Use Case를 도출하지 말아야 한다.
	Use Case 다이어그램에서 관계는 Actor와 Use Case, Actor 상호간의 관계 및 Use Case 상호간의 관계로 구분할 수 있다. Actor와 Use Case 사이의 관계는 일반적인 관계이며, 모든 Actor는 하나 이상의 Use Case와 관계를 가지게 된다.

- 문제정의 시에 면담의 대상이 되었던 업무 담당자를 Actor로 도출한다.
- 사람뿐만 아니라 시스템과 인터페이스가 필요한 소프트웨어, 하드웨어 및 외부 업무 시스템도 Actor로 도출한다.
- 시스템 외부에 존재해야 한다. 시스템 내부에 존재하는 선행 업무 등은 Actor로 도출할 수 없다.
- Actor는 특정한 역할을 수행해야 한다.

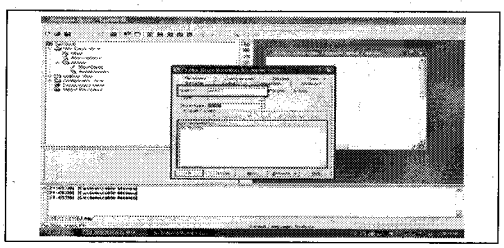
(2) Actor 카탈로그 작성

도출한 Actor에 대해서 [표 3]과 같은 Actor 카탈로그라는 문서를 작성해야 한다.

[표 3] Actor 카탈로그 작성 예

Actor 명	Actor 설명	구분
과정 담당자	새로운 과정을 개발하고 이를 관리하는 사람	사람
회계 시스템	교육비에 대한 정보를 넘겨주어야 하는 시스템	시스템

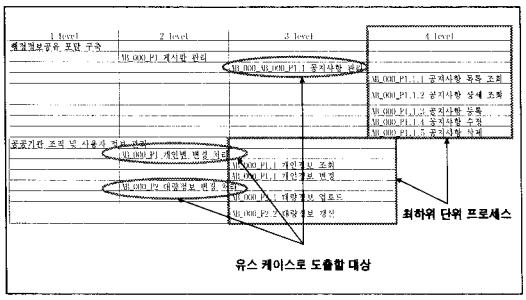
문서화 작업은 통합 모델러가 Rose로부터 자동생성하며 각 업무 분석 담당자는 [그림 3]과 같이 Rose에 Actor의 설명을 등록해주어야 한다.



[그림 3] Rose에 Actor의 설명을 등록하는 방법

(3) Use Case의 도출

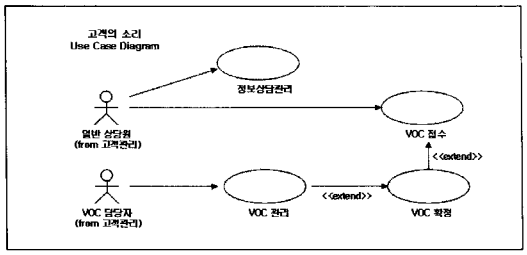
Use Case는 기능 분해도를 이용하여 추출하며 기능 분해도의 레벨이 각 업무별로 상이할 수 있으나, 최하위 레벨은 입력, 수정, 삭제, 조회의 단위로 이루어진다. 또한 Use Case는 최하위 레벨의 바로 윗 레벨 단위로 하나씩 생성하며, 하나의 Use Case는 입력, 수정, 삭제, 조회와 같은 최소 단위의 프로세스가 묶여진 단위로 작성한다. [그림 4]는 Use Case를 도출한 예이다.



[그림 4] Use Case 도출 예

(4) Use Case 다이어그램 작성

Actor와 Use Case를 도출한 뒤에는 이를 다이어그램의 형태로 표현해야 한다. [그림 5]는 Use Case 다이어그램에서 하나의 Actor와 관련이 있는 모든 Use Case 또는 하나의 Use Case와 관련있는 모든 Actor들 사이의 연관관계를 표현한 다이어그램의 예이다.



[그림 5] Use Case 다이어그램의 예

(5) Use Case 명세서 작성

Actor가 업무를 처리하기 위해서 Use Case를 사용하면 일정한 작업 흐름이 발생하게 된다. Actor가 동일한 업무를 수행하더라도 다양한 작업 흐름이 발생할 수 있다.

Use Case 명세서에 다음과 같이 상세하게 기술되어야 한다.

- Use Case의 시나리오는 아래의 사항이 표현될 수 있도록 기술한다.
  - Use Case가 어떻게 시작되고 종료되는가?
  - Use Case가 어떤 Actor와 상호 작용하는가?
  - Use Case와 Actor가 어떤 데이터를 주고받는가?
  - Use Case가 어떻게, 언제 데이터를 시스템에 저장하는가?
- 예외사항이 발생하지 않은 기본 흐름은 반드시 기술하며, 예외사항에 따라 다양하게 분기되는 선택흐름은 분기시점을 반드시 기술 하여야 한다.
- 이벤트의 흐름을 순서대로 기술하고, 사용자 및 개발자가 이해하기 쉬운 용어사전에 등록된 도메인 용어를 사용한다.
- Use Case가 수행되면서 충족시켜야 할 요구사항을 기술하며, Use Case가 수행되기 위한 사전 및 사후조건을 정의한다.
- 복잡한 비즈니스 로직의 경우와 모든 Use Case에 공통으로 적용되는 사항은 별도의 문서로 작성될 수 있다.

[표 4]는 Use Case 명세서의 항목 설명을 나타내고 있다.

[표 4] Use Case 명세서 항목 설명

구성 항목	기술 방법
1. 개요	Use Case의 역할과 목적 및 개괄적인 내용에 대해서 기술한다.
2. 이벤트 흐름	Actor가 Use Case를 어떻게 시작 시키는지, Actor와 Use Case가 주고받는 정보는 무엇이지, Use Case가 어떻게 종료하는가 등을 Actor가 일으키는 이벤트와 거기에 대한 Use Case의 반응 중심으로 시간순서대로 다음의 세부항목(2.1 ~2.3)에 따라 기술한다.
2.1. 기본 흐름	Use Case의 주된 업무 흐름을 시작부터 종료까지의 기본흐름으로 기술한다.
2.1.1 세부 흐름	기본 흐름에서 상세로 내보내야 하는 흐름을 기술한다. (예) 급여를 계산한다. 세부흐름 : 급여 계산의 방법을 기술한다.
2.2. 선택 흐름	기본 흐름 내에서 Actor가 발생시킨 이벤트에 따라선택적으로 수행되는 흐름을 기술한다. 비즈니스로 언급되어 있는 오류를 포함한다. 대부분의 Use Case 는 하나의 기본 흐름이 수행되면서 여러 개의 선택 가능한 흐름으로 분기되어 표현된다. 이 때 분기되는 시점을 기술해 주면 이해에 도움이 된다.
2.2.1. 세부 흐름	선택 흐름 중 상세로 내보내야 하는 흐름을 기술한다.
2.3. 예외 처리	Actor와 Use Case가 상호작용을 하면서, Actor가 잘못된 선택을 하거나(입력 Validation 오류), 시스템이 비정상적으로 반응할 때 발생하는 흐름을 기술한다. 시스템에서 에러로 처리되는 경우를 명시한다.
3. 연관관계	관련 Actor에 의해 Use Case가 실행될 시 일정 부분의 흐름에서 Reference되거나 일정 흐름을 담당하게되는 타 Use Case, Actor가 존재하면 기술을 하도록 한다.
3.1 관련 Use Case	Use Case 다이어그램에서 <<In clude>> 관계 혹은 <<extend>> 관계로 표현되는 Use Case 명과 기능을 간단히 설명한다.

3.2 관련 Actor	해당 Use Case를 시작하고 참조하며 사용하는 Actor들을 기술한다.
4. 사전/사후 조건	Use Case가 시작되기 전에 혹은 Use Case가 수행되고 난 이후에 반드시 충족 시켜야 할 조건이나 제약 사항 등이 있으면 기술하고, 없으면 "없음" 이라고 기술한다.
5. 비고	해당 Use Case에서 참조해야 될 사항이 있으면 기술을 하도록 한다. 즉, 이벤트 흐름에 기술하기 힘든 Use Case의 모든 요구사항, 설계모델에 반영할 비기능적 요구사항을 기술한다. 없으면 '없음'이라고 기술한다.

3.2.3 Use Case 명명 규칙

Use Case 모델링 단계에서 필요한 명명 규칙은 [표 5]와 같다.

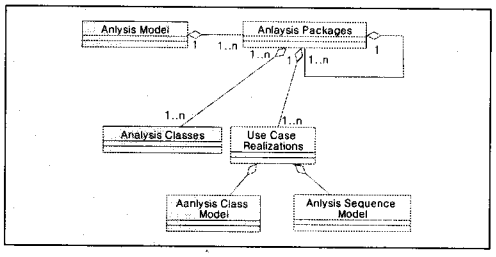
[표 5] Use Case의 명명 규칙

대 상	규 칙	예 제	명명 방법
Actor	명사	기사	용어사전을 바탕으로 고객과 합의된 용어를 사용한다.
Use Case	명사+명사형 동사	출장접수	
Use Case 패키지	명사(서브시스템)	출장서비스	
Use Case 다이어그램	패키지명+Use Case 다이어그램	출장서비스 Use Case 다이어 그램	

3.3 분석 모델링 방법

3.3.1 분석 컴포넌트 모델링

요구사항 수집이 소프트웨어 사용자의 관점에 따라 진행하는데 반해 분석부터는 개발자의 관점에서 시스템을 설명하게 된다. 분석 모델링은 Use Case 명세서에 나타난 이벤트의 흐름을 분석하여 이를 Sequence Diagram으로 표현하고, Sequence Diagram에 나타난 객체를 컴포넌트로 추상화하는 과정이다. [그림 6]은 분석 모델의 구조를 나타내고 있다.



[그림 6] 분석 모델의 구조

(1) 작업 절차 및 산출물

분석 모델링의 산출물은 아래의 [표 6]과 같다.

[표 6] 분석 모델의 산출물과 설명

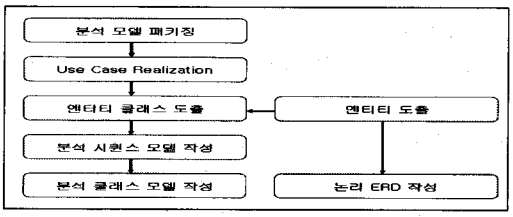
산출물 명	설 명
분석 모델 패키지	분석 모델을 관리하기 용이한 단위로 분할한 모델의 관리 구조 이다.
Use Case Realization	하나의 Use Case가 실행되는 데 어떤 컴포넌트가 필요하며, 그 컴포넌트 간의 상호작용이 어떻게 이루어져 원하는 기능을 수행하는 가를 표현하는 것으로, 하나의 Use Case Realization은 하나 이상의 Component Diagram과 Interaction 다이어그램으로 구성된다.
분석 컴포넌트 상호작용 모델	분석 컴포넌트 상호작용 모델은 Use Case 명세서에 나타난 이벤트의 흐름에 따라 각각의 객체가 상호간에 어떤 메시지를 송수신하는지를 정의하는 것이다.
분석 컴포넌트 구조 모델	분석 컴포넌트 구조 모델은 하나 이상의 분석 컴포넌트 상호작용 모델에서 각각의 시나리오에 참여하는 객체들을 추상화하여 컴포넌트로 도출하여 그들 사이의 관계를 표현한 모델이다.
논리 데이터 모델	논리 데이터 모델은 관리해야 하는 데이터가 중복없이 관리할 수 있도록 조직화하여 표현한 모델이다.

분석 모델링에서는 컴포넌트에 대한 세 가지 스테레오타입을 정의해서 사용하며, 아래의 [표 7]은 각 스테레오타입에 대한 설명이다.

[표 7] 분석컴포넌트 구조 스테레오타입에 대한 설명

컴포넌트 스테레오 타입	설 명
바운더리 컴포넌트	시스템 외부의 Actor와 메시지를 주고받는 컴포넌트, 즉 보통 화면 컴포넌트들이나 다른 시스템 또는 장치와 인터페이스하는 컴포넌트를 의미한다.
컨트롤 컴포넌트	Use Case에 종속적인 Sequence를 관리하는 컴포넌트의 유형으로 하나의 Use Case에 하나의 컨트롤 컴포넌트를 생성하여 사용한다.
엔티티 컴포넌트	업무에서 필요로 하는 정보를 저장하고 있거나, 정보를 제공하는 역할을 수행하는 컴포넌트로 프로그램이 종료한 후에도 영구적으로 저장되어야 하는 정보를 관리한다.

[그림 7]은 분석 모델링 작업 절차를 나타내고 있다.



[그림 7] 분석 모델링 작업 절차

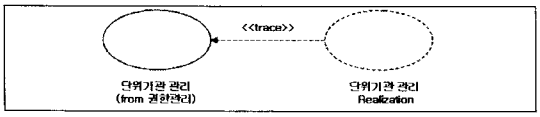
(2) 분석 모델 패키징

하나의 분석 모델은 하나 이상의 관리가 용이한 단위

로 분리하는 것이 가능하다. 이와 같이 하나의 모델을 분할하는 것을 패키징이라고 한다.

(3) Use Case Realization

Use Case Realization이란 분석 모델 안에서 특정 Use Case가 어떻게 현실화되고 실행되며, 분석 모델의 객체들 사이에서 어떻게 상호작용하는 것인가를 정의하는 것이다. [그림 8]은 Use Case Realization 예를 보여 주고 있다.



[그림 8] Use Case Realization 예

(4) 엔티티 컴포넌트 도출

업무에서 필요로 하는 정보를 저장하고 있거나, 정보를 제공하는 역할을 수행하는 컴포넌트로 프로그램이 종료한 후에도 영구적으로 저장되어야 하는 정보를 관리한다. 엔터티 컴포넌트는 처음에 통합 모델러가 일괄적으로 등록한 뒤 사용한다.

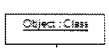
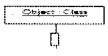
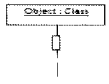
(5) 분석 컴포넌트 상호작용 모델링

분석 컴포넌트 상호작용 모델은 Use Case 명세서에 나타난 이벤트의 흐름에 따라 각각의 객체가 상호간에 어떤 메시지를 송수신하는지를 정의하는 것이다.

① 분석 컴포넌트 상호작용 모델의 구성 요소

[표 8]은 분석 컴포넌트 상호작용 모델의 구성 요소에 대한 설명이다.

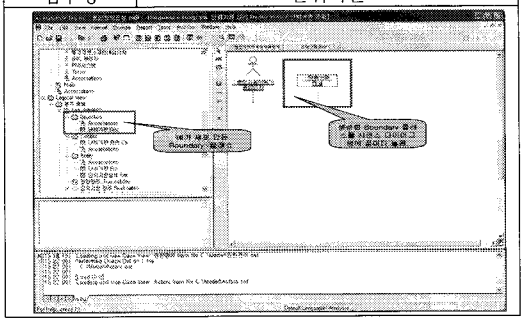
[표 8] 분석 컴포넌트 상호작용 모델의 구성 요소

모델 구성 요소	설 명
	Sequence 모델에서 기본적인 구성요소는 컴포넌트가 아니라 객체이다. 동일한 컴포넌트가 하나 이상 필요한 경우에는 다른 객체명을 부여해서 작성해야 한다.
	객체 아래에 있는 박스는 Activation Bar이다. Acti-vation Bar는 객체가 활성화되는 기간을 나타낸다.
	Sequence 모델은 시간에 따라 상호 호출되는 객체 사이의 관계를 가지기 때문에 순서를 나타낼 수 있도록 Time Line을 가지게 된다. 객체 아래에 수직으로 늘어선 점선은 시간의 경과를 나타내는 Time Line을 의미한다.
	메시지 송수신은 Actor로부터 시작되며, Actor가 발생시킨 최초의 메시지는 시스템 내부에서 하나 이상의 객체 사이의 메시지 송수신을 유발시킨다.

② 분석 컴포넌트 상호작용 다이어그램 작업 절차

[그림 9]는 Sequence Diagram의 바운더리 컴포넌트 생성 예를 보여주고 있다.

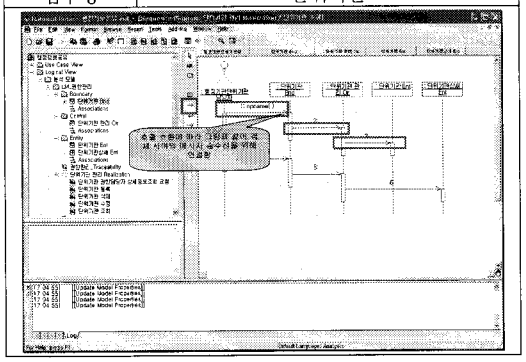
프로세스	바운더리 컴포넌트 생성
업무명	단위기관



[그림 9] Sequence Diagram의 바운더리 컴포넌트 생성

[그림 10]은 Sequence Diagram의 컴포넌트 사이의 흐름 정의를 보여주고 있다.

프로세스	컴포넌트 사이의 흐름 정의
업무명	단위기관

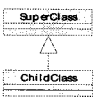
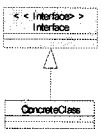
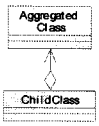


[그림 10] Sequence Diagram의 컴포넌트 사이의 흐름 정의

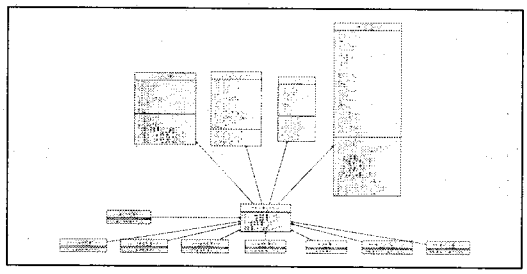
(6) 분석 컴포넌트 구조 모델링

분석 컴포넌트 다이어그램은 Realization된 Use Case별로 하나씩 작성하며, 분석 컴포넌트 상호작용 다이어그램에 나타난 객체를 클래스로 정의하여 클래스 다이어그램의 형태로 작성한다. 분석 컴포넌트 모델은 하나 이상의 분석 컴포넌트 상호작용 모델에서 각각의 시나리오에 참여하는 객체들을 추상화하여 클래스로 도출하여 그들 사이의 관계를 표현한 모델이다. [표 9]는 구성 요소에 대한 설명이다.

[표 9] 분석 컴포넌트 구조 모델의 구성 요소

모델의 구성 요소	설 명
	클래스는 소프트웨어를 구성하는 어떤 것으로 현실세계의 물리적인 사물(도서, 시설 등)이나 관념적인 어떤 것(교육등) 일 수도 있다. 클래스는 명칭, 속성, 오퍼레이션으로 정의된다. 모든 속성과 오퍼레이션은 Visibility(가시성)을 가지게 된다.
	일반적인 클래스 사이의 관계는 Association 관계이다. Association 관계에 있는 클래스는 시퀀스 모델에서 메시지의 송수신이 발생한다. Association 관계는 가장 일반적인 관계이기는 하지만 상위 수준의 관계이며, 보다 물리적인 구현에 가까워질수록 Aggregation 관계로 정제되어 간다.
	Generalization 관계는 클래스 사이의 상속관계에 있을 때 발생한다. 상속관계의 클래스는 하위 클래스에서 상위 클래스의 속성과 오퍼레이션을 이용하는 것이 가능하다. 또한 상속관계가 발생할 때 하위 클래스에서는 상위클래스의 오퍼레이션을 재정의(Override)하는 것이 가능하다.
	Realize 관계는 컴포넌트와 인터페이스 사이의 관계를 정의하며 상속관계와 유사한 방식으로 관계를 정의하지만 점선으로 이루어진다. 인터페이스는 처리로직을 가지지 않기 때문에 연산 단위(Computational Unit)이 아니며 인터페이스를 Realize하는 Concrete 컴포넌트에서 실질적인 처리를 담당하게 된다. 하나 이상의 컴포넌트에서 동일한 오퍼레이션을 제공하는 경우에 공통된 페이스를 제공하기 위해서 Realization 관계를 정의하게 된다.
	Aggregation 관계는 포함관계를 정의한다. Aggregation 관계가 발생하면 두 클래스 사이에 전체와 부분의 관계가 발생하게 된다. 모든 클래스 사이의 Association 관계는 정제되면서 Aggregation 관계로 발전하게 된다. Aggregation 관계는 결합의 정도에 따라 Aggregation과 Composition 관계로 구분할 수 있으며, 전체 클래스가 소멸될 때 부분 클래스가 같이 소멸하는 경우에 Composition 관계로 정의할 수 있다.
	Dependency 관계 Association 관계와 유사하게 대등한 두 클래스 사이의 관계를 정의하지만 Association 관계와 약간의 차이가 있다. 클래스가 다른 컴포넌트를 오퍼레이션의 매개변수나 반환값으로 참조하는 경우에 발생한다.

[그림 11]는 분석 컴포넌트 모델을 클래스 다이어그램의 형태로 작성한 모델의 예이다.



[그림 11] 분석 컴포넌트 모델의 예

3.3.2 분석 컴포넌트 명명 규칙

분석 단계에서 사용할 컴포넌트의 종류별 명명 규칙은 [표 10]과 같다.

[표 10] 분석 컴포넌트의 명명 규칙

대상	규칙	예제
Boundary 컴포넌트	명사 + 화면기능 + Bnd	권한 승인 Bnd 권한 변경 Bnd
오퍼레이션	명사 + 명사형동사 등록: 명사 + 등록 조회: 명사 + 조회 수정: 명사 + 수정 리스트조회: 명사 + 리스트 조회 삭제: 명사 + 삭제	신규권한담당자 등록 권한 수정 권한 조회 권한담당자 리스트 조회 권한 삭제
속성	명사	담당자명

3.3.3 분석 모델 Self 체크리스트

(1) 상호작용 다이어그램 체크리스트

[표 11]은 상호작용 다이어그램 체크리스트의 항목을 보여주고 있다.

[표 11] 상호작용 다이어그램 체크리스트

항 목	Y	N	P	NA
Sequence Diagram을 작성하기 전에 시나리오는 작성하였는가?				
Sequence Diagram에 객체(Object) 또는 컴포넌트 (Class)들 간의 상호작용을 최초 시작(Drive)하는 Actor가 나타나있는가?				
Sequence Diagram에서 객체(Object) 또는 컴포넌트(Class)간의 주고받는 메시지가 명확한가?				
메시지와 오퍼레이션의 구분을 명확히 하고 있는가?				
오퍼레이션의 경우 IN/OUT 파라미터를 명시하였는가?				
Sequence Diagram을 작성하기 전에 각 컴포넌트들에 대해 일반적인 스테레오타입(Common Stereotype)으로 분류 되어 있는가?				
각 Sequence Diagram은 해당 Use Case의 시나리오를 정확히 반영하고 있는가?				
Sequence Diagram에서 Actor와 컴포넌트 간의 상호작용을 보여주기 위한 바운더리 컴포넌트(Boundary Class)가 나타나는가?				
Sequence Diagram에서 주석과 설명을 보여주는 주석(Note Symbol)을 적절히 사용하고 있는가?				
Sequence Diagram에서 컴포넌트들 간의 메시지전달이 시간적인 순서로 정확히 표현되어 있는가?				
모든 시나리오에 대해 Sequence Diagram을 작성하였는가?				

Y: Yes / N: No / P: Partially yes / NA: Not applicable

(2) 컴포넌트 구조 다이어그램 체크리스트

[표 12]는 컴포넌트 구조 다이어그램 체크리스트의 항목을 보여주고 있다.

[표 12] 컴포넌트 구조 다이어그램 체크리스트

항 목	Y	N	P	NA
Sequence Diagram에 표현된 컴포넌트나 객체와 omponent Diagram에 표현된 컴포넌트가 Mapping이 되는가?				
모든 데이터는 컴포넌트 내에 감춰져 있는가?				
컴포넌트가 컴포넌트 사용자(Users)들에게 종속되어 있지는 않은가?				
사용자 인터페이스(User Interface)에 지나치게 종속된 컴포넌트는 없는가?				
시스템외부에 있는 컴포넌트가 있지는 않은가?				
Collaboration Diagram에 모델과 관계없는 컴포넌트는 없는가?				
컴포넌트명, 속성(Attribute)명, 오퍼레이션(function)명이 명명규칙에 의거 작성되었는가?				
유사한 기능을 가진 오퍼레이션이 많은 컴포넌트가 있는가?				
타 컴포넌트의 데이터와 오퍼레이션을 지나치게 많이 사용하는 오퍼레이션을 가진 컴포넌트는 없는가?				
무관한 정보를 많이 가진 컴포넌트는 없는가?				

Y: Yes / N: No / P: Partially yes / NA: Not applicable

3.4 설계 모델링 방법

1990년대 중반부터 소프트웨어 산업에 적용되기 시작한 CBD 접근법은 독립적인 소프트웨어의 재사용을 통한 개발 비용과 시간의 단축을 강조하고 있다. 소프트웨어에서와 비즈니스에서 CBD가 크게 가시적인 효과를 이루지 못하는 이유 중의 가장 큰 것은 바로 컴포넌트라면 무조건 재사용으로 접근한다는 점이다. 소프트웨어에서의 CBD의 접근은 소프트웨어에서 적용할 컴포넌트의 종류를 정의하고 그 컴포넌트에 따라 목적을 다르게 가져야 하는 것이 중요하다.

3.4.1 설계 모델링

Use Case 모델링과 분석 모델링 과정을 통하여 도출해낸 비즈니스 성격의 분석 모델을 이에 해당하는 Operation을 시스템 구현과 밀접한 형태의 모델로 변환하여 정제하는 단계이다.

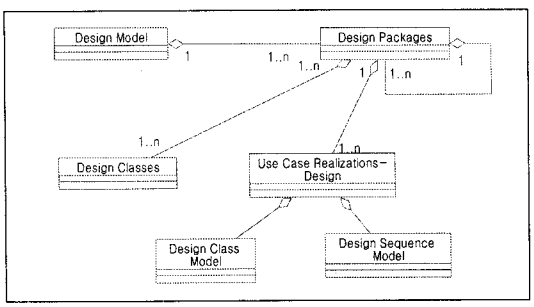
(1) 작업 절차 및 산출물

[표 13]은 설계 모델의 산출물에 대한 설명을 하고 있다.

[표 13] 설계 모델의 산출물과 설명

산출물 명	설 명
설계 모델 패키지	분석 모델을 관리하기 용이한 단위로 분할한 모델의 관리 구조이다.
Use Case Realization	하나의 Use Case가 실행되는 데 어떤 컴포넌트가 필요하며, 그 컴포넌트 간의 상호작용이 어떻게 이루어져 원하는 기능을 수행하는 가를 표현하는 것으로, 하나의 Use Case Realization은 하나 이상의 Component Diagram과 Interaction Diagram으로 구성된다.
설계 포넌트 상호작용모델	설계 컴포넌트 상호작용 모델은 Use Case 명세서에 나타난 이벤트의 흐름에 따라 각각의 객체가 상호간에 어떤 메시지를 송수신하는지를 정의하는 것이다.
설계 포넌트 구조 모델	설계 컴포넌트 구조 모델은 하나 이상의 설계 컴포넌트 상호작용 모델에서 각각의 시나리오에 참여하는 객체들을 추상화하여 컴포넌트로 도출하여 그들 사이의 관계를 표현한 모델이다.
물리 데이터 모델	물리 데이터 모델은 관리해야 하는 데이터가 중복없이 관리될 수 있도록 조직화하여 표현한 모델이다.

[그림 12]는 설계 모델 상호간의 관계를 보여주고 있다.



[그림 12] 설계 모델 상호간의 관계

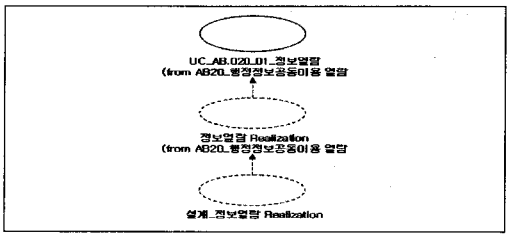
[표 14]는 설계 모델의 스테레오타입에 대한 설명을 하고 있다.

[표 14] 설계 모델의 스테레오타입과 설명

스테레오타입	설 명	분석스테레오타입
JSP	Presentation Tier로 사용자와의 인터페이스 역할을 하는 HTML을 포함하는 사용자 화면이다.	Boundary
Servlet	서브 시스템당 하나의 Servlet이 구성되며, JSP에서 사용자가 입력한 데이터를 Control tier로 넘기거나, 혹은 Control tier에서 나온 결과를 JSP에 넘겨주는 역할을 한다. 본 논문에서는 하나의 servlet을 생성하고, 이는 Command 역할을 하는 객체에게 사용자가 입력한 데이터를 넘겨주고 Business Controller에서 넘겨 받은 자료를 사용자가 사용할 수 있게 넘겨주는 역할을 한다.	Boundary
command	Business Tier와 Presentation Tier 사이의 연결자로 입력받은 Data를 Value Object로 가공하여 이를 Control tier에게 가공된 결과를 넘겨주는 역할을 한다. Command 객체는 하나의 액션 혹은 이벤트 당 하나씩 만들어져야 한다.	
Business controller	관련된 비즈니스를 하나로 묶은 비즈니스 컴포넌트의 역할을 수행한다. Command로부터 업무처리 요청과 관련 데이터를 받아서 처리해야 될 업무의 흐름에 따라 Data Access Object를 이용하여 업무처리를 수행한다. Business controller 는 Use Case당 하나씩을 만들어서 사용한다.	Control
Data Access Object	Database 혹은 Legacy System과 연동하며, 데이터를 직접 조작하는 오퍼레이션을 가지고 있다. Use Case당 하나의 Data Access Object를 생성하여 사용한다.	Entity

(2) 설계 모델의 Use Case Realization

Use Case와 Use Case Realization을 분리하는 목적 중의 하나는 이미 Baseline 화된 Use Case에 영향없이 설계를 수정할 수 있는 유연성과 분석 단계의 Use Case가 설계 단계에서 어떠한 형태로 변화되는지를 추적할 수 있는 Trace를 찾기 위함이다. 만약에 분석 단계에서 Use Case가 도출되지 않았다면 분석 단계의 Use Case를 다시 잡아 분석 단계 이후의 Process를 진행하도록 하고, 그 외에 도출은 되었으나 Realization 시 변경이 일어나면 이를 Use Case Trace Diagram을 이용하여 추적성을 관리할 수 있도록 한다. 설계 단계의 Use Case Realization은 통합 모델러가 일괄적으로 작성하도록 한다. [그림 13]은 설계 모델의 Use Case Realization 과정을 보여주고 있다.

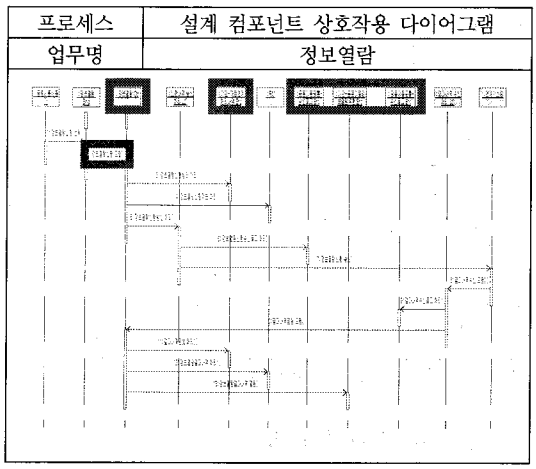


[그림 13] 설계 모델의 Use Case Realization

(3) 설계 컴포넌트 상호작용 모델링

① 설계 컴포넌트 상호작용 모델링 원칙

분석 단계에서 Boundary, Control, Entity 세 가지 스테레오타입으로 분류하여 정의한 컴포넌트는 설계 모델에서 더 작은 단위로 분할되고 정제된다. 아래의 [그림 14]는 분석 모델에서 작성한 분석 Sequence Diagram을 설계 컴포넌트 상호작용 다이어그램으로 변환하는 방법을 보여준다. Boundary 컴포넌트에서 호출하던 Control 컴포넌트의 오퍼레이션은 설계 모델에서 하나의 Command 컴포넌트로 분화된다. 또한 분석 모델에서 엔터티 컴포넌트는 설계 모델에서 하나의 DAO로 합쳐지며 각 Entity 컴포넌트에 대한 입력、수정、삭제、조회에 대한 요청은 DAO 컴포넌트의 오퍼레이션으로 변환된다.

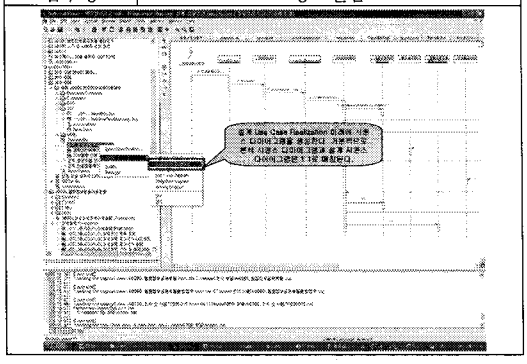


[그림 14] 설계 컴포넌트 상호작용 다이어그램

② 설계 컴포넌트 상호작용 모델링 절차

[그림 15]는 설계 컴포넌트 상호작용 다이어그램 생성과 설계 컴포넌트 상호작용 다이어그램의 Actor 배치를 보여주고 있다.

프로세스	설계 컴포넌트 상호작용 다이어그램
업무명	정보열람

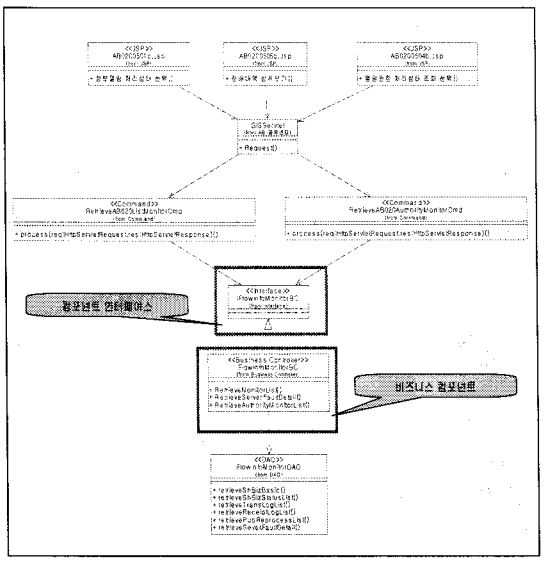


[그림 15] 설계 컴포넌트 상호작용 다이어그램 생성

(4) 설계 컴포넌트 구조 모델링

설계 컴포넌트 구조 다이어그램은 Use Case Realization 당 하나씩 생성하며, [그림 16]과 같이 해당 Use Case Realization이 가지고 있는 컴포넌트 상호작용을 표현하는 시퀀스 다이어그램에 참여하는 모든 컴포넌트를 식별하여 클래스 다이어그램의 형태로 그린다.

Business Controller로 구현되는 컴포넌트는 컴포넌트의 인터페이스를 Implementation하여 구현하게 되며, Business Component에 해당하는 Business Controller는 데이터베이스에 대한 입출력을 담당하는 DAO 계층과 외부 시스템의 연계를 담당하는 API 호출부분을 담당하는 부분을 추상화하는 역할을 수행한다.



[그림 16] 설계 컴포넌트 구조 모델링

3.4.2 설계 모델 명명 규칙

설계 단계에서 사용할 컴포넌트의 종류별 명명 규칙은 [표 15]와 같다.

[표 15] 설계 모델의 명명 규칙

대 상	규 칙	예 제
JSP	XXXXX ①②③④ X XXXXX : 패키지구분 코드 ①② : 업무 기능 번호 ③④ : 화면 번호 x : Frame 유형별 화면 유형 식별 코드 f : frame m : menu b : body p : popup (sequence 1자리) r : report h : help i : iframe	AB0200103b.jsp
Command (Action별)	Retrieve+XXXXX + 의미있는 명 + ListCmd : 리스트 조회 용 Cmd Create + XXXXX + 의미있는 명 + Cmd : 데이터생성 Command Update + XXXXX + 의미있는 명 + Cmd : 데이터수정 Command Delete + XXXXX + 의미있는 명 + Cmd : 데이터삭제 Command XXXXX : 패키지 구분 코드	RetreiveAB020ListInspMstCmd
DAO (Use Case 당 하나)	Use Case의 영문명 + DAO	정보열람 DAO ReadInfoDAO
Business Controller (Use Case 당 하나)	Use Case의 영문명 + BC	정보열람 ReadInfoBC
Operation 명	속성 Return : get + 의미 있는 명 속성 설정 : add + 의미 있는 명 Boolean Return : is + 의미있는 명 단건 데이터 생성 : create+ 의미있는 명 단건 데이터 조회 : retrieve+의미있는 명 단건 데이터 삭제 : delete+의미있는 명 단건 데이터 수정 : update+의미있는 명 다건 데이터 생성 : create+의미있는 명 + List 다건 데이터 조회 retrieve+의미있는 명 + List 다건 데이터 삭제 : delete+의미있는 명 + List 다건 데이터 수정 : update+의미있는 명 + List	getMaxCount() addJuminNo(String pst_jumin) isValidateDate(Date pdt_date) createSlip () retrieveSlip() deleteSlip () updateSlip () insertSlipList() retriveSlipList() deleteSlipList() updateSlipList()
Attribute 명	초기 문자 : 소문자 단어와 단어 사이는 대문자 용어집에 존재하는 명 사용	사 번 : empNo 접수 번호 : recNo

3.5 제안 모델의 평가

제안된 모델에 대한 평가 항목은 소프트웨어 감리지침과 컴포넌트 기반 개발 도구의 기능적 요구사항들로 구성하였다. 즉, 기능적 요구 사항별로 선정한 도구 및 제안한 개발 모델의 기능의 지원 정도를 [표 16]과 같이 평가하였다.

[표 16] 제안 모델의 평가

기 능		제안 모델
분석 및 설계 지원	컴포넌트 설계지원(순공학)	○
	역공학	×
	비즈니스 프로세스 공학	○
	분석 및 설계 모델 사이의 입출력 지원	◎
	설계 편의 기능 지원	◎
	산출물 사이의 일관성 검증	◎
	프로토타이핑 및 시뮬레이션	△
	데이터베이스 모델링	○
	사용자 인터페이스 설계	△

◎ : 상, ○ : 중, △ : 하, × : 지원 안됨

4. 결론

객체지향 개발방법론을 실제 적용하는 프로젝트를 업무에 필요한 단계별 세부 지침을 적극적으로 반영하여 객체지향 정보 시스템 개발 프로젝트의 생산성 및 품질 향상에 기여하여야 한다. 따라서 본 논문에서는 객체지향 환경의 소프트웨어 개발 생명주기에서 소프트웨어 생산성 향상을 위한 사용자 요구분석, 설계 방법을 고려한 효율적인 객체지향 Use Case 모델링 방법을 제안하였다. 본 논문에서는 제안한 방법은 객체지향 환경의 대규모 프로젝트 구축에서 신속하고 반복적인 개발 방법론은 사용하여 소프트웨어 신뢰성, 재사용성, 유지보수성이 우수한 장점을 제공하고 있다. 향후 연구과제는 실제 프로젝트에 편리하게 사용할 수 있으며 보다 정형화되고 표준화된 생산성이 우수한 객체지향 개발 방법론에 대한 연구가 절실히 요구된다.

참고문헌

[1] Kim, Soo Dong; Lessons Learned from a Nation-wid e CBD Promotion Project, Commu-nications of the A CM, Vol.45, Issue.10, pp.83-87, Oct. 2002.

[2] 김영희; 프로세스와 산출물을 통합 관리하고 웹서비스를 수용하는 CBD 도구 모델, 박사학위논문, 숭실대학교, 2005.

[3] 정연대, 임진수, 오연재; S/W 컴포넌트 조립도구기반 조립방법론 연구, 한국정보처리학회, 제10권, 제3호, pp.74-88, 2003.

[4] Erich Gamma, et al; Design Patterns:Abstraction and Reuse of Object-Oriented ECOOP'93, pp.406-421, July 1993.

[5] J. Cheesman and J. Daniels; UML Components : A Simple Process for Specifying Component-Based Software, Addison-Wesley, 2000.

[6] J. Rumbaugh, Ivar Jacobson, Grady Booch; "The Unified Modeling Language Reference Manual, Second Edition.", Boston, MA:, Addison-Wesley, 2005.

[7] Rational Software Corporation; Rational Unified Process, 2002.

[8] Mikio Aoyma; New Age of Software Development : New Component-Based Software Engineering Changes the Way of Software Development, International Wor kshop on Component-Based Software, ICSE, 2003.

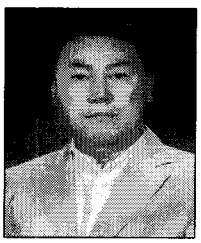
[9] 삼성SDS IT Review, CBD 동향 보고서; 2003년 상반기 기술 및 시장 동향 보고, 2003.

[10] 김영규; 소프트웨어 생산성 향상을 위한 프레임워크 설계:EOOM, 박사학위논문, 강원대학교, 2008.2.

[11] LG CNS;분석 모델링 가이드, GIC-221-05, Ver.1.0

[12] LG CNS;유스케이스 모델링 가이드,GIC-221-01

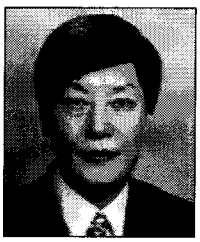
김 영 규(Young-Gyu Kim) **[정회원]**



- 1989년 8월 : 충북대학교 컴퓨터과학과 석사
- 2008년 2월 : 강원대학교 컴퓨터과학과 이학박사
- 2008년 11월 현재 : 한림성심대학 인터넷비즈니스과 부교수

<관심분야>
소프트웨어공학, ERP, 정보통신

양 해 술(Hae-Sool Yang) **[정회원]**



- 1991년 日本 오사카대학 정보공학과 S/W 공학전공(공학박사)
- 1980년~1995년 강원대학교 전자계산학과 교수
- 1999년~현재 호서대학교 벤처전문대학원 교수
- 2001년~현재 한국정보처리학회 부회장
- 2005년~현재서울벤처정보대학원대학교 교무처장

<관심분야>
소프트웨어공학, ERP, 정보통신

최 형 진(Hyung-Jin Choi) **[정회원]**

- 1990년 일본 동경공업대학 정보공학졸업(공학박사)
- 1990년~1991년 한국전자통신연구원 선임 연구원
- 1991년~현재 강원대학교 컴퓨터과학과 교수

<관심분야>
영상처리, 인공지능, 컴퓨터그래픽스