UML을 기반으로 한 실무 중심의 객체지향 방법론

조 은 숙† · 김 수 동†† · 류 성 열††

요 약

소프트웨어 개발의 규모가 대형화되고 인터넷 기반의 네트워킹, 멀티미디어화, 분산화 등과 같이 복잡, 다양해짐에 따라 이에 효율적으로 대처할 수 있는 소프트웨어 개발 방법론이 요구되고 있다. 특히, 소프트웨어 규격 및 시장의 국제화로 인해 ISO 9000-3등의 국제 품질 기준을 수용할 수 있어야 하며, 선진국의 객체지향 개발 기술의 급속한 보급과 표준화로 인해 객체지향 기술을 적용한 객체지향 개발 방법론이 필요하다. 최근 OMG에서 UML을 표준 객체지향 모델링 언어로 채택함에 따라 Java나 CORBA와 같은 객체지향 기술을 적용하여 소프트웨어를 개발할 경우, 대부분 UML을 사용할 것으로 예상된다. 그러나, 현재 UML은 실무 프로젝트에 적용하기에는 적절한 개발 프로세스가 정의되어 있지 않을 뿐만 아니라 UML에서 제공하고 있는 많은 기법들을 개발 과정에 효율적으로 적용할 수 있는 구체적인 지침이 제시되어 있지 않다. 국내 대부분의 방법론들은 기존의 구조적 또는 OMT 기반의 객체지향 방법론만을 제시하고 있다.

본 논문에서는 UML을 실무에 보다 효율적으로 적용할 수 있도록 하기 위해, UML을 기반으로 한 개발 프로세스 정의와 개발 단계별 업무들에 대한 구체적인 지침을 제시한다. 또한 ISO 9000-3의 권고에 기준한 단계별 업무간의 작업흐름을 정의한다. 특히 UML에서 제공하고 있지 않는 사용자 인터페이스 분석 및 설계 기법과 웹 어플리케이션 구축 시 필요한 실무적인 시스템 개발 기법을 제시한다.

UML-Based Industry-Strength Object-Oriented Methodology

Eun-Sook Cho† · Soo-Dong Kim†† · Sung-Yul Rhew††

ABSTRACT

As the complexity of software development is increasing due to networking, multimedia, and diverse system architecture, the need for effective software development methodology is increasing. Especially, due to software standard and internalization of software market, it is necessary to accept international quality such as ISO 9000-3. In addition, object oriented development methodology is required due to rapid propagation of OO technology and standardization. Recently, UML was accepted by the OMG as standard object-oriented modeling language for distributed environment. When we develop Java and CORBA-based software, often UML is applied to Java and CORBA-based projects. However, current UML does not provide industry-strength process, and concrete guidelines. Most of domestic methodologies only provide structural or OMT-based object-oriented methodologies.

In this paper, we propose UML-based development process and concrete guidelines for each phase in order to apply UML to software development practically and effectively. Also, we define the transition guidelines and semantics between various development tasks. In addition, the analysis and design techniques of user interface and system development techniques needed in Web application development are presented.

† 준 회 원 : 숭실대학교 대학원 전자계산학과
†† 종신회원 : 숭실대학교 컴퓨터학부 교수
논문접수 : 1998년 9월 30일, 심사완료 : 1999년 1월 5일

1. 서 론

소프트웨어 개발 과제가 대형화 되고, 네트워킹, 멀티미디어화, 분산화 등이 요구됨에 따라서 소프트웨어 개발의 복잡도가 더욱 커지기 시작했다. 현재 대부분의 회사들이 객체지향 기술인 Java[1,2,3]와 CORBA[1]를 기반으로 한 웹 어플리케이션 개발에 주력을 하고 있다. 이러한 변화를 효율적으로 지원할 수 있는 소프트웨어 개발 방법론이 절실히 요구되고 있는 실정이다. OMG(Object Management Group)에서 표준 객체지향 모델링 언어로 UML(Unified Modeling Language)을 채택하였다. 이 UML[4,5,10]이 산업계 표준으로 채택되었기 때문에 여러 회사들은 이 방법론을 적용하여 소프트웨어를 개발하게 될 것이다. UML이 표준 객체지향 모델링 언어로 채택되었지만 이것을 실무에 바로 적용하기가 어렵다. 최근에 Objectory Process에 이어서 Unfied Process가 소개되었지만 개발 프로세스의 업무 정의나 업무들에 대한 지침이 개략적으로 제시되어 있다[14,15]. 특히, UML에서 제공하고 있는 여러 다양한 다이어그램들을 소프트웨어 개발 과정에 어떻게 적용해야 하는 지에 대한 구체적인 지침이 마련되어 있지 않기 때문에, 개발자들이 소프트웨어를 개발하는 데 많은 어려움을 겪게 될 것이다. 본 논문에서는 UML을 실무에 효율적으로 적용 가능하도록 하기 위해, UML에 추가적으로 개발 프로세스를 정의하며, 전체 개발 과정에 대한 단계 정의 및 단계별 업무들을 정의한다. 단계 및 업무들 간의 효율적인 상호 관계를 식별하기 위한 단계별 업무들 간의 업무흐름을 정의한다.

한편 국·내외 대부분의 회사들이 소프트웨어를 인터넷 기반의 웹 어플리케이션으로 구축하고 있으며, 이에 Java와 CORBA와 같은 객체지향 기술들을 적용하고 있다. 인터넷을 기반으로 한 웹 어플리케이션을 구축하는 경우는 소프트웨어의 성능, 신뢰성, 보안성, 재사용성, 고품질 등의 다양한 요소들이 요구되며, 개발 과정에 이러한 요소들을 지원할 수 있는 기법들이 제시되어야 한다. 웹 어플리케이션에서는 사용자 인터페이스에 대한 설계와 성능 및 시스템의 효율성을 향상시키기 위한 시스템 설계 기법이 특히 요구된다. 본 논문에서는 기존의 UML에서 제시하지 않는 사용자 인터페이스 분석 및 설계기법을 구체적으로 제시하며, 웹 어플리케이션 구축 시 중요하게 요구되는 시스템 설계를 논리적인 시스템 설계와 물리적인 시스템 설계로 구분하여 각각에 대한 설계 기법을 구체적으로 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로서 기존의 대표적인 객체지향 개발 방법론들의 특징 및 장·단점을 설명하고, 본 방법론의 특성을 제시한다. 3장에서는 UML을 기반으로 한 개발 프로세스 정의와 소프트웨어 개발 과정의 각 단계별 업무들에 대한 정의 및 업무별로 적용되는 UML 기법에 대해 소개한다. 4장에서는 논문에서 제시한 개발 방법론을 적용한 사례 연구 및 타 방법론과의 평가를 제시한다. 마지막으로, 5장에서 본 논문에 대한 결론 및 향후 연구과제를 제시한다.

2. 관련 연구

현재까지 Rumbaugh의 OMT[6], Booch의 OOAD[7], Jacobson의 Use Case-Driven[8] 방식 등의 많은 객체지향 방법론들이 소개되어 왔다. 본 장에서는 이러한 대표적인 객체지향 방법론들의 특징 및 장·단점에 대해 살펴보고, 본 논문에서 제안하고 있는 방법론의 특성에 대해 제시한다.

2.1 Rumbaugh의 OMT(Object Modeling Technique)

분석 단계에서 기본적으로 시스템에 대해 3가지 측면에서 관찰하여 각각의 측면에 대한 모델인 객체 모델(Object Model), 동적 모델(Dynamic Model), 그리고 기능 모델(Functional Model)들을 제시하고 있으며, 설계 단계에서 이 3가지 모델들을 하나로 통합하고 있다. 장점은 각각의 모델에서 사용하고 있는 다이어그램들 간의 흐름이 단계적으로 잘 정의되어 있으며, 각 다이어그램에 대한 접근 경로 검증 단계가 존재해서 다이어그램에 대한 검증이 비교적 잘 이루어 진다. 단점으로는 요구사항이 잘 정의되어 있다는 가정하에서 분석에 들어가기 때문에, 요구 사항 명세서가 매우 복잡하거나 난해한 경우에는 분석 단계에서 클래스들을 바로 추출하기가 어렵다. 계산적인 부분에 미약하기 때문에, 계산을 많이 요구하는 도메인에는 적용하기가 어렵다. 이 방법론은 개발 프로세스가 분석과 설계에 대한 업무만 제시하고 계획이나 구현 및 시험과 같은 단계에 대해서는 구체적으로 제시하고 있지 않다.

2.2 Booch의 OOAD

설계의 문서화를 강조하며 다이어그램 중심으로 시

스템을 개발하는 방법이다. 이 방법론에서는 다른 방법론들과는 달리 분석과 설계를 분리하지 않았으며, 시스템을 논리적 관점과 물리적 관점을 포함한 정적 모델과 동적 모델로 크게 나누어 설계하고 있다. 이 방법론의 장점은 표기법들이 다양해서 시스템을 가시화하는 데 유용하며, 표기법이 잘 정의되어 있다. 단점으로는 설계에 치우쳐 있고 프로세스보다는 표기법에 집중되어 있어서 단계적으로 업무들에 대한 정의가 제시되어 있지 못할 뿐만 아니라 업무들 간의 순서도 제시되어 있지 않다. 이로 인해 개발자가 방법론에서 제시하고 있는 다이어그램들을 업무에 적용하기가 어렵다. 계산을 많이 요구하는 부분을 모델링할 수 있는 기법이 미약하고, 너무 많은 표기법의 제공으로 인해 정형화(Formality)가 떨어진다.

2.3 Jacobson의 OOSE

이 방법론의 특징은 문제 도메인에 대한 분석을 수행하기 이전에 도메인을 Use Case 단위로 나누어서 분석한다는 것과 객체 유형(Object Type)을 3가지 형태로 구분한다는 것이다. 장점으로는 요구사항 수집 능력이 뛰어나다는 것과 객체 유형을 3가지 형태로 구분함으로써 시스템의 견고성과 효율성을 높일 수 있다. 단점으로는 분석 단계에서 구현에 관한 사항들을 고려한다는 것과 요구 사항 수집을 위한 부분에 치우쳐 있음으로 인해서 분석에 치중된 방법론이다. 이 방법론 또한 다른 방법론처럼 전체 개발 프로세스를 다루고 있지 않고 부분적인 단계와 업무만 다루고 있다. 업무의 입력물이나 산출물들에 대한 정의도 없을 뿐만 아니라 업무의 순서나 업무에 대한 구체적인 지침이 제시되어 있지 않다.

2.4 UML

UML은 위의 3가지 방법론들을 통합하여 확장한 모델링 언어로서, 기존의 방법론들의 장점들을 흡수한 것이다. UML에서는 도메인을 기능적으로 분할할 수 있는 장치인 Use Case 다이어그램과 Use Case 정의서를 제공하고 있으며, 시스템의 정적, 동적, 기능적인 부분을 모델링할 수 있는 기법으로 클래스 다이어그램, 순차도 또는 협력도, 상태 전이도, 그리고 활동도를 제공하고 있다. 관련있는 클래스들을 결합할 수 있는 장치로 패키지 다이어그램을 제공하고 시스템에 대한 물리적 장치를 표현할 수 있는 기법으로 Deployment 다이어그램을 제시하고 있다. UML은 이처럼 많은 다이어그램들을 제공하고 있지만, 회사별로 프로젝트 규모나 성격에 따라 적용할 수 있는 프로세스나 지침이 추가적으로 필요하다.

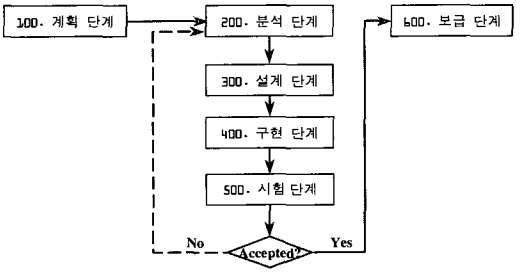
3. 개발 프로세스 및 단계별 업무 정의

본 장에서는 본 개발 방법론에서 제시하고 있는 소프트웨어 개발 프로세스에 대한 정의와 각 단계별 업무들에 대한 정의 및 각각의 업무들에 대한 지침을 제시한다.

3.1 개발 프로세스 정의

본 방법론에서는 점진적이면서 반복적인 소프트웨어의 개발을 지원하는 개발 프로세스를 정의한다[12]. (그림 1)에서 보이는 것처럼 전체 소프트웨어 개발 공정은 100 단계인 계획 단계부터 시작하여 600 단계인 보급 단계로 종결 지어진다. 중간의 200.분석, 300.설계, 400.구현, 500.시험 단계를 반복적으로 적용할 수 있기 때문에 중·대형 과제에 적용 가능하다.

본 논문에서 제안한 개발 프로세스는 프로젝트 성격 및 규모에 따라 다양한 프로세스로 적용 가능하다. (그림 1)에서 점선 부분이 없으면 선형 프로세스로 적용 가능하다. 즉, 100.계획 단계부터 600.보급 단계까지 반복없이 순차적으로 진행된다. 이는 요구사항이 명확히 정의되어 있는 중·소형 개발 과제에 적용 가능하다.



(그림 1) 개발 프로세스
(Fig. 1) Development Process

도메인이 독립적인 모듈들로 분할될 수 있는 경우와 짧은 기간(2~4개월)내에 시스템 개발이 요구되는 경우는 200에서 500에 이르는 단계들을 여러 팀들로 구성하여 개발을 진행할 수 있다. 즉 RAD(Rapid Application

Development) 프로세스로 적용 가능하다[9].

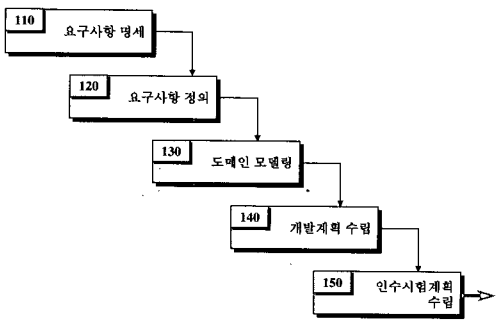
요구사항 정의가 미흡하거나 중·대형의 복잡한 과제인 경우는 시스템을 기능별로 나누어서 주요 기능부터 점차적으로 개발해야 한다. 이러한 경우는 (그림 1)의 프로세스에서 모듈별로 200에서 500 단계를 반복적으로 적용하면서 시스템을 개발한다.

3.2 단계 및 단계별 업무 정의

전체 소프트웨어 개발 프로세스는 6개의 단계인 100.계획 단계, 200.분석 단계, 300.설계 단계, 400.구현 단계, 500.시험 단계, 600.보급 단계로 구성되어 있다. 그리고, 단계별 업무들은 필수 업무와 선택 업무로 구성되어 있다. 선택업무는 프로젝트의 성격에 따라 생략 가능한 업무를 의미한다.

3.2.1 계획 단계(100)

계획 단계는 목표 시스템의 개발 이전에 개발 목표 및 범위의 선정, 개발 프로세스 정의 및 과제 관리 등 종합적인 개발 계획을 수립하는 단계로서, 목표 시스템의 아키텍처에 대한 기준선(Baseline)을 선정하고 도메인에 대한 이해를 위해 Use Case 모델링을 한다. 계획 단계의 업무들과 업무들간의 흐름이 (그림 2)에 제시되어 있다.

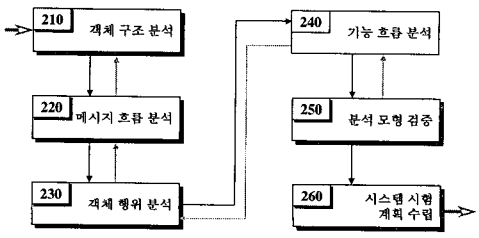


(그림 2) 계획 단계
(Fig. 2) Planning Phase

계획 단계의 130. 도메인 모델링 업무에서는 도메인에 대한 이해를 돕기 위해 도메인에 대한 개략적인 모델링을 하는 업무로서, UML의 개략 클래스 다이어그램(Conceptual Class Diagram)을 작성하고 목표 도메인에 대한 작업 흐름이나 기능흐름을 관찰하여 모델링하는 개략 활동도(Conceptual Activity Diagram)을 작성한다. 개략 클래스 다이어그램에서는 주요 클래스들과 클래스들 간의 관계들 만을 정의한다.

3.2.2 분석 단계(200)

분석 단계에서는 요구사항 명세서의 분석 및 현업 조사 등을 통하여 목표 도메인을 이해하고, 나아가 목표 시스템의 논리적 모델을 설정한다. 분석 단계에서는 업무들 간의 재귀적 흐름이 정의되어 있으며, 기능 흐름 분석 업무는 도메인이 많은 계산을 요구하는 경우가 아니면 생략 가능한 선택 업무이다. 분석 단계에서는 목표 도메인을 정적인 측면, 동적인 측면, 기능적인 측면에서 모델링하기 위한 업무들로 구성된다. 210. 객체 구조 분석 업무에서 정적인 측면을 모델링하는데 UML의 명세 클래스 다이어그램(Specification Class Diagram)을 사용한다. 이 명세 클래스 다이어그램을 작성하기 위해서는 이전 단계의 요구사항 정의서와 Use Case 명세, 그리고 개략 클래스 다이어그램을 기반으로 클래스들을 추가하며, 클래스에 속성이나 다중성 등을 추가한다. 220. 메시지 흐름 분석과 230. 객체 행위 분석 업무에서는 도메인의 동적인 측면을 모델링하는 데 UML의 순차도(Sequence Diagram) 또는 협력도(Collaboration Diagram)와 상태 전이도(State Diagram)를 사용한다. 순차도와 협력도는 추출된 객체들 간의 상호 작용을 분석하기 위한 모델링 기법으로, Use Case의 시나리오 단위로 순차도나 협력도를 작성한다. 상태 전이도는 시간에 따른 객체의 변화와 행동을 모형화 하는 기법으로서, 시스템 내에서 객체의 상태 변화와 상태 변화를 가져오는 흐름을 표현한다. 상태 전이도는 이전 메시지 흐름 분석 업무에서 생성된 순차도 또는 협력도의 객체 단위로 작성하게 된다.



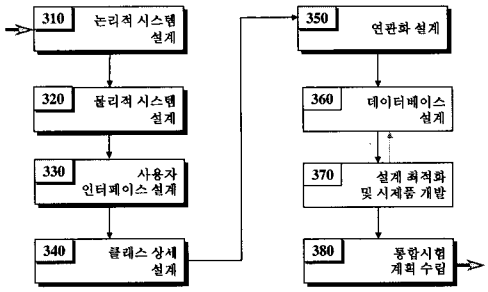
(그림 3) 분석 단계
(Fig. 3) Analysis Phase

240. 기능 흐름 분석 업무는 목표 시스템의 작업 흐름(Workflow)를 모형화하기 위해 UML의 활동도를 사용하여 시스템에서의 기능들이 수행되는 과정을 표현한다. 이 업무는 목표 시스템이 많은 계산을 요구하지 않거나 작업 흐름이 단순한 경우는 생략가능하기 때문에 선택 업무로 정의하였다. 활동도를 작성할 때에도 순차도처럼 Use Case 단위로 작성한 후 활동도들을 통합 또는 분할하게 된다.

3.2.3 설계 단계(300)

설계 단계는 객체지향 분석 모델을 기반으로 구현 도구 및 운영 환경을 고려한 시스템 구조 설계 및 세부 설계를 수행하는 단계이다. 설계 결과 모델을 구현에 직접 활용될 수 있는 정도의 구체성과 실무성을 지녀야 한다. 특히, 본 방법론에서는 설계 단계에서 웹 어플리케이션 구축시 시스템 성능 향상을 위해 시스템 설계를 310. 논리적 시스템 설계와 320. 물리적 시스템 설계로 구분하여 정의한다. 논리적 시스템 설계는 재사용 가능한 단위로 관련있는 객체들을 그룹화하는 업무로서, 객체들을 그룹화하기 위한 장치로 UML의 패키지 다이어그램을 사용한다. 패키지는 명세 클래스 다이어그램에 있는 클래스들 간의 의존성(응집도/결합도)을 측정하여 의존성이 높은 클래스들을 패키지 단위로 그룹화한다[11,16,17]. 특히, 웹 어플리케이션을 구축하는 경우에 있어서는 시스템의 성능 및 효율성을 향상시키기 위해 논리적 시스템 설계를 통해 객체들을 클러스터링해야 한다. 물리적 시스템 설계 업무는 목표 소프트웨어를 구성하는 컴포넌트들을 찾아서, 이 컴포넌트들 간의 인터페이스를 정의하여 구현과정에서 개발 작업의 단위로 사용하도록 하는 데 목적이 있다. 그리고, 클라이언트/서버 또는 분산 구조를 위한 시스템 아키텍처를 작성하여 물리적으로 객체들을 클러스터링하는 업무이다. 이 업무에서는 목표 시스템의 아키텍처를 크게 사용자 인터페이스 계층(User Interface Layer), 정보처리 계층(Processing Layer), 그리고 데이터 계층(Data Layer)으로 분류하여 각각의 계층에 속할 객체들을 클러스터링한다[13,17,20]. 이를 위해 UML의 Deployment 다이어그램을 사용한다.

특히 웹 어플리케이션을 구축하는 경우에서 고려해야 할 사항 중의 하나로 사용자 인터페이스 설계가 해당된다. 330. 사용자 인터페이스 설계 업무는 효율적이고 신뢰성 있는 사용자 인터페이스를 설계하기 위해 사용



(그림 4) 설계 단계
(Fig. 4) Design Phase

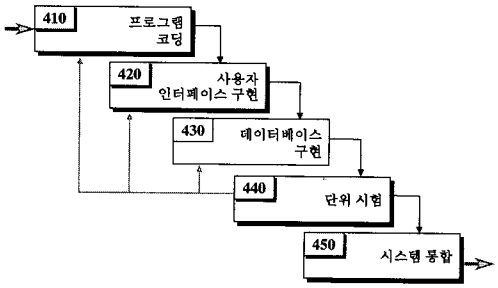
자 인터페이스 시나리오 선정, 사용자 인터페이스 메타포어 선정, 항해 트리(Navigation Tree) 설계, 그리고 사용자 인터페이스 컴포넌트 설계 등의 지침들로 구성된다.

사용자 인터페이스 시나리오 선정은 요구사항 정의서의 인터페이스 요구사항을 세부적으로 이해하고, Use Case 정의서와 순차도 또는 협력도를 기반으로 사용자가 원하는 시스템의 사용 시나리오를 작성하는 것이다. 이때, 목표 시스템의 주화면이나 메인 메뉴를 가정하여 사용자와 시스템간의 사건 흐름들을 순차적으로 정의한다.

사용자 인터페이스 메타포어 선정은 시스템과 사용자 간의 자연스럽고 효과적인 정보교류를 제공하기 위한 방식, 장치, 전략 및 수단인 메타포어를 정의하는 것이다. 예를 들어, 풀-다운 메뉴, 윈도우, 책, 비즈니스, 일기장, 지도, 카드, 테이블, 폴더, 케비넷 등과 같은 것이 메타포어의 예제이다. 사용자 인터페이스 항해 트리는 정해진 메타포어에서 사용되는 사용자 인터페이스 컴포넌트들 간의 모든 가능한 항해의 경로를 표시하는 것이며, 사용자 인터페이스 컴포넌트 설계는 사용자 인터페이스 각각의 컴포넌트에 대한 설계를 하는 것이다. 예를 들어, 윈도우 컴포넌트의 경우는 윈도우 유형, 제목, 초기 크기, 경계 색, 배경색 등을 결정하는 것이다[18].

3.2.4 구현 단계(400)

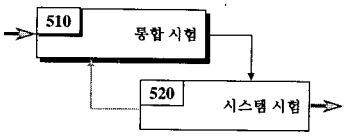
설계 모델을 기반으로 객체지향 언어 및 도구들을 활용하여 목표 시스템을 구축하는 단계로서, 사용자 인터페이스 모듈, 어플리케이션 모듈, 그리고 데이터베이스 모듈을 구현한다. 이 단계는 특정 언어나 개발 도구의 특징 및 제약 사항들을 고려해야 한다.



(그림 5) 구현 단계
(Fig. 5) Implementation Phase

3.2.5 시험 단계(500)

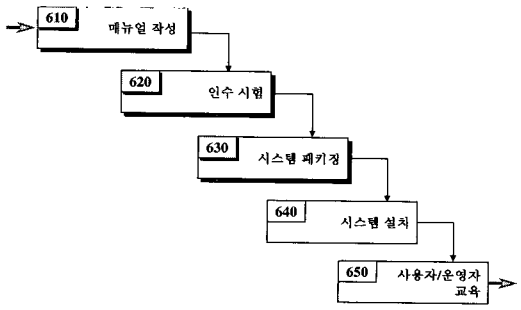
시험 단계는 구현된 프로그램에 대해 통합 시험과 시스템 시험을 객체지향 방식으로 적용하며, 필요한 수정 및 보완을 한다. 통합 시험은 통합된 시스템에 대한 품질 측정 및 시험을 통하여 종합적인 평가를 하는 과정으로서, 객체지향 통합 시험에서는 Use Case 단위로 통합 시험을 실시하게 되는 데, 점차 Use Case 들을 통합하면서 서로 다른 Use Case들이 적절히 작동되는지 시험한다.



(그림 6) 시험 단계
(Fig. 6) Testing Phase

3.2.6 보급 단계(600)

보급 단계는 시험이 완료된 소프트웨어를 요구되는 형태로 패키징하며, 사용자 매뉴얼 등과 함께 사용자에게 제공한다.



(그림 7) 보급 단계
(Fig. 7) Deployment Phase

4. 사례 연구

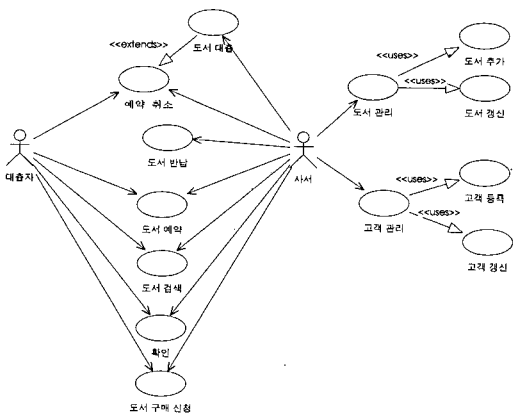
본 장에서는 본 논문에서 제시한 개발 방법론을 이용하여 도서 관리 시스템 구축에 적용한 사례를 제시하고, 제안된 방법론을 기존의 방법론들과 비교 및 평가한 결과를 제시한다.

4.1 실 험

도서 관리 시스템 구축을 하는 데 있어서 본 논문에서 제안한 방법론의 단계 및 업무별 지침을 적용하였다. 각 단계별로 사례 연구 결과들을 살펴보면 다음과 같다.

4.1.1 계획 단계

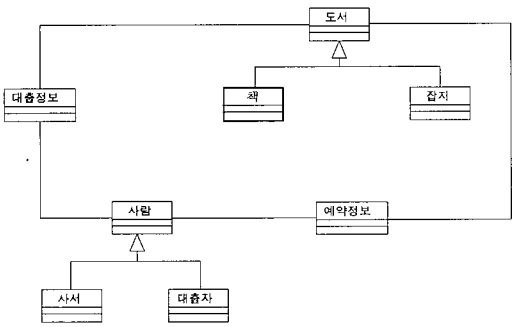
계획 단계는 프로젝트에 대한 목적 및 범위를 이해하고, 시스템의 모든 요구 사항들을 추출하기 위한 단계이다. 계획 단계의 첫번째 업무는 요구사항 명세서를 작성하는 업무이고, 이를 바탕으로 보다 명확하고 분명하게 도메인을 이해하고 개발할 시스템의 주요 기능들을 식별하기 위해 도메인 모델링을 한다. 본 사례 연구에서는 도메인 모델링을 위해 도서관리 시스템에 대한 Use Case 다이어그램을 작성하고, 각각의 Use Case 별로 Use Case 정의서를 작성하였다. Use Case 정의서는 분석 단계의 220. 메시지 흐름 분석 업무에 입력물로 사용된다. 도서 관리 시스템의 주요 기능들은 (그림 8)에 표현되어 있다. 본 시스템의 행위자(Actor)로는 대출자와 사서가 있으며, 도서 관리나 고객 관리 기능은 사서만 접근할 수 있다. 그리고, 도서 대출과 예약



(그림 8) Use Case 다이어그램
(Fig. 8) Use Case Diagram

취소간의 관계는 extends 관계를 갖는다. 그 이유는 도서 대출시 예약된 도서를 대출하는 경우는 예약을 취소해야 하기 때문이다.

요구 사항 정의서와 Use Case 정의서를 토대로 도메인에 속하는 개념들을 추출하여 개략 클래스 다이어그램을 작성하게 된다. 개념들을 추출하기 위해서는 요구사항 정의서와 Use Case 정의서에 있는 명사나 명사구들을 식별하고, 해당 도메인에 대한 지식을 바탕으로 개념들을 식별한다. 이러한 과정을 통해 산출된 것이 개략 클래스 다이어그램으로서 이에 대한 예가 (그림 9)에 나타나 있다.

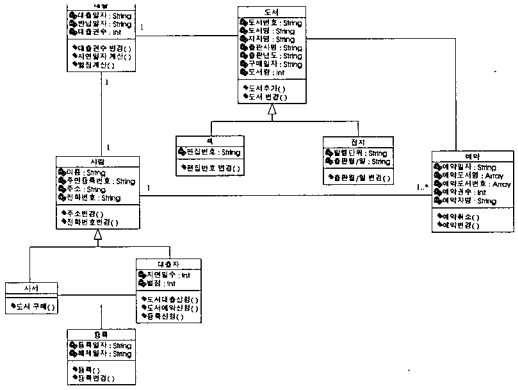


(그림 9) 개략 클래스 다이어그램
(Fig. 9) Conceptual Class Diagram

(그림 9)에서 보이는 것처럼 개략 클래스 다이어그램에는 일부 클래스들과 클래스들 간의 관계만을 표현하였다.

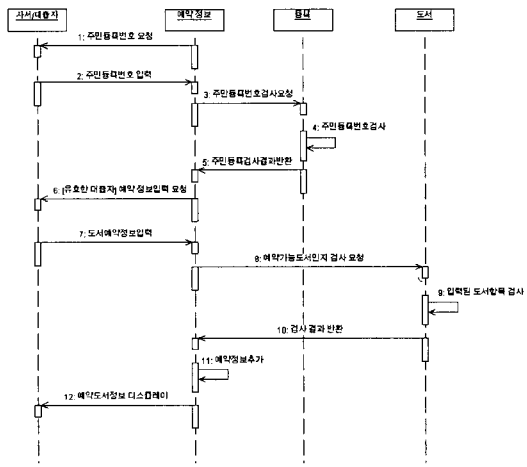
4.1.2 분석 단계

분석 단계는 도메인에 대해 이해를 하고 시스템이 수행해야 할 것이 무엇인지를 파악하기 위해 시스템을 3가지 측면인 정적, 동적, 그리고 기능적인 관점에서 모델링하게 된다. 정적인 관점에서 모델링하기 위해서는 210. 객체 구조 분석 업무를 통해 클래스 다이어그램을 작성하게 되는데, 이는 계획 단계의 개략 클래스 다이어그램을 보다 세부적으로 표현하는 것이다. 즉, 클래스의 속성을 추가하고 클래스들 간의 다중성을 추가하게 된다. 이후에 220. 메시지 흐름 분석이나 230. 객체 행위 분석을 통해 클래스의 메소드를 추출하여 이 클래스 다이어그램에 함수들을 추가하게 된다. 객체 구조 분석을 통해 생성된 명세 클래스 다이어그램이 (그림 10)에 제시되고 있다.



(그림 10) 명세 클래스 다이어그램
(Fig. 10) Specification Class Diagram

동적인 관점에서 도메인을 모델링하기 위해서 순차도를 통해 객체들 간의 메시지 흐름을 분석하였는데 이를 위해서는 계획 단계의 Use Case 다이어그램과 Use Case 정의서를 입력물로 사용하였다. 순차도는 Use Case 정의서에 있는 시나리오를 기반으로 작성하게 되는데 이에 대한 예가 (그림 11)에 제시되어 있다.

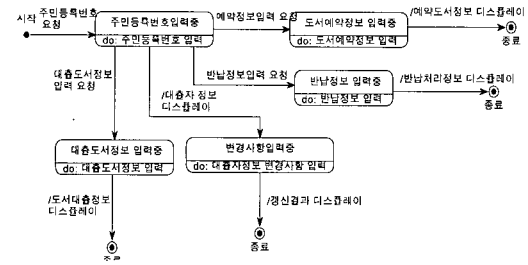


(그림 11) 순차도
(Fig. 11) Sequence Diagram

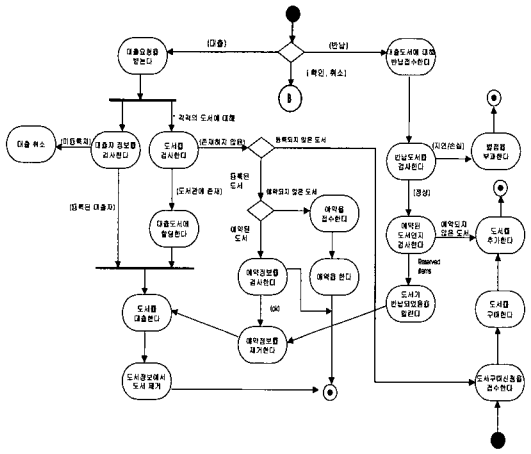
그리고, 객체 행위를 분석하기 위해서는 상태 전이도를 사용한다. 상태 전이도는 시스템 내에 있는 클래스들의 상태 변화를 모델링하는 것으로서, 각 클래스별로 상태 변화를 모델링한다. 이를 위해서는 이전 업무의 순차도를 기반으로 클래스에 전송되는 메시지들이

상태 전이도의 전이(Transition)가 되며 클래스가 전송하는 메시지들이 상태 전이도의 상태로 대응 된다. 도서 관리 시스템에서 '사서' 클래스에 대한 상태 전이도가 (그림 12)에 제시되어 있다.

기능적인 관점에서는 활동도를 이용하여 시스템 전체의 작업 흐름을 모델링 하였고, 활동(Activity)들 간의 순서를 제시할 수 있는데, (그림 13)에 활동도에 대한 예가 제시되어 있다.



(그림 12) '사서' 클래스에 대한 상태 전이도
(Fig. 12) State Diagram

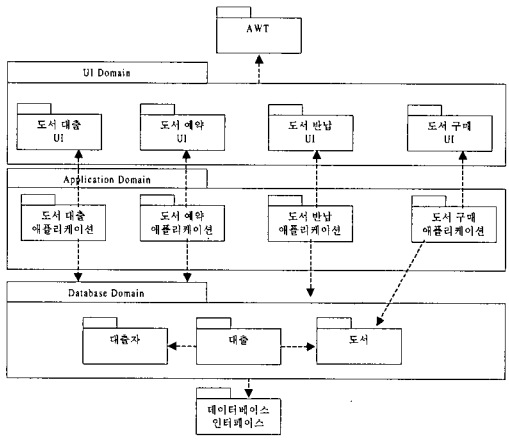


(그림 13) 활동도
(Fig. 13) Activity Diagram

4.1.3 설계 단계

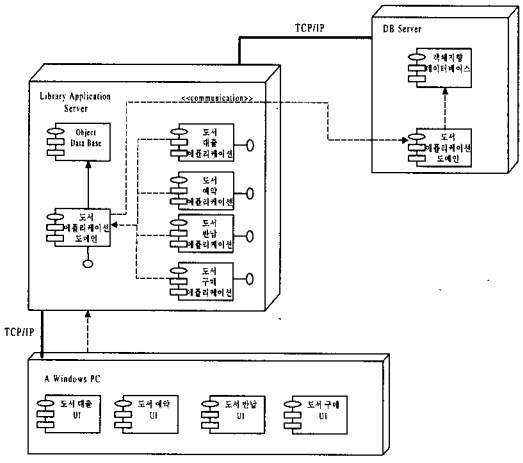
설계 단계에서는 논리적 시스템 설계 및 물리적 시스템 설계를 패키지 다이어그램과 Deployment 다이어그램을 이용하여 설계하고, 구현 시 사용할 언어와 데이터베이스 등을 고려하여 클래스 다이어그램을 보다 상세화 시킨다. 또한 사용자 인터페이스 부분을 설계한다.

논리적 시스템 설계에서 클래스들을 패키지 다이어그램을 통해 그룹화 한 결과가 (그림 14)에 제시되고 있다.



(그림 14) 패키지 다이어그램
(Fig. 14) Package Diagram

패키지 다이어그램을 통해 논리적 시스템을 설계한 결과를 가지고 시스템 아키텍처를 선정하여 각각의 패키지들을 물리적 노드로 할당하였는데 이것이 물리적 시스템 설계이다. 물리적 시스템 설계는 시스템의 하드웨어 장치를 고려하여 노드별로 표현하는 Deployment 다이어그램을 통해 모델링하였다. (그림 15)에 Deployment 다이어그램을 통한 물리적 시스템 설계 결과가 제시되고 있다.



(그림 15) Deployment 다이어그램
(Fig. 15) Deployment Diagram

사용자 인터페이스 설계에서는 이 도서 관리 시스템을 사용할 시나리오와 메타포어 선정, 항해 트리 설계,

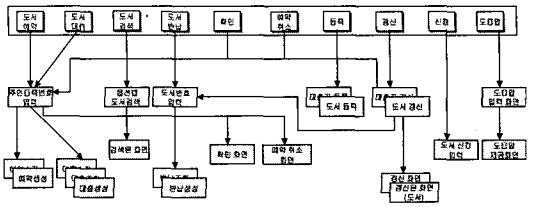
그리고 컴포넌트 설계등을 하였다. 사용자 인터페이스 시나리오를 작성할 경우는 Use Case 정의서의 시나리오를 기반으로 작성하였다. (그림 16)에 사용자 인터페이스 시나리오 예가 제시되어 있다.

- 시나리오#1
  - 화면에 도서예약, 도서 대출, 도서 검색, 도서반납, 확인, 예약취소, 등록, 갱신, 신청 메뉴가 뜬다.
  - 대출자 서 세원이 도서 예약 메뉴를 선택한다.
  - 서 세원이 주민등록번호: '551212-2936461'를 입력하고 확인 버튼을 누른다.
  - 시스템이 도서예약화면을 띄운다.
  - 서 세원이 도서명: '천년의 사랑', 도서번호: '1111', 성명: '서세원', 예약일자 : '98/3/37'을 입력하고 예약 버튼을 누른다.
  - 화면에 예약된 내용이 디스플레이 된다.
  - 서세원이 내용을 확인하고 종료 키를 누른다.

(그림 16) 사용자 인터페이스 시나리오
(Fig. 16) UI Scenario

사용자 인터페이스 시나리오가 선정되면, 사용자 인터페이스 메타포어를 선정하게 되는 데 도서 관리 시스템 예제에서는 사용자 인터페이스 메타포어를 윈도우 메타포어로 선정하였다.

사용자 인터페이스 메타포어가 선정되면, 사용자 인터페이스 항해 트리를 작성하게 되며, 그 이후 사용자 인터페이스 컴포넌트 설계를 하게 된다. 사용자 인터페이스 항해 트리 설계 결과가 (그림 17)에 제시되어 있고, 사용자 인터페이스 컴포넌트 설계 결과가 (그림 18)에 제시되어 있다.



(그림 17) 사용자 인터페이스 항해 트리
(Fig. 17) UI Navigation Tree

	폰 트	바탕색	크 기
전체바탕		연녹색	
명령버튼	굴림체	짙은 분홍, 짙은회색	가로76, 세로37
도 움 말	굴림체	회색	가로400, 세로250
제 목	굴림체	검정색	

(그림 18) 사용자 인터페이스 컴포넌트 설계도
(Fig. 18) UI Compoment Design

4.2 평 가

본 논문에서 제시한 방법론은 기존의 UML을 기반으로 확장한 방법론으로서, 기존의 많은 객체지향 방법론들과 비교한 평가 내역은 <표 1>과 같다. <표 1>에서 보이는 것처럼, 기존의 대부분의 객체지향 방법론들은 소프트웨어 전체 생명주기를 지원할 수 있는 프로세스가 정의되어 있지 않고, 일부 단계에 대해서만 정의되어 있다. 따라서, 방법론의 지원 범위가 특정 단계에 치중되어 있으며, 단계별 업무의 정의나 업무들 간의 작업 흐름이 매끄럽게 정의되어 있지 않다. 반면에

〈표 1〉 국외 방법론과의 비교 평가
〈Table 1〉 Comparison of Method Ologies

방법론 유형 / 비교 항목	Rumbaugh's OMT	Booch's OOAD	Jacobson's OOSE	UML	제안한 방법론
프로세스 정의	분석/설계	Macro/Micro 프로세스 정의	일부 지원 (분석에 치중)	개략적 정의 (Unified Process)	전체 생명주기 지원
단계/업무 정의	분석/설계만	개략적	개략적	개략적	3단계 정의 (단계, 업무, 지침)
지원 범위	분석에 치중	설계에 치중	분석에 치중	분석/설계에 치중	계획, 분석, 설계, 시험, 보급 지원
문서화	산출물에 국한	산출물에 국한	산출물에 국한	산출물에 국한	표준양식, 표준항목, 표기법으로 구분하여 표시
적용성	요구사항이 명확한 경우에 적합	실시간/대규모 프로젝트에 적합	요구사항이 불명확한 도메인에 적합	다양한 도메인에 적용가능	다양한 도메인에 적용 가능

본 논문에서 제안한 방법론은 전체 생명주기를 지원하는 프로세스를 정의할 뿐만 아니라, 단계 정의, 그리고 단계별 업무의 정의로 인해 체계적으로 소프트웨어를 개발할 수 있도록 지원한다.

5. 결론 및 향후 연구과제

본 논문에서 제시한 방법론은 UML을 실무에 효율적으로 적용할 수 있도록 확장 적용한 것이다. 따라서, 계획에서부터 보급에 이르는 전 과정을 지원하는 프로세스를 정의했고, 각각의 단계별로 필요한 업무들과 업무들간의 순서 흐름과 지침들을 제시함으로써, 개발자로 하여금 쉽고 편리하게 업무를 수행할 수 있도록 지원했다. 많은 개발자들이 UML에서 제공하는 많은 기법들을 개발 단계의 적용시점을 찾는데 많은 어려움을 지녔고, 기존의 대다수의 방법론들이 요구하는 비대한 문서화로 인해 방법론 적용을 기피해왔다. 본 논문에서 제안한 방법론을 사용함으로써 개발자들이 단계별 또는 업무별로 필요한 기법들을 쉽게 적용할 수 있을 뿐만 아니라 간결한 문서화로 인해 방법론 사용의 부담을 줄일 수 있을 것이다.

향후 연구과제로는 본 방법론에 대한 자동화 도구의 지원 및 소프트웨어 재사용성을 향상 시키기 위한 프레임워크 개발 방법론으로 통합 및 확장하는 것이다.

참 고 문 헌

[1] R. Orfali, D. Harkey, J. Edward, 'The Essential Distributed Object survival guide,' Jon Wiley & Sons: New York, NY, 1997.

[2] R. Orifali, D. Harkey, 'Client/Server Programming with JAVA and CORBA,' Jon Wiley & Sons: New York, NY, 1997.

[3] Bruce Eckel, 'Thinking in Java,' http://www.EckelObjects.com/Eckel/, Aug. 1997.

[4] OMG, 'UML Specification version 1.1,' Object Management Group, November 1997.

[5] Martin Fowler, 'UML DISTILLED: Applying the Standard Object Modeling Language,' Addison-Wesley, 1997.

[6] James Rumbaugh et. al., 'Object-Oriented Modeling and Design,' Prentice Hall, 1991.

[7] Grady Booch, 'Object-Oriented Analysis and Design with Applications,' Benjamin/Cummings, 1994.

[8] Ivar Jacobson, 'Object-Oriented Software Engineering - A Use Case Driven Approach,' Addison-Wesley, 1992.

[9] Roger S. Pressman, 'Software Engineering - A Practitioners Approach,' McGraw Hill, 1996.

[10] Craig Larman, 'Applying UML and Patterns - An Introduction to Object-Oriented Analysis and Design,' Prentice Hall, 1998.

[11] Cho, Eun Sook, Kim, Chul Jin, Kim, Soo Dong, and Rhew, Sung Yul, "Static and Dynamic Metrics for Effective Object Clustering," Asia-Pacific Software Engineering Conference(APSEC'98), Taipei, Taiwan, Dec. 1998.

[12] Yang, Young Jong, Kim, Soon Yong, Choi, Gui Jai, Cho, Eun Sook, Kim, Chul Jin, and Kim, Soo Dong, "UML-based Object-Oriented Framework Development Methodology," Asia-Pacific Software Engineering Conference(APSEC'98), Taipei, Taiwan, Dec. 1998.

[13] Cho, Eun Sook, Kim, Soo Dong, Rhew, Sung Yul, Lee Sang Duck, and Kim, Chang Gap, "Object-Oriented Web Application Architectures and Development Strategies," Asia-Pacific Software Engineering Conference(APSEC'97) and International Computer Science Conference(ICSC'97), Hong Kong, pp.322-331, Dec. 1997.

[14] Rational, 'Objectory Process 4.1,' Rational Software Corporation, 1997.

[15] Rational, 'Unified Process 5.0,' Rational Software Corporation, 1998.

[16] 김철진, 조은숙, 김수동, "효율적인 객체지향 설계 및 성능 측정을 위한 정적/동적 메트릭", 한국정보과학회 논문지(B), 제25권 제11호, pp.1657-1666, 1998.

[17] 김동관, 김수동, "클라이언트 서버 및 분산 소프트웨어 개발을 위한 객체 클러스터링 기법", 한국정보과학회 논문지(B), 제25권 제7호, pp.1053-1063, 1998.

[18] 송치양, 김혁만, 신의섭, 김수동, "전사적 통합 소프트웨어 개발 방법론 구축", 한국정보과학회 논문지(C), 제4권 제1호, pp.63-47 1998.

[19] 김수동, "Java 기반 인터넷 어플리케이션 아키텍쳐 및 설계 기법", 한국정보과학회 학회지, 제16권 제4호, pp.9-15, 1998.

[20] 김수동, 김철진, "객체지향 클라이언트/서버 시스템 아키텍쳐", 한국정보처리학회 학회지, 제4권 제6호, pp.16-29, 1997.

조 은 숙

e-mail : escho@selab.soongsil.ac.kr

1993년 동의대학교 전산통계학과 졸업(이학사)

1996년 숭실대학교 대학원 전자계산학과 졸업(공학석사)

1996년~현재 숭실대학교 대학원 전자계산학과 박사과정

관심분야 : 객체지향 개발방법론, 객체지향 재사용 기법(Framework, Component-Based Software Engineering), 분산 객체 컴퓨팅(CORBA, Client-Server Web)

김 수 동

e-mail : sdkim@computing.soongsil.ac.kr

1984년 전산학 학사, Northeast Missouri State University

1988년 전산학 석사, The University of Iowa

1991년 전산학 박사, The University of Iowa

1991년~1993년 한국통신 연구개발단 연구실장/선임연구원

1993년~1994년 The University of Iowa, 교환교수

1994년~1995년 현대전자 S/W연구소 책임연구원

1995년 9월~현재 숭실대학교 컴퓨터학부 조교수

관심분야 : 객체지향 개발방법론, 객체지향 재사용 기법(Framework, Component-Based Software Engineering), 웹 기반 분산 객체 컴퓨팅(CORBA, Intranet, Client-Server Web, Web Agents), 인터넷 상거래 시스템 기술

류 성 열

e-mail : syrhew@computing.soongsil.ac.kr

1997년 2월 아주대학교 컴퓨터공학부(공학박사)

1997년 3월~1998년 2월 George Mason University 교환교수

1981년 3월~현재 숭실대학교 컴퓨터학부 교수

1998년 3월~현재 숭실대학교 정보과학대학원 원장

관심분야 : 리엔지니어링, 분산객체컴퓨팅, 소프트웨어 재사용