# 客體指向 시스템 設計에 관한 小考

安重鎬* · 洪賢基** · 金範洙***

《目 次》



본 연구에서는 정보시스템을 보다 효율적으로 구축하기 위하여 客體指向接近法을 이용한 7단계의 시스템 設計 方法을 제시하였다.

① 시스템 要求 收容
② 客體 및 서어비스의 定義
③ 客體間 相互關係 設定
④ 客體의 詳細한 內容 定義
⑤ 클래스라이브러리 檢討 및 下部시스템 構築
⑥ 클래스 位階圖表 作成
⑦ 시스템 具現을 위한 最終設計

1단계는 시스템 설계와 연계되는 단계이고 2,3,4,5,6단계는 본격적인 시스템 분석단계인데 이 단계는 상호반복적으로 분석이 이루어진다. 7단계는 시스템 구축을 위해 분석과 구축이 서로 연결되는 단계이다. 또한, 각 개발단계가 하나의 패러다임내에서 추진되므로 壽命週期別로 다른 패러다임을 이용하는 여타의 방법론보다 효과적으로 시스템을 구축할 수 있다.

이러한 방법론을 이용하여 정보시스템을 개발할 경우 객체지향접근법의 장점인 소프트웨어 시스템개발 소요시간·비용 절약, 소프트웨어 중복개발방지 및 재사용, 보다 복잡하고 광범위한 컴퓨터환경에서 시스템 構築 등의 효과가 있다.

* 서울大學校 經營大學 敎授
** 가하 컨설팅(대성산업) 理事
*** 서울大學校 經營學 碩士

# 1. 서 론

## 1.1. 硏究의 背景

최근 利用者의 복잡한 要求事項에 副應하는 소프트웨어 시스템을 開發, 維持하기 위한 必要性이 커져서 소프트웨어 設計와 開發에 관한 연구들이 활발하게 진행되고 있다. 그중에서 古典的 壽命週期 接近法(classical life cycle approach)인 構造的 分析·設計(structured system analysis & design) 方法 등을 비롯한 기존의 개발방법들은 소프트웨어의 再使用, 각 段階別 履行이나 維持補修 등에 많은 어려움이 있다.

이러한 가운데 도입된 客體指向패러다임(object-oriented paradigms)은 實世界(real world)의 문제를 客體(object)라는 매개를 이용하여 컴퓨터세계의 해결영역에서 자연스럽게 표현할 수 있다. 〔Henderson-Sellers, 1990〕 그러므로, 이러한 패러다임(paradigm)내에서 소프트웨어 시스템을 개발하는 경우에 소프트웨어 개발자는 再使用性, 擴張性 등 여러가지 長點을 享有할 수 있다.

정보시스템 관리자의 중요한 역할 중 하나는 다양한 목표시스템의 이용자가 원하는 바를 가능한 한 빠른 시일내에 그리고 정확하게 달성할 수 있도록 시스템 개발 프로젝트를 관리하는 것이다. 올바른 分析과 設計는 이러한 사용자의 요구를 그대로 반영할 수 있는 지름길이며 이 단계에서 시스템을 정확하게 분석하면 차후에 생기는 문제점과 비용을 최소화시킬 수 있기에 그 중요성은 매우 크다고 할 수 있다. 그래서, 이 연구에서는 客體指向 소프트웨어 시스템 개발시 設計段階의 指針과 시스템 導入을 보다 효과적으로 유도할 수 있는 客體指向設計 技法을 제시하고자 한다.

## 1.2. 硏究의 目的

효과적으로 시스템을 구축하기 위해서 분석-설계-구현의 시스템개발수명주기(SDLC)단계 중 가장 큰 비중을 가지는 단계가 분석과 설계 단계이다. 〔안중호, 1990〕 이러한 분석 및 설계는 과거에 구조적 기법의 경우에서와 마찬가지로 객체지향패러다임에서도 객체지향 프로그래밍, 객체지향 설계, 객체지향 분석의 順으로 연구가 진행되어왔다. 구조적기법이 착실한 이론적 기반하에서 출발한 것임에 반해 객체지향개념은 주로 현장의 엔지니어, 설계자, 이용자 등에 의해 그 필요성이 인식되었고 이에 대한 꾸준한 연구가 이루어져왔다.

객체지향설계는 1986년 Booch가 그 개념을 정의하고 방법론을 제시한 이래 많은 연구가 발표되었으나 대부분의 방법론이 Ada를 위주로한 객체지향 프로그래밍 언어를 위한 설계방

법이거나 과거의 구조적 기법에 객체지향의 특성을 가미한 방법론에 불과하였다. 이러한 복합적 개발방법론이[1] 경우에 따라서는 객체지향법을 널리 보급하는데 장애가 되어온 것이 사실이다. 즉 이러한 방법론은 실제 현장에서 받아들여지지 못하고 있고, Yourdon 등이 객체지향방법론은 대규모 프로젝트에 이용할 수 있는가에 대해 강한 의문을 제기하는 결과를 낳았다. [OOPSLA '90] 그래서 이 논문에서는 객체 지향접근법의 특성을 기본으로 하여 대규모 시스템 개발에 적합한 객체지향설계방법론을 제시하고자 한다.

그리고, 이러한 객체지향접근법이 보다 빨리 보급되어 현재 기업환경에서 어려움을 겪고 있는 시스템개발 프로젝트 관리자, 프로그래머, 분석가, 설계자 등의 어려움을 덜어줄 수 있도록 MRP 시스템 구축을 위한 시스템 設計를 사례로 들어 연구를 진행하였다.

### 1.3. 硏究의 意義

'소프트웨어 危機'[2]라는 말이 나온 이래로 구조적 기법등 여러가지 시스템개발방법이 소개되었으나 대부분의 기업에서는 빠르게 변화하는 환경에서 신속하게 적합한 시스템을 구축하는데 실패하였고 전산부서가 밀린일감(backlog)을 해결하지 못하는 한계를 내보였다. 객체지향접근법은 이러한 한계상황을 극복할 수 있는 하나의 방법론이라고 볼 수 있다.

이러한 객체지향 패러다임에서 출발한 객체지향설계방법을 연구하는 의의는 첫째, 이러한 종합적인 연구가 지금까지 있어온 객체지향설계의 혼돈을 어느 정도 정돈시킬 수 있다. 둘째, 설계방법이 단계적이고 세부적인 사항을 차례로 제시하였으므로 이러한 설계방법론에 따른 시스템 개발도구(CASE tool) 제작을 위한 기초를 제공할 수 있다. 셋째, MRP 시스템을 객체지향접근법의 적용사례로 들어 객체지향접근법이 더 이상 컴퓨터공학이나 소프트웨어공학의 학문적 탐구대상이 아니라 실제 기업환경에서도 활용할 수 있음을 보여준다. 넷째, 현재까지 시스템을 구축하면서 발생하였던 분석·설계 단계의 문제점을 해결할 수 있다.

## 2. 시스템 設計에 관한 硏究

### 2.1. 客體指向設計

시스템 설계에 대한 기본적인 이해를 도울 수 있도록 2.1에서는 객체지향설계, 객체지향

---

1) 복합적 개발방법이란 구조적접근법에서 사용하는 방법을 객체지향속성에 맞추어 수정한 방법론을 말한다.

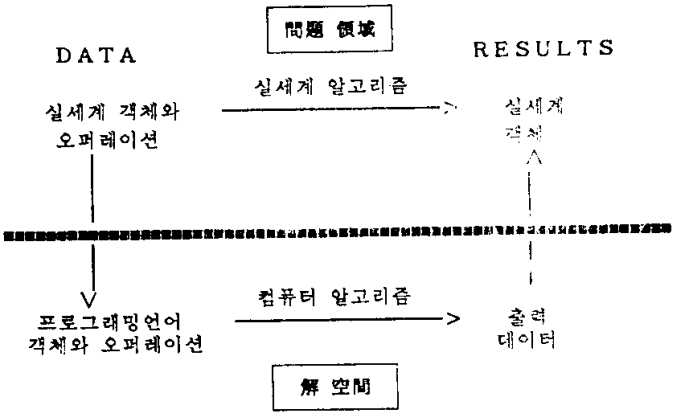
2) 소프트웨어 危機(software crisis)는 컴퓨터소프트웨어 개발에서 당면하는 문제들을 말한다. 문제는 개발일정 및 비용, 개발자의 생산성, 소프트웨어 품질 등에 관한 것이다.

설계과정, 도형화 기법 등에 대해 살펴본다.

2.1.1. 客體指向設計

정보은폐, 자료추상화 등을 기초개념으로 하는 客體指向設計는 프로그램의 단위를 객체(object)로 간주하고 이들 사이의 관계를 설정하는 과정이다.

이러한 客體指向設計도 다른 정보지향설계방법과 마찬가지로 실세계 문제 영역에 관한 표현을 생성하며 이 표현을 소프트웨어 해결 영역인 컴퓨터로 대응시킨다. [Pressman, 1987] 이 과정을 그림으로 나타내면 〈그림 2-1〉과 같다.



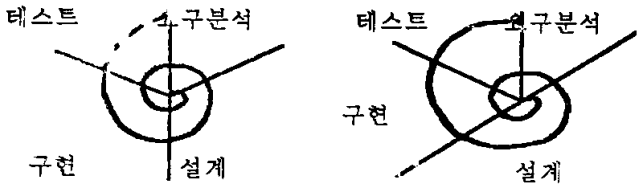

〈그림 2-1〉 問題領域과 解空間

자료원 : Henderson-Sellors[1990] 再引用

객체지향에 관한 정의는 관점에 따라 여러 문헌에서 서로 다르게 사용하고 있다. '객체지향(object-oriented)'이라는 용어가 사람마다 다른 의미로 표현되어 논란의 대상이 되고 있으나 대개 모듈화, 정보은폐, 캡슐화, 추상자료형 등의 용어와 동의어로 간주된다.

시스템설계를 위한 방법론으로 客體指向設計를 할 때, 설계자가 직면하는 어려움은 요구사항명세서가 객체지향적인 방법으로 이루어지지 않을 수 있다는 것과 이러한 명세서로부터 적절한 객체와 관련 연산을 결정하는 설계를 유도하는 방법 등에 관한 것이다. 이러한 점은 설계과정 자체가 설계자의 숙련도, 교육정도 및 직관과 경험에 의존하는 창조적인 과정이어서 공식과 같은 형태로 표현될 수 없기 때문이다. [Sommerville, 1989] 따라서, 설계자는 가능한 한 다수의 설계를 가정하고 이를 테스트하여 인정할 만한 해가 발견될 때까지 이러한 과정을 반복하여야 한다.

客體指向設計에서 설계가 차지하는 역할은 과거의 구조적 기법에 의한 소프트웨어개발자보다 훨씬 커졌다. 과거의 소프트웨어 개발수명주기(SDLC)와 객체지향소프트웨어 수명주기를 비교한 〈그림 2-2〉에서 이를 확실히 알 수 있다.




(a) 전통적 수명주기 (b) 객체지향소프트웨어 수명주기

〈그림 2-2〉 소프트웨어 설계의 비중
자료원 : Wirfs-Brock[1990, pp. 18-19]

2.1.2. 客體指向設計過程

客體指向設計에서 얻을 수 있는 가장 큰 성과물은 클래스 位階(a hierarchy of classes)이다. 각각의 클래스는 데이터구조(data structure)와 통제(control)를 함께 가지고 있는 모듈(self-contained module)이라고 볼 수 있다. 문제 영역은 객체의 집합 즉, 클래스와 이와 관계된 method로 보다 자연스럽고 현실감있게 표현할 수 있다.

客體指向設計의 가장 기본적인 요소는 객체(object)이다. 그 다음에는 이 객체들의 공통점을 추출해서 클래스라는 단위로 묶게 되면 이 클래스는 보다 추상적인 클래스의 하위클래스가 된다. 클래스의 가장 추상화된 단계는 일반적으로 프레임웍(framework)이라고 한다. 프레임웍은 관련된 애플리케이션(applications)의 그룹에 맞는 설계를 표방하는 클래스의 집합이다. 구조적 접근법과 이의 구조도표(structure chart)에서는 애플리케이션을 '자료를 공유하는 서로 독립적인 모듈의 순서적인 집합'으로 정의하지만 객체지향접근법에서는 애플리케이션을 '외부에서 내부를 제한적으로 볼 수 있는 서로 의사소통을 하는 자료와 통제를 내부에 포함하고 있는 모듈의 집합'이라고 정의한다.

본질적으로, 客體指向設計는 다음의 4단계를 거쳐서 이루어진다.

1) 객체와 클래스의 정의
2) 클래스간의 관계 작성
3) 클래스위계에서 프레임웍의 작성
4) 재사용 가능한 클래스 라이브러리와 애플리케이션 프레임웍 수립

객체지향접근법은 原型接近法(prototyping approach)과 마찬가지로 이러한 設計段階가 순환적으로 이루어진다. 설계하는 방법은 시스템 설계자나 프로그래머가 일단의 클래스를 정의하는 것으로 설계를 시작하여 점차 이를 확장하고 장차 애플리케이션의 프레임웍으로 모아가게 된다. 최종 사용자와의 상호작용은 이러한 原型(prototype)을 수정하도록 하고, 그 다음에는 설계자와 프로그래머가 다시 재설계 및 프로그래밍을 하고, 또 다시 이러한 과정

을 반복하게 된다. 이러한 순환과정에서 클래스가 보다 엄밀하게 정의되고 조직되어서 기본적인 라이브러리(standard library)에 추가되고 이러한 클래스가 필요할 때 그 애플리케이션에서 쉽게 다시 이용할 수 있는 프레임웍 설계자는 애플리케이션간의 유사점에 관심을 두고 차후에 비슷한 문제에 봉착하였을 때를 대비하여 클래스를 기준으로한 프레임웍을 개발한다.

지금까지는 객체지향설계에 대해 개괄적으로 알아보았고, 다음의 2.2에서는 Booch, Shlaer & Mellor, OOSD 등에 대해 언급하고 새로운 객체지향설계 방법론을 소개한다.

### 2.2. 客體指向設計 方法論

#### 2.2.1. Booch의 방법

Booch의 방법은 Ada개발에 많이 응용되었으며, 문서화된 시스템명세(textual specification)나 비공식적 설계(informal design)에서 도출한 것으로 다음과 같이 요약할 수 있다. [Booch, 1986].

첫째, 객체들과 이의 특성을 확인한다. 실제 모델에서의 역할과 문제영역에서의 actor, server, agent를 인지한다.

둘째, 각 객체들간에 상호 영향을 미치는 오퍼레이션을 확인한다. 각 객체들의 행동(behavior)이나 class의 특성을 기술한다. 즉, 각 객체들간에 수행될 오퍼레이션을 결정함으로써 객체의 정적 의미(static semantics)를 설정하고 시간·공간상의 제약조건을 확인함으로써 동적 기능(dynamic behavior)을 설정한다.

셋째, 다른 객체들과의 관계를 고려하여 각 객체들간의 可視性을 설정한다. 즉, 객체들이나 객체의 class 내에 정적종속성을 확인하는 것으로 실제모델로부터 객체의 位相(topology)을 설정하기 위한 것이다.

넷째, 각 객체의 인터페이스를 설정한다 적절한 표기법을 사용하여 모듈 사양서를 생산하는 것으로 전 단계에서 확인된 각 객체나 class간의 연락자로서 의미를 갖는다. 즉, 인터페이스는 내적측면과 외적측면간의 경계를 형성한다.

다섯째, 각 객체를 구현한다. 객체나 객체 class의 이름, 변수들의 특성 등에 대하여 적절한 표기를 선택하고 전 단계에서 설정한 인터페이스를 구현한다.

위의 개발절차는 공식절차를 위한 메카니즘으로 보이지만 자동처리절차는 아니며 설계자가 설계에 대한 많은 지식과 문제에 대한 직관적인 이해가 있어야 한다. 이 방법은 시스템명세서에 기술된 명사(noun)와 동사(verb)에 주목한다. 즉, 객체는 명사로부터 도출해내고, 객체의 operation은 동사로부터 이끌어낸다. 이때 불필요한 명사나 동사는 제외시키고 나머

지 개념들을 설계객체로 변환시키기 위한 어떤 판단기준이 필요하다. 일단 객체가 정의되면 설계는 Ada의 package 및 task 등과 관계된 표기법에 따라 도식화하여 표현한다. 이 Booch의 방법론이 객체지향접근법 발전에 근간이 되었지만 규모가 큰 시스템에는 적용하기가 곤란하고 도식화 표기법이 설명을 분명하게 하고 있지만 필요한 만큼 상세하거나 엄밀하지 못하다. 또한 객체의 위계적 분해를 도식화하여 표현할 수 있는 방법이 제시되지 않아서 이 방법이 완전한 설계표기법이라고는 볼 수 없다. [Seidewitz, 1988] Booch 방법론의 또 다른 단점은 시스템명세서와 관계된 것이다. 만약 이 시스템 명세서가 간단한 문장으로 쉽게 기술되어 있는 경우에는 아무런 문제없이 문제를 해결해 갈 수 있으나 시스템의 요구사항이 많고 복잡할 경우에는 문서의 양이 방대하기 때문에 이러한 분석기법을 이용할 수 없다. 이러한 단점은 이런 문서를 이용하는 대신 시스템 분석단계에서 명시된 DFD를 이용하는 것으로 어느 정도 해결할 수 있다.

2.2.2. Shlaer & Mellor의 방법[Shlaer, 1988]

이 방법은 시스템 분석방법과 ERD를 기초로 한 정보구조도표(information structure diagram)이라고 불리는 도형화 기법을 위주로 하고 있다. 이 표기법은 모델의 客體(object), 屬性(attribute), 關係(relationship) 등을 표기하는 데 이용된다. Shlaer & Mellor의 방법은 분석을 정보모형(information model), 상태모형(state model), 프로세스모형(process model) 등의 3가지의 모델로 나누어 설명하고 있다.

(1) 정보모형(information model)

정보모형은 객체, 속성, 관계를 정의하고 이를 그림이나 문자로 표시한다. 정보모형을 문자로 나타낼 때는 각각의 객체는 表, 각각의 인스턴스(instance)는 表의 行이 된다. 그림으로 나타낼 때는 각 개체의 전체적인 조망을 보여준다. 즉, 다음의 〈그림 2-3〉과 같이 나타낼 수 있다.

객체의 속성은 다음과 같은 규칙에 따라 정의한다.

① 각 객체의 인스턴스는 각 속성마다 하나의 값을 가진다.

```
CAR
  *title #
  *state
  ○license #
  ○manufacturer
  ○owner(r)
```

*: 사전에 결정되는 속성
○: 사전에 결정되지 않은 속성
(r): 관계를 공식화하는데 사용되는 속성

〈그림 2-3〉 Shlaer & Mellor의 客體表現方式

② 어떤 속성도 내부 자료구조를 가지지 않는다.

③ 각 속성은 객체의 식별자가 아니라 객체의 특징을 보이는 것이다.

④ 각 속성은 다른 속성의 특징이 아니라 전체 객체의 특징을 표시한다.

관계는 객체간에 연결된 화살표로 표시하며, 관계는 크게 1對 1(one-to-one), 1對多(one-to-many), 多對多(many-to-many) 관계로 나누어 표시할 수 있다.

(2) 상태모형(state model)

상태모형은 정보모형에서 정의된 객체나 관계의 행동을 표시한 것으로 정보모형의 확장이라고 볼 수 있다. 상태모형은 상태전이도표(state transition diagram, table)를 통해 설명되며 상태전이도표는 각 객체에 대한 사건명세(event list)나 메소드의 명세(method list)이다.

(3) 프로세스모형(process model)

마지막으로 개발되는 모형이 프로세스모형이다. 프로세스모형에서 각각의 메소드나 상태에 대한 데이터흐름도(DFD)가 작성된다. 각각의 프로세스를 설명하기 위해 수정된 DeMarco 도표가 사용된다.

요약하면, Shlaer & Mellor[1988]는 시스템 분석방법, 시스템 표기법, 객체지향시스템 디자인에 대한 기본적인 설명 등에 대해 언급하였다. 이 방법론은 객체를 데이터베이스의 레코드로 보고, 데이터 모형화를 위한 데이터 분석방법을 제시하고 있다. 그러나, 객체지향분석의 관점에서 보면 Shlaer & Mellor의 방법론은 객체지향의 많은 중요한 개념들을 설명하지 못하고 있다. 표기법은 기존의 구조적기법에서 사용하였던 데이터흐름도(DFD), Entity-relationship diagram, state transition diagram 등을 기본으로 하고 있으며 메소드, 메시지 같은 객체지향의 개념을 표기할 방법이 없다. 또한 데이터가 함께 포함된 프로시저란 개념이 전혀 들어있지 않고 단지 시스템 분석결과를 표기하는 방법이다.

2.2.3. OOSD: Wasserman, Pircher & Muller의 방법[Wasserman, 1989]

OOSD(Object-oriented system design) : HOOD, GOOD, MOOD[3]의 초기모형을 개선한 것으로 Coad[1989]의 방법론[4] 보다 확실한 설명을 하고 있다. OOSD는 과거의 top-down 방식과 bottom-up 방식의 시스템 구조화의 분명한 차이를 통합하려는 노력의 일환으로 相

3) GOOD: General object-oriented software development,
HOOD: Hierarchical object-oriented design,
MOOD: Multiple-view object-oriented design methodology 등의 방법은 Ada 프로그래밍 환경을 기반으로 한 소프트웨어 개발방법론이다.

4) Coad의 방법론은 subject, object, structure, attribute, service layer 등의 단계로 나누어 시스템을 분석·설계한다.

續性, 一般化, 能動客體, 受動客體 등을 포함하는 광범위한 객체지향개념을 지원할 뿐 아니라 機能的 分割槪念도 제공한다.

OOSD의 기본적인 아이디어는 다음의 네가지에서 출발한 것이다.

○구조적 설계의 구조도표

○Ada package와 task에 대한 Booch의 표기를 언어독립적으로 일반화

○클래스 位階와 相續性 原理

○동시프로그래밍(concurrent programming)을 위한 Hoare의 모니터

OOSD는 설계의 절차보다는 도형을 위주로한 표기에 중점을 두었다. 시스템 설계 방법을 시스템 설계자에게 맡겨두었기에 설계방법론이 부족한 것이 이 방법론의 단점이다.

이 방법은 다음과 같은 순서로 이루어진다.

첫째, 클래스를 정의한다. 이때 클래스는 객체에 대한 표현적 정보를 분리하여 클래스 사용자는 외부적으로 정의된 오퍼레이션만 접근할 수 있다.

둘째, 해당 클래스의 객체를 생성한다.

셋째, 클래스에 예외정보 처리에 대한 내용을 첨가한다.

넷째, 상속성과 일반성 표시

2.2.4. 7단계 客體指向設計 方法論

위에 언급한 방법들은 시스템 分析과 設計 階階를 명확하게 구분하고 있는 것처럼 보인다. 그러나 실제로 설계과정을 살펴보면 分析과 設計段階를 엄격하게 구분하지 못하고 있으며 또한 이러한 구분으로 시스템 개발상의 단계별 이행을 방해하는 측면이 지적되고 있다.

Booch의 방법론은 위에서 지적하였듯이 object-based language인 Ada를 기준으로 구축되었기에 객체지향접근법의 특징인 相續性을 적절하게 표현하지 못한다. Shlaer & Mellor의 방법과 OOSD는 기존의 구조적기법과 객체지향기법을 함께 이용할 수 있도록 고안되었지만 그 복잡성과 혼란으로 아직 널리 이용되지 못하고 있다.

그래서, 시스템 개발 각 단계의 연결을 보다 자연스럽고 원만하게 할 수 있도록 객체지향기법을 이용한 시스템 분석단계와 구현단계의 일부를 포함하는 설계방법을 제시하고자 한다.

이 설계방법은 다음의 7단계를 거쳐 이루어진다.

(1) 시스템 요구 명세서 수용

객체와 객체가 제공하는 서비스의 개괄적인 분석이 이루어지는 단계이다. 이 단계에서는 Coad[1989], Shlaer[198] 등의 시스템분석 방법을 전체적으로 적용하여 객체지향 요구명세

서를 작성한다. 객체지향 요구명세서는 개발시기, 사용할 하드웨어, 비용예산, 그리고 여러 문서들을 포함한다.

(2) 객체 및 서어비스의 定義

이 단계는 시스템 분석과 높은 수준의 설계 단계로, 객체나 엔터티와 이의 특성 및 이것이 제공하는 서어비스 등을 설정한다.

클래스를 설계하는 指針은 다음과 같다. [Korson, 1990]

① 클래스의 public interface 요소만 클래스의 오퍼레이터가 되어야 한다.

② 클래스 A의 인스턴스는 클래스 B의 요소에 바로 메시지를 보낼 수 없다.

③ 클래스의 인스턴스 사용자만이 이용할 수 있을 때만 오퍼레이터가 public 해야한다.

④ 클래스에 포함되는 각각의 오퍼레이터는 클래스 데이터에 접근 또는 수정한다.

⑤ 하나의 클래스는 가능한 한 소수의 클래스에 종속되어야 한다.

⑥ 두 클래스간의 상호작용은 명시되어야 한다.

⑦ 각각의 하부클래스는 상위클래스의 public interface가 하부클래스의 public interface가 되는 상위클래스의 특정 예로서 개발되어야 한다.

⑧ 相續構造의 최상위 클래스(root class)는 객체개념의 추상화된 모형이어야 한다.

(3) 객체간 상호관계 설정

관계(relationship)는 클래스[5]들간의 논리적인 결합이다. 각 관계는 클래스의 이름과 관련된 클래스의 쌍으로써 정의되며, 일반화·집단화·결합화의 3가지 형태로 나타낼 수 있다.

① 일반화 관계(generalization relationship)

일반화는 유사한 객체의 집단을 파악하기 위해 사용된다. 이것은 "is_kind_of", "is_a", "category", "specialization" 등의 관계로 命名된다.

일반화의 구문(syntax)은 다음과 같이 정리할 수 있다.

〈Subclass〉 is_kind_of 〈Superclass〉

〈Superclass〉 can be a 〈Subclass 1〉 or a 〈Subclass 2〉...

이러한 일반화의 관계가 클래스 상속성에 있어서 주요한 요소이다. 상위 클래스의 속성, 메소드, 관계 등이 일반화를 통해서 하부클래스에 상속된다. 이러한 상속성이 상위클래스의 속성, 메소드, 관계의 명세 및 구현에 대한 재사용을 가능하게 한다.

② 집합화 관계(aggregation relationship)

집합화는 "is_part_of"의 관계이며 객체의 그룹화를 보여주기 위해서 사용된다. 집합화에

5) 타입(type)/클래스(class)는 같은 행위나 특성을 갖는 물리적 혹은 추상적인 객체의 집합으로 묘사된다.

서 하나의 타입은 집합(assembly)의 역할을 가지고 있으며 다른 것은 요소의 역할을 가지고 있다. 각 집합의 인스턴스는 요소인스턴스의 집합으로 구성된다.

집단화의 구문(syntax)은 다음과 같이 정의할 수 있다.

〈component〉 is_part_of 〈assembly〉

〈assembly〉 contains 〈cardinality 1〉 〈component 1〉 and 〈cardinality 2〉 〈component 2〉...

집단클래스의 속성값은 해당 요소클래스 각각에 선별적으로 전달될 수 있다. 이 속성에는 轉移(transitive)와 非轉移(intransitive)屬性이 있다. 轉移屬性은[6] 전체집합 클래스와 그것의 요소 클래스 모두에 적용되고, 非轉移屬性[7]은 전체적으로 집합클래스에만 적용되는 속성이다. 3절에서 언급할 자재명세서(BOM)는 이러한 집합화의 원리를 이용하여 모형화하게 된다. 요소클래스의 메소드는 그것의 집합클래스의 메소드를 시행함으로써 선택적으로 수행할 수 있다.

③ 결합화 관계(association relationship)

결합화 관계는 使用者 定義 關係이다. 즉, 이 관계는 결합된 클래스와 관계되는 사용자 정의의 역할을 한다. 예를 들면, 사람과 자동차 사이의 결합화 관계는 사람과 관계되는 소유자의 역할과 자동차와 관계되는 재산의 역할이 연결되는 것이다.

결합화의 구문은

〈role 1〉 of 〈class 1〉 is 〈cardinality 1〉 〈class 2〉와 같이 표현된다.

일반적으로 객체지향언어가 이러한 결합화를 직접적으로 표현하지 못하므로 이 방법에서는 모델화를 생략한다.

(4) 객체의 상세한 내용을 정의

정보흐름도(information flow diagram)나 확장된 데이터흐름도(EDFD) 등을 활용하여 객체의 내부 상세한 내용을 정의한다.

각 객체의 관계에 있어서는 가시성(visibility)과 인터페이스(interface)에 대한 기술이 있어야 한다. 이 단계가 끝나면, 시스템 분석가는 객체들간의 종속성과 객체에 대한 명세서를 가지게 된다. 이 단계가 있기 이전의 세단계는 실제현상으로부터의 모델을 수립하는 것이며 이 단계 이후부터는 이전 단계에서 수립한 모델의 의미와 구현방법에 대한 記述로 이루어진다.

이 단계는 bottom-up 접근법을 따라 이루어진다고도 볼 수 있으며, 객체지향의 전략적 측면에서 이 단계가 재사용가능한 설계요소, 클래스 등을 구분하는 중요단계이다.

6) 轉移屬性은 색깔, 재료, 공정회사등을 그 예로 들 수 있다.
7) 非轉移屬性으로는 무게, 가격, 길이 등이 있다.

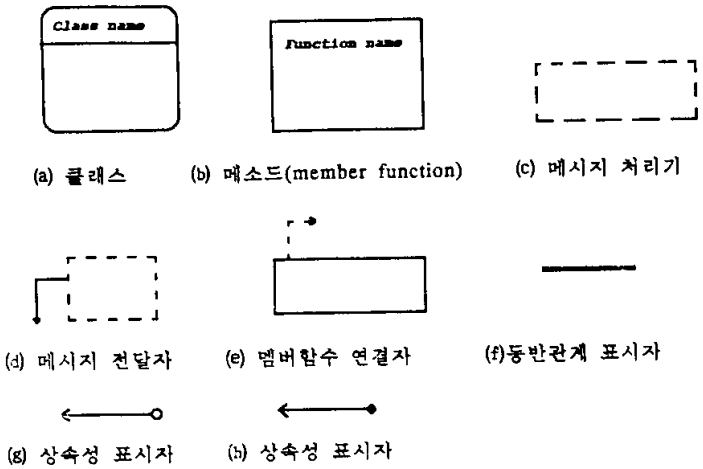
(5) 再使用部門 檢討 및 下部시스템 構築

기존의 클래스 라이브러리를 이용하여 재사용가능한 부분을 검토하고 이에 대한 응용지침을 수립하는 단계이다. 그래서 이 단계는 상향식(bottom-up)설계 방법론에 따라 이루어진다고 할 수 있다. 또한, 이 단계에서는 관련된 객체의 결합을 이용하여 새로운 하부시스템을 구축할 수도 있다.

(6) 클래스 位階圖表(class hierarchy diagram)의 提示

클래스의 통합과 일반화과정을 통해 지금까지 설계한 클래스를 위계도표를 통해 나타낸다.

위의 1단계에서 3단계까지는 거의 순차적으로 설계가 이루어진다고 볼 수 있다. 그러나 4단계부터 6단계 그리고 마지막 7단계에 이르는 과정은 반복적이고 순환적으로 이루어지는 특성을 가지고 있다.




〈그림 2-4〉 圖形 表記法

(7) 具現을 위한 最終 設計

위의 여섯단계의 설계과정을 거쳐서 실제현상(real world)의 모형이 설계된다. 이러한 설계를 바탕으로 객체지향언어를 이용하여 소프트웨어 시스템을 개발할 수 있도록 상세한 설계를 한다. 이 연구에서는 C++를 그 주대상으로 삼고 개발한다.

이러한 개발단계에서는 위에서 언급한 바와 같이 그래픽 表示方法(圖形化 技法)이 유용하게 사용될 수 있다. 즉, 이러한 접근법으로 시스템 현상을 분석한 후 이를 나타내는 그래픽 표시방법(graphic notation)은 〈그림 2-4〉와 같은 것이 사용될 수 있다.

### 2.3. 7단계 객체지향설계 方法論을 이용한 시스템 설계의 타당성

시스템 소프트웨어개발 방법의 효과성은 B. Boehm[1984]이 제시한 다음의 7가지 소프트

웨어 공학의 기준에 의해 평가될 수 있다.

① 체계적인 수명주기계획을 이용하여 관리하여야 한다.

② 제품통제를 엄정하게 하여야 한다.

③ 지속적인 검증을 수행하여야 한다.

④ 하향식(top-down) 설계를 지원하여야 한다.

⑤ 수명주기의 각 단계에 대한 명확한 결과가 있어야 한다.

⑥ 적은 인원을 보다 잘 이용하여야 한다.

⑦ 처리과정을 개선하기 위한 어떤 의도가 있어야 한다.

기존의 객체지향접근법과 비교하여 보면 위의 설계 방법론은 위의 기준을 잘 준수하고 있다. 또한 한규창[1991]의 'software 품질평가기준'에도 합당한 소프트웨어 설계방법론이다.

기존의 다른 객체지향설계법은 즉, Boock의 방법론은 언급한 바와 같이 MRPⅡ 시스템과 같은 대규모 프로젝트 설계에 적합하지 않다. Shlaer & Mellor의 방법론은 과거의 구조적인 기법에 너무 집착한 나머지 객체지향의 특성을 제대로 표시하지 못하였다. 또한, OOSD 방법론은 여러가지 방법론을 혼용하여 사용하고 세부적인 지시사항은 잘 설명하였으나 전반적인 진행단계는 제대로 설명하지 못하였다. 그러나 7단계 객체지향설계 방법은 비교적 그 방법이 간단하고, 객체지향의 속성을 가능한한 제대로 구현할 수 있도록 하며 실제 경영환경에 맞는 시스템을 구축한다는 것을 전제로 만들어졌으므로 다른 시스템 분석방법보다 이러한 MRP 시스템 구축에 보다 효과적으로 이용될 수 있다.

이러한 7단계 설계방법을 사용하였을 때, 얻을 수 있는 효과는 다음과 같다. ① software reusability 증대, ② software components redundancy 감소, ③ 기존 개발 시스템 요소 등의 활용으로 시스템 개발 壽命週期(life cycle)의 短縮, ④ 시스템 變更·更新의 便利增大, ⑤ 最終利用者를 위한 다양한 報告書의 出力, 使用者 인터페이스의 改善, ⑥ 분석·설계·구현 단계의 효과적인 연결 등의 效果를 가져올 수 있다.

다음에는 이러한 방법론으로 시스템을 개발한 사례를 들어 이 방법론의 적용 과정 및 실효성을 살펴보자.

## 3. 事例硏究

이 절에서는 N社의 사례를 통해 위에서 언급한 객체지향설계 방법론을 시스템 분석, 설계, 구축에 어떻게 이용하는지 확인하고 검토해 본다.

### 3.1. N社의 事例

N社는 현재 약 355명의 종업원이 근무하는 제조업체이며 여기에서 생산하는 주요제품은 A, B, C의 3가지이다. 이러한 제품의 조립은 제 1 공장 근처에 있는 창고에서 이루어진다.

N社의 자재명세서(BOM)은 다음 〈표 3-1〉과 같다. 이를 통해서 제품을 만드는데 필요한 부품과 원자재를 파악한다.

또한, 각 제품의 제조과정의 자료에 대해 간략히 살펴보면 다음의 〈표 3-2〉와 같다.

〈표 3-1〉 N社 자재명세서

| 제 품 A | 제 품 B | 제 품 C |
|---|---|---|
| .A | .B | .C |
| .D(4) | .F(2) | .G(2) |
| .I(3) | .G(3) | .I(2) |
| .E(1) | .I(2) | .H(1) |
| .F(4) | | |

〈표 3-2〉 N社 제품관련자료

| 작업장번호 | 작업가능시간 | 비용(시간당) |
|---|---|---|
| 작 업 장 1 | 6000시간 | $ 20 |
| 작 업 장 2 | 4500시간 | 25 |
| 작 업 장 3 | 2400시간 | 35 |
| 작 업 장 4 | 1200시간 | 65 |

| 항 목 | 작업장 번호 | 작업시간 (단위당시간) | 현보유 재고량 | 조달 기간(週) | 재고비용 | 취소비용 |
|---|---|---|---|---|---|---|
| 부품 A | 1/4 | .2/.1 | 100 | 1 | $ 2.0 | $ 20 |
| 부품 B | 2/4 | .3/.08 | 200 | 1 | 2.0 | 20 |
| 부품 C | 3/4 | .1/.05 | 175 | 1 | 2.0 | 20 |
| 부품 D | 1/4 | .15/.1 | 200 | 1 | 1.5 | 14 |
| 부품 E | 2/4 | .15/.05 | 195 | 1 | 1.5 | 14 |
| 부품 F | 2/3 | .15/.2 | 120 | 1 | 1.5 | 14 |
| 부품 G | 1/2 | .3/.1 | 200 | 1 | 1.5 | 14 |
| 부품 H | 1/3 | .05/.1 | 200 | 1 | 1.5 | 14 |
| 원재료 I | | | 300 | 1 | 1.0 | 8 |

위와 같은 자료를 바탕으로 다음과 같이 MRP 시스템을 構築한다.

### 3.2. MRP 시스템 分析 및 設計

위의 2절에서 언급한 바와 같이 MRP 시스템 분석과 설계는 다음과 같은 7단계에 걸쳐서 이루어진다.

1단계 : 시스템 要求 受容

목표 시스템의 요구사항을 분석하면 다음과 같다.

우선, 시스템 분석시 우선 고려하여야 할 사항들에 대해 살펴보면 다음과 같다.

○재계산방법의 결정(再計算法, 純變更法)

○MRP 시스템 계획 시간범위를 얼마로 하는가?

○계획기간의 길이는 어느 정도로 할 것인가?

○Lead time을 고정, 변동, 구성화일에 의한 조정 방법중 어느 것을 적용할 것인가?

○소요량 원천정보(pegging)는 단일 수준, 전 수준, 최종제품수준 중 어느 것을 선택할 것인가?

○복수 시간 소요량의 분리전개는 확정수주인가 또는 예측인가?

○Lot size의 방법과 보정기능은 어떻게 정할 것인가?

○Loss率에 의한 소요량 수정 기능은 계산과정에서 Loss率을 사용하여 산출할 것인가?

○설계변경의 실시기능은 확인되었는가?

○출고예정수의 기간별 관리는 되었는가?

○안전재고의 취급은 어떻게 하였는가?

○Order변경 메시지의 선별기능은 되었는가?

○工場月曆(shop calender)의 채용은 어떻게 되었는가?

○제조 level을 장래의 확장을 고려하여 몇 단계까지 허용할 수 있는가?

○시뮬레이션 기능을 활용하였는가?

N社의 MRP 시스템은 다음과 같은 절차를 따라 운용된다.

(1) 제품의 수요예측

제품의 수요예측은 ① 일정기간에 있었던 총제품의 주문량, ② 동일기간에 대한 판매예측, ③ 이 두 자료를 토대로 한 경영자의 판단에 의한 생산량 등에 의해 결정된다.

구성품의 총수요량은 최종제품의 순수요량이 결정된 다음 최종제품과 구성부품간의 관련성 즉, 제품구조에 따라 결정된다.

(2) 제품의 생산일정과 생산량파악

(3) 제품분석도의 작성

(4) 품목별 재고현황과 조달기간의 파악

(5) MRP 계획표의 작성과 운용

MRP 계획표의 작성 절차를 각각 구분하여 기술하면 다음과 같다.

① 총수요량의 결정

② 순수요량의 결정

순수요량=총소요량－{(보유재고량＋수입확정량)－안전재고량}

③ 현재고의 결정

기말재고=기초재고＋수입확정량－총소요량

④ 안전재고량의 산정

⑤ order量 산출 및 납기지정

(6) 주문서(order)발행

2단계 : 客體 및 서어비스의 定義

위의 MRP 시스템 특성으로부터 MRP 시스템의 객체와 서어비스는 1차적으로 다음과 같이 정의할 수 있다.

○고객(Customer): Place(주문의 전달), Amend(주문변경), Cancel(주문취소), Deliver (고객에게 배달)

○주문(Order): Place, Amend, Cancel, Deliver, Delay(배달지연), Allocate(주문에 해당하는 제품할당)

○제품(Product): Allocate, Deliver, Manufacture(필요한 제품제조)

○관리자(Manager): Contact, Manufacture

○창고(Warehouse): In(입고), Out(출고)

○부품(Part, component): Locate, Combine(조립)

○기계(Machine): Product, Time

이상과 같은 객체와 서어비스가 정의될 수 있으나 이 정의가 최상의 정의인가 여부를 알기 위해서는 계속적인 反復定義 및 修正이 필요하며 시스템 분석가의 경험과 나름대로의 체계가 필요하다.

3단계 : 客體間 相互關係 設定

이 단계는, 1단계 및 2단계에서 정의한 객체간의 상호관계를 정의하는 단계로 우선 제품명세서에서 얻을 수 있는 정보로 제품(Product)과 부품(Part/Component)은 "is_part_of", "1對多 혹은 1對 1"의 관계가 있음을 알 수 있다.

4단계 : 客體의 詳細한 內容 定義

이 단계에서는 7단계의 최종 설계가 끝나고 구현단계에 들어갔을 때 바로 이용할 수 있도록 객체의 내용을 보다 자세히 기술한다.

5단계 : 클래스라이브러리 檢討 및 下部시스템 構築

소프트웨어 시스템 구축의 효율을 높이기 위하여 기존에 미리 구축하여 두었던 클래스를 이용할 수 있는데, 기존에 구축된 클래스의 예는 다음의 〈표 3-3〉과 같다.

위에 언급된 클래스를 적절히 이용하면 시스템 소프트웨어 요소의 재활용성을 높일 뿐 아니라 시스템 수정 및 변경이 용이하여 시스템 분석가, 설계자, 개발자, 그리고 이용자에게 많은 도움을 줄 수 있다.

〈표 3-3〉 主要 클래스 라이브러리

| | |
|---|---|
| Object | |
| Bank Simulation | Input State |
| Behavior | Inspector |
| Benchmark | Message |
| Binary Choice | Method Description |
| Bit Blt | Music Formatter |
| Boolean | Parse Node |
| Browser | Pipe |
| Change | Point |
| Change Set | Pop Up Menu |
| Checker | Project Browser |
| Class Category Reader | Rectangle |
| Class Organizer | Remote String |
| Collection | Server |
| Compiled Method | Shared Queue |
| Compiler | Stream |
| Controller | String Holder |
| Control Manager | Text Style |
| Customer | Tile |
| Display | Tree |
| Disp Object | Tree Node |
| Explainer | Undefined Object |
| File Model | Variables |
| Financial History | View |
| Input Sensor | Windowing Transformation |

이 사례에서는 기존의 클래스에 MRP 클래스를 추가하고 여기에 2단계에 추출한 객체와 서어비스 기능을 명시하여 하나의 하부시스템으로 설계한다.

```
예) Basic  ┌─Location──┬─Warehouse
   Class   │           └─Dealer
           │
           │ Inventory──┬─Warehouse_Inv
           │            └─Dealer_Inv
           │
           │ Customer
           └─Sales
```

6단계 : 클래스 位階圖表 作成

전단계에서 이루어진 분석을 바탕으로 클래스 위계를 표시한다. 이때 Smalltalk에서는 하나의 root object만 있어야 하지만, C++로 구현하는 경우에는 여러 개의 root object가 있을 수 있다.

7단계 : 具現을 위한 最終 設計

마지막 최종설계 단계에 대한 설명은 다음의 3.3에서 한다.

이러한 7段階의 設計過程은 단순히 주어진 順序에 따라 一回的으로 처리되는 것이 아니라 反復的이고 循環的인 過程을 거쳐 設計가 完了된다.

### 3.3. MRP 시스템의 具現

위의 2절에서 완료한 시스템 분석 및 설계의 다음 단계는 시스템 구현단계이다. 이 단계에서 이루어지는 구현의 例 즉, 객체지향프로그램의 일부를 보이면 다음과 같다.

```
//file part.h

class Part
{
int part_id;

struct product_produce
    {
    int product_id;

    int part_ needed;

    product_produce*next;
    };
int part_stock;

struct part_supplier
    {
    char* company_name;

    char* company_address;

    part_supplier*next;
    };
public:

part(int part_id);

~part( );

void print_supplier( );

void print_product( );

void print_stock( );

int stock_part(int amount);
```

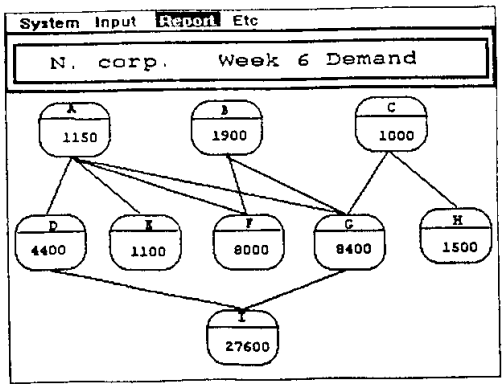
```
int use_part(int amount);
int update_product_data(int product_id, int part_needed);
int update_supplier_data(char* company_name, char*company_address);
}


//file product.h
class Product
{
int product_id;
int product_stock;
int manufacture_day;
struct part_needed
    {
    Part *part;
    int product_stock;
    int manufacture_day;
    struct part_needed;
        {
        int part_id;
        int part_amount;
        part_needed*next;
        };
public:
product(int product_id);
~product( )
void print_part_data( );
void print_stock( );
int stock_product(int amount);
int use_product(int amount);
int manufacture_product(int product_id);
}
```

이상과 같이 시스템이 구현될 수 있다.

위의 기본적인 내용을 바탕으로 N社는 다음과 같은 MRP 시스템 정보를 비롯하여 여러 가지 다양한 보고서를 얻을 수 있다.

MRP 시스템 구축시 추가로 고려하여야 할 사항은, 시스템 운영에 있어서 수요의 예측이




〈그림 3-1〉 N社 MRP시스템 報告書

상당히 중요한 비중을 차지하며 시스템에 입력되는 자료의 정확도가 시스템의 성패에 미치는 영향이 무척 큼에도 불구하고, 정확한 수요예측에 대한 기법이 전혀 제시되지 않고 있다는 점이다. 〔金炯郁, 1975〕 그러므로 Fuzzy이론, Neural Network 등에서 이용되는 추론 및 예측기법을 도입하여 보다 정확한 예측으로 효율적인 시스템 계획 및 운영을 할 수 있도록 하여야 한다.

## 4. 結 論

지금까지 본 연구에서는 정보시스템의 구축을 위한 새로운 설계방법을 제시하였다. 이 방법론은 다음과 같이 7단계로 나누어진다.

① 시스템 要求 收容

② 客體 및 서어비스의 定義

③ 客體間 相互關係 設定

④ 客體의 詳細한 內容 定義

⑤ 클래스라이브러리 檢討 및 下部시스템 構築

⑥ 클래스 位階圖表 作成

⑦ 시스템 具現을 위한 最終設計

이 方法論은 기존의 객체지향방법과는 달리 시스템 分析·設計 및 具現에 걸친 시스템 開發 壽命週期上의 전과정에 걸쳐서 활용이 가능하며 다음단계로의 이용이 자연스럽다는 특징이 있다.

1단계는 시스템 설계와 연계되는 단계이고 2, 3, 4, 5, 6단계는 본격적인 시스템 분석단계인

데 이 단계는 상호반복적으로 분석이 이루어진다. 그 다음의 7단계는 시스템 구축을 위해 분석과 구축이 서로 연결되는 단계이다. 또한, 각 개별단계가 하나의 패러다임내에서 추진되므로 壽命週期別로 다른 패러다임을 이용하는 여타의 방법론보다 효과적으로 시스템을 구축할 수 있다.

이러한 방법론을 이용하여 기존의 MRP 시스템을 개발할 경우 객체지향 접근법의 장점인 소프트웨어 시스템개발 소요시간·비용 절약, 소프트웨어 중복개발 방지 및 재사용, 보다 복잡하고 광범위한 컴퓨터환경에서 시스템구축 등의 효과가 있다.

이 연구는 차후에 다음과 같은 부분의 연구의 기초가 될 수 있다.

첫째, 구조적기법과 마찬가지로 명세 분석과 시스템 설계를 하기 위한 자동화도구(CASE tools) 제작의 기초가 될 수 있다.

둘째, 이러한 방법론을 이용하였을 때 어떠한 프로그래밍 언어로 개발하는 것이 효과적일 것인가에 대한 연구가 필요하다.

셋째, 시스템개발 수명주기상의 다른 부분, 예를들어 시스템 분석 및 구현 단계 뿐 아니라 시스템 테스트·유지보수 단계에도 적용할 수 있는 방법론을 제시할 수 있도록 방법론을 확강한다.

넷째, 객체지향접근법과 인공지능 분야의 연구를 결합하여 보다 정확한 자료를 산출하여 이를 바탕으로 시스템을 구축하였을때 보다 효과적인 시스템이 될 수 있어야 한다.

## 參 考 文 獻

〔1〕安重鎬 譯, 經營情報시스템디자인, 서울 : 법문사, 1989.

〔2〕安重鎬, 經營電算處理, 서울 : 법문사, 1990.

〔3〕金炯郁, "컴퓨터시대의 첨단경영기법 : MRP 시스템," 경영과 컴퓨터, 8월호, 1985.

〔4〕河秀澈, "C++客體指向 프로그래밍을 위한 圖形化 技法," 홍익대학교(박사학위논문), 1990.

〔5〕韓圭晶, "客體指向 프로그램의 품질평가기준과 테스팅屬性에 관한 연구," 중앙대학교(박사학위논문), 1991.

〔6〕Boehm, B.W., *Verifying and validating software requirements and design specifications*, IEEE Software, Jan., 1984, pp. 75-88.

〔7〕Booch, G., Object-oriented development, *IEEE Trans. on software engineering*, Vol.

SE-12, No.2, Feb., 1986.

[8] Coad, P., Object-oriented analysis, *American programmer*, special issue on object orientation, Vol.2, No.7-8, Summer, 1989.

[9] Henderson-Sellers, B. & Edwards J.M., Object-oriented systems life cycle, *Communications of ACM*, Vol. 33, No. 9, Sep., 1990, pp. 143-159.

[10] Korson, T. & McGregor, J.D., Understanding object-oriented: A unifying paradigm, *Communications of ACM*, Vol.33, No.9, Sep., 1990.

[11] Pressman, R.S., *Software engineering*, 2nd ed. McGraw-Hill, 1987.

[12] Seidewitz, F., General object-oriented software development: Background and experience, *21st Hawaii international conf. on systems sciences*, Jan., 1988, pp. 262-270.

[13] Shlaer, S. & Mellor, S., *Object-oriented systems analysis: modeling the world in data*, Yourdon press computing series, Englewood Cliffs, N.J.: Prentice-hall, 1988.

[14] Sommerville, I., *Software engineering*, 3rd. ed., Addison-Wesley, 1989.

[15] Wasserman, A.I., Pircher, P.A. & Muller, R.J., An object-oriented structured design method for code generation, *ACM SIGSOFT software engineering notes*, Vol.14, No.1, Jan., 1989.

[16] Winblad, A.L., Edwards, S.D. & King, D.R., *Object-oriented software*, Addison-Wesley, 1990.

[17] Wirfs-Brock, R., Wilkerson, B. & Wiener, L., *Designing object-oriented software*, Prentice-Hall, 1990.