□ 기술해설 □

객체지향 소프트웨어 개발 방법론 동향

명지대학교 최성운* · 도흥석** · 계원경**

● 목 차 ●

1. 서 론

소프트웨어 개발은 크게 문제를 조직적으로 이해하는 과정 및 정리된 문제를 컴퓨터가 이할 수 있는 언어로 표현하는 과정으로 나뉘어질 수 있다. 소프트웨어 개발 방법론(Software Development Methodology)이란 이러한 과정에 지속적으로 적용될 수 있는 방법이나, 절차, 기술 등을 말한다. 또한 방법론은 소프트웨어 개발을 위해 수행되는 절차, 기법, 산출물 등의 구성요소들로 이루어진 하나의 시스템으로 정의되기도 한다.

이러한 소프트웨어 개발 방법론에 대해서는 많은 연구 개발이 이루어져 왔고, 실제 소프트웨어 개발에 널리 적용되어 왔다. 이중 객체지향 소프트웨어 개발 방법론(Object-Oriented Software Development Methodology)은 1990년대 들어서면서 소프트웨어 개발에 본격적으로 적용되기 시작하였고, 객체지향 방법론의 재사용성(Reusability), 이식성(Portability), 확장성(Extensibility) 등을 기반으로 많은 분야에서 적극적으로 활용되고 있다.

객체지향 방법론은 객체지향 패러다임(Paradigm : 세상을 바라보는 시각)을 기반으로 하고 있으며, 여러 가지 개발 방법(Development Method)들이 개발되어 있다. 이러한 객체지향 개발 방법들은 기본적으로는 모두 유사한 과정과 기법을 사용하고 있으나, 그 적용 분야나 강조되는 특성에 따라 개발 방법별로 차이가 있다. 본 논문에서는 현재 개발되어 적용 가능한 여러 가지 객체지향 방법들을 비교

*정 회 원
**학생회원

분석하며, 향후 객체지향 소프트웨어 개발 방법들의 방향에 대해 살펴보고자 한다.

2장에서는 객체지향 방법들의 종류 및 장단점에 대해, 3장에서는 비교에 대해, 4장에서는 여러 방법들의 통합 시도인 Unified Method (UM)에 대하여 간략히 기술한 후, 5장에서는 객체지향 방법론의 미래 동향을 기술한다.

2. 객체지향 개발 방법의 종류

객체지향 패러다임이란 "상호작용 하는 객체들의 집합"으로 세상을 바라보는 시각이다. 결국 객체지향 방법들에서는 목적 시스템을 객체와 그들의 상호작용으로 파악하고 표현하게 된다. 이러한 시각을 바탕으로 객체지향 방법들은 객체 및 그 구조를 정의하는 정적 관계(Static Relation) 부분과 객체들간의 상호작용을 정의하는 동적 행위(Dynamic Behavior) 부분으로 소프트웨어를 파악한다.

정적 관계는 객체 및 클래스들 사이에 관계가 문제 영역이 존재하는 시간 동안 시간의 흐름에 관계없이 항상 고유하게 유지되는 것이다. 예를 들어 포함 관계(Aggregation), 계승 관계(Inheritance) 등이 이에 속한다. 동적 행위는 정적 관계와 상대적인 개념으로 시간의 흐름에 따라 변하는 관계를 말한다. 동적 행위는 객체들의 상태 변화와 이런 상태 변화를 유발시키는 행위에 중점을 둔다. 동적 행위는 시간이 흐르면서 연속적으로 이루어지는데, 이러한 연속적인 흐름이 모여 어떤 하나의 작업을 구성한다.

객체지향 소프트웨어 개발은 이러한 정적 관계와 동적 행위를 분석, 설계, 구현 등 소프트웨어 개발(Development Process) 전 과정에 걸쳐 일관되게 적용한다. 객체지향 소프트웨어 개발 과정은 일반적으로 기존의 소프트웨어 개발 과정과 동일하나 일관된 객체지향 방법을 전 과정에 동일하게 적용한다는 면에서 기존의 방법론과 다르다. 즉 분석 단계에서는 객체를 추출하고 그들 간의 관계를 파악하며, 객체간의 행위를 파악하게 된다. 설계 단계는 시스템을 환경을 고려하여 분석 과정을 그대로 반복하게 되며, 실행 단계에서는 컴퓨터 언어를 고려하여 설계 과정을 반복하게 된다.

객체지향 방법들은 위와 같은 공통적인 특성을 가지고 있으나, 각각의 적용 범위와 특화 분야가 방법들에 따라 다양하다. 다음은 현재 연구 개발되어진 대표적인 방법들에 대한 개요와 장단점에 대한 설명이다.

2.1 Rumbaugh의 방법(OMT : Object Modeling Technique)

Rumbaugh 그의 저서에서[1] 소프트웨어 개발에 있어서 기존의 방법론들과 객체지향 방법론의 근본적인 차이는 객체지향 방법론이 기능적인 분리를 기본으로 하지 않고, 역할을 수행하는 실세계 객체들의 묘사를 중심으로 한다는 것이라고 하였다. 다시 말해서 객체가 어떻게 이용되는가 보다는 객체가 무엇인가를 명시하는 것에 초점을 둔다는 것이다.

Rumbaugh의 OMT를 이용한 객체지향 개발 방법은 구현 영역 개념(Implementation-Domain Concept)이 아닌 응용 영역 개념(Application-Domain Concept)에 대해서 이를 명세화하고 조직화하였다는 데서 그 특성을 찾아볼 수 있다. 또한 다양한 모형화 방법을 포함하고 있다는 특성을 갖는다. 특히 객체 모형의 경우는 다른 방법론에 비해 가장 풍부한 모형화 개념을 포함하고 있다.

OMT는 3가지의 모델링 기법을 적용하는데, 이것은 기존의 방법을 상당 부분 계승한 것으로 정보 모델과 구조화 분석, 설계와 유사성을 가지고 있다. OMT에서 주로 사용되는 기법은 다음과 같다.

- ▶ 시스템 내에서 객체의 정적인 구조와 그 관계를 묘사하는 객체 모형화(Object Modeling)
- ▶ 시스템의 제어적인 면을 묘사하는 동적 모형화(Dynamic Modeling)
- ▶ 시스템 내부의 기능을 묘사하는 기능적 모형화(Functional Modeling)

2.2 Jacobson의 방법(OOSE : Object-Oriented Software Engineering)

OOSE[2]는 오랜 시간 동안 사용되어 왔던 세 가지의 다른 기법들을 혼합한 방법이라 할

수 있다. 그 첫 번째 기법은 1960년도부터 개발되고 있으며 현재까지 많은 응용 영역에서 사용되고 있는 객체지향 프로그래밍이다. OOSE는 객체지향 프로그래밍의 캡슐화(Encapsulation)와 상속(Inheritance), 클래스와 인스탄스 간의 관계와 같은 주요 개념을 사용하였다. 그리고 두 번째로 모형화(Modeling)란 개념이다. OOSE에서는 시스템 개발을 연속적인 모델의 생성 및 분석으로 간주한다. 모델은 시스템의 이해를 도울 뿐 아니라 시스템의 구조를 체계적으로 정의할 수 있게 해준다. 마지막으로 블록 설계(Block Design)가 하드웨어의 설계로부터 도입되었는데 블록 설계는 소프트웨어의 변화와 유지를 잘 수행할 수 있게 한다.

OOSE는 '사용사례 접근 방법(Use Case Driven Approach)'이라고 하기도 한다. 이렇듯이 Jacobson의 OOSE에서는 사용사례(Use Case)를 가장 특징적인 개념으로 볼 수 있다. 사용사례 모형은 시스템 외부와 시스템이 어떻게 상호 작용하는지를 파악함으로써 시스템의 기능을 설명한다. 이러한 사용사례 모형에 기반한 OOSE의 개발 과정은 분석과 제작(Construction), 그리고 테스트로 나뉜다.

분석의 목표는 시스템의 기능적인 요구에 따른 시스템에 대한 이해이다. 객체를 찾고, 조직화하고 객체 상호작용을 설명한다. 객체들의 연산과 내부적인 관점은 분석 과정에서 이루어진다. 제작은 소스 코드의 설계와 구현을 모두 포함한다. 이 과정에서는 분석 과정의 객체가 역으로 추출된다는 것이 중요하다. 테스트에서는 시스템의 정확성을 테스트한다.

OOSE의 기법에는 요구 모형(Requirement Models), 분석 모형(Analysis Models), 설계 모형(Design Models), 구현 모형(Implementation Models), 테스트 모형(Test Models)이 있다. 요구 모형은 다시 사용사례 모형(Use Case Model)과 문제 영역 객체 모형(Problem Domain Object Model), 인터페이스 묘사(Interface Description)로 나누어진다. 이 방법론의 특징은 객체를 인터페이스 객체, 엔터티 객체, 제어 객체로 구분하므로 써 시스템의 변화에 유연하게 대처할 수 있게 했다는 것이다.

2.3 Booch의 방법(OOADA : Object-Oriented Analysis and Design with Applications)

OOADA[3]는 시스템을 논리적인 구조와 물리적 구조로 나누어 각각에 대해 정적 모델과 동적 모델로서 정의한다. 논리적 구조는 시스템의 구현과는 직접적인 관계가 없는 시스템의 주요 추상체의 식별과 의미를 기술하는 것을 말하며, 물리적 구조는 구현을 위한 소프트웨어와 하드웨어 구성품을 기술하는데 이용되는 모듈 구조나 프로세스 구조 등을 말한다.

이 개발 방법은 Waterfall 모델처럼 순차적이지 않고, 논리적인 구조와 물리적인 구조의 구별을 통해 점진적이고 반복적으로 파악하는 "Round-Tip Gestalt Design"을 적용한다. OOADA는 크게 마이크로 프로세스와 매크로 프로세스 두 부분으로 나뉘어 진다.

마이크로 프로세스 부분은 다음 네 가지 단계로 이루어져 있다.

- ▶ 추상화 단계별로 클래스와 객체를 식별한다.
- ▶ 객체와 클래스의 의미를 식별한다.
- ▶ 클래스와 객체들간의 관계를 식별한다.
- ▶ 클래스와 객체들을 구현한다.

매크로 프로세스는 마이크로 프로세스를 제어하기 위해 사용된다. 이 단계는 다섯 가지 활동으로 설명된다.

- ▶ 개념화(Conceptualization) : 요구의 핵심을 정립
- ▶ 분석(Analysis) : 요구되는 행위 모델을 생성
- ▶ 설계(Design) : 시스템 구조가 생성
- ▶ 전개(Evolution) : 구현
- ▶ 유지(Maintenance) : 유지 보수 관리

OOADA의 표기법에는 각각 객체 다이어그램(Object Diagram), 클래스 다이어그램(Class Diagram), 상태 전이도(State Transition Diagram), 상호작용도(Interaction Diagram), 모듈 다이어그램(Module Diagram), 처리도(Process Diagram) 등이 있다.

Booch는 OOADA를 제안할 때 Ada를 이용한 소프트웨어 개발에 초점을 두었으므로 객체와 클래스, 객체간의 상호 작용에 대한 풍부한

개념을 지원하고 있다. 그러므로 이 방법은 실시간 시스템이나 병행성을 지닌 시스템을 모형화 하기에 가장 바람직하다. 또한 전체 시스템 가시화에 유용하고 설계를 위한 문서화 기법을 강조하였다.

2.4 Wirfs-Brock의 방법(DOOS : Designing Object-Oriented Software method)

DOOS[4]에서는 추상화(Abstraction)를 객체지향 방법의 본질로서 파악한다. 즉 추상화를 통해 실세계의 복잡성이 체계화, 단순화되는 것이다.

실세계의 개체(Entity)들은 객체(Object)들로 추상화 되거나 캡슐화 된다. DOOS는 객체를 찾는 것으로부터 시작하여, 그 객체들의 각각의 책임성(Responsibility)을 파악하여 기술한다.

DOOS는 두 가지 부분으로 나누어지는데, 탐사 단계(Exploratory Phase)와 분석 단계이다. 탐사 단계에서는 객체들과 실세계의 역할을 수행하는 객체들의 책임성(Responsibility)과 협력성(Collaboration)을 정의한다. 또한 분석 단계에서는 앞 단계를 재 정의하고 능률적으로 정리함으로써 보다 자세하게 분석한다.

책임성(Responsibility)이란 하나의 객체가 가지고 있는 지식과 객체가 수행할 수 있는 행위라 할 수 있다. 협력성이란 각 객체들의 책임성들을 수행하기 위해 요구되는 다른 객체들과의 협력관계이다. DOOS에서 객체들간의 협력성은 클라이언트/ 서버 방식으로 나타난다.

Wirfs-Brock의 DOOS는 처음으로 객체의 책임성이라는 개념을 도입하였다. DOOS는 설계 방법론이기 때문에 캡슐화의 특성을 포함하여 모형화 하도록 하고 있다.

그러나 이러한 특성이 분석 단계부터 고려될 필요는 없다. 이 방법은 가장 자세한 설계 단계를 정의하고 있으며, 각 설계 단계마다의 검증을 위한 워크쓰루(Walkthrough) 방법도 제공한다. 설계에 있어 동적인 특성은 전혀 고려하지 않으며 설계 지원 방법임에도 불구하고 모듈 인터페이스 개념을 지원하지는 않고 있다.

2.5 The Object-Oriented Analysis and Design(OOA/OOD) method by Coad & Yourdon

이 방법[5, 6]은 정보 모델링, 객체지향 프로그래밍(Object-Oriented Programming Language), 지식 기반 시스템(Knowledge Based System)을 기반으로 한 객체지향 분석의 하나이다.

이 방법론의 OOA 부분은 다섯 가지 주요 활동으로 이루어진다.

- ▶ 클래스와 객체 추출(Finding Class & Object)
- ▶ 구조의 식별(Identifying Structures)
- ▶ 서브젝트 식별(Identifying Subjects)
- ▶ 속성 정의(Defining Attributes)
- ▶ 서비스 정의(Defining Services)

이러한 주요 활동들에 따르면, 분석 과정에서 생성된 OOA 모델은 다섯 가지 층을 가진다.

- ▶ 서브젝트 층(Subject Layer) : 서브젝트는 한 번에 모델의 어느 정도를 제어할 수 있는가에 대한 수단.
- ▶ 클래스와 객체 층 : 문제 영역을 묘사하는 적절한 클래스를 찾기 위한 층
- ▶ 구조 층 : 분류 구조는 일반화와 특수화를 반영한 클래스 조직을 묘사한다. 어셈블리 구조는 전체와 컴포넌트 파트(Component Part)를 반영한 집단화를 보여준다.
- ▶ 속성 층 : 속성 다이어그램과 인스탄스 다이어그램이 나온다.
- ▶ 서비스 층 : 서비스는 메시지 수령 시에 행해지는 처리.

분석(OOA)의 다섯 층은 설계(OOD)의 4개 컴포넌트에 적용된다.

- ▶ 인간 상호작용 컴포넌트(Human Interaction Component) : 사용자 인터페이스의 설계를 위한 객체 추가 작업을 실시
- ▶ 문제 영역 컴포넌트(Problem Domain Component) : 분석의 결과에 새로운 객체를 추가하는 등의 세부 작업이 이루어짐
- ▶ 작업 관리 컴포넌트(Task Management

Component) : 시스템에서의 태스크를 확인하기 위한 단계

▶ 자료관리 컴포넌트(Data Management Component) : 데이터의 저장과 추출에 관한 데이터 측면이 관련

Coad/Yourdon 방법의 특징은 체계적인 방법이 아닌 대화적으로 개발을 진행해 나가는데 있다. 그러나 이 방법은 객체 간의 통신 지원 개념이 취약하고 대규모 프로젝트 보다는 작은 시스템 개발에 적합하다는 단점이 있다.

2.6 Martin/Odell의 방법(OOAD : Object-Oriented Analysis and Design)

Martin과 Odell의 방법론 OOAD[7]는 복잡한 세계를 추상화(Abstraction), 일반화(Generalization), 합성화(Composition)에 의해 분석하고 관리하는 방법이다. 분석 시 종종 정적인 특징, 동적인 특징으로 나누어지기도 하지만 이들의 방법은 이들 두 가지 특성을 하나로 묶으려고 시도하였다. 이들은 객체 중심의 분석이 행동(Behavioral)에 관한 것, 즉 동적인 것에 기초를 두어야 된다고 말하고 있고 정적인 부분은 그에 대한 동적인 부분으로부터 끌어내 설명되어져야 된다고 말하고 있다. 이들의 방법은 영역 내의 목표를 구별하는 것에서부터 시작하여서 객체(Object)와 사건(Event)을 정의하고 또 정의되어진 새로운 사건(Event)을 가진 모든 사건(Event)이 정의되어질 때까지 순환하는 방식으로 진행하게 된다.

2.7 Shlaer와 Mellor의 Object Lifecycles (OL)

Shlaer/Mellor의 객체지향 방법[8, 9]은 시스템, 도메인, 서브시스템, 객체, 상태, 프로세스의 6가지 계층으로 하나의 대형 시스템을 최상위 시스템 수준으로부터 최하위 수준의 시스템 수준까지 하향식 방식으로 표현하고 있다. 이들의 주요 기술은 정보 모형화(Information Modeling)라고 한다. 그리고 이 기술은 지식을 형식화(Formalization) 하는데 사용된다.

OL은 시스템의 개발을 객체 중심 분석(OOA) 그리고 객체 중심 설계(OOD)로 나눈다. 첫 번째 객체 중심의 분석은 문제의 영역 안에서 객체와 속성에 따라 개념적인 실체를 추상화하는 정보 모형화(Information Modeling) 과정, 객체와 객체들 사이에서 시간이 지나감에 따른 관계를 공식화하는 상태 모형화(State Modeling) 과정, 그리고 각 상태(State)에서 필요로 되어지는 행위(Action)를 포함하는 처리 모형화(Process Modeling) 과정으로 이루어져 있다. 그리고 두 번째, 객체 중심의 설계는 메인 프로그램과 초기화, 상태 검사 등을 하는 4개의 클래스 그리고 몇 개의 응용(Application) 클래스들로 구성되어진 각각의 프로그램에 의하여 표현되어진다.

2.8 그 밖의 방법론

앞에서 기술된 방법론 외에도 많은 방법론이 있다. Rubin/Goldberg 방법의 경우 다른 방법론들에 비해 문제를 파악하는 기법을 제공하고 있어서 객체를 식별하는데 편리하다. 그러나 객체의 속성을 파악하는 과정이 명확하지 않다는 단점을 갖고 있으며, 반면에 기존의 Harel 차트를 동적 모형화에 사용하므로 써 동적 모형의 정형화가 가장 뛰어난 방법이다. 이 방법론은 시나리오 기반의 모형화 방법으로 분류할 수 있으며, 프로토타이핑 기술에 가장 적합하다. 또한 다양한 기술서를 제공하고 있기는 하지만 이에 상응하는 도형화 도구가 부족하다.

현재 연구가 진행되고 있는 서비스 중심적 소프트웨어 개발 방법론의 경우 분석, 설계, 구현, 테스트의 단계로 이루어지는 기존의 방법론에 비해 문제를 보다 체계적으로 이해하고 파악하는 부분 즉, 요구 분석 단계를 추가하여 객체 식별을 정형화 시켰다고 볼 수 있다.

3. 객체지향 개발 방법들의 비교

방법론의 개별적인 장단점은 이미 2장의 개발 방법론들의 개요 설명에 기술되어 있다. 이러한 방법들의 객관적이고 일반적인 비교는 기준을 설정하기 어려울 뿐 아니라 비교 자체도 별 의미를 가지지 못한다. 이 장에서 사용된 용어 및 개념 등에 대한 개괄적인 비교를 기술한다.

3.1 사용된 용어의 비교

Booch	Rumb-augh	Martin/ Odell	Shlaer/ Mellor	Coad/ Yordon	Wirfs-Brock	Jacobson
객체	객체	객체	객체	객체	객체	객체
클래스	객체 클래스	객체 타입	객체	클래스	클래스	클래스
지료	속성	지료	속성	속성	속성	속성
메시지	사건	시간	사건	메시지	자극	메시지
오퍼레이션	오퍼레이션	오퍼레이션	오퍼레이션	서비스	오퍼레이션	메소드

3.2 객체 사이의 관계

Booch	Rumbaugh	Shlaer/ Mellor	Coad/ Yourdon	Wirfs-Brock	Jacobson
링크	객한 연관관계	수퍼/서브 타입 관계	전체-부분구조 (Whole-Part)	협력관계 (Collaborations)	연관관계 acquaintance
포함관계	포함관계	이전/삼전 (binary/terna-ry)관계	인스턴스연결 메시지연결		consist-of communication

3.3 클래스 간의 비교

Booch	Rumbaugh	Shlaer/Mellor	Coad/ Yourdon	Wirfs-Brock	Jacobson
연관관계				is-kind-of	
상속	일반화 관계	클라이언트	일반화/특수화	is-analogous-to	상속연관
포함관계		/서비 프렌트	구조	is-part-of	확장연관
인스턴스관계			(Gen-Spec)	has-knowledge-of	인스턴스연관
메타근레스	메타클래스			depends-upon	

4. The Unified Method(UM)

2장에서 소개한 바와 같이 1980년 후반부터 객체지향 방법들은 짧은 시간에 매우 다양하게 연구 개발되어 분석과 설계에 적용되어지고 시험되어져 왔다. 하지만 이런 많은 방법들이 있다는 것은 반대로 여러 가지 불일치 및 호환성의 문제를 야기시킬 수가 있을 가능성 또한 내포하고 있음을 말해준다.

이런 여러 방법들의 Artifacts를 단일화 시켜 방법들이 가지고 있는 불필요한 것들을 제거하고 앞으로 사용할 사용자들의 혼동을 최소한으로 줄이며, 최대로 안정하고 효과적인 방법론을 제시하자는 의도에서 Rumbaugh, Booch, Jacobson이 경영하는 Rational Software사가 제안된 표준 안이 바로 Unified Method (UM) [10]이다. UM(이 이름은 작업 단계의 일시적인 표기 명으로써 앞으로의 버전에 새로운 이름으로 바뀔 것임)은 객체지향 시스템 개발 시 필요한 Artifacts를 정의, 표기, 설명하기 위해 연구 개발되고 있다. 이 결과물은 현재 미완성으로서 Rational Software사에서는 계속해서 여러 방법론 연구 개발자들로부터 자문을 구하고 수정 보완함으로써 1996년 말경 완성된 버전을 만들어 내려 하고 있다.

UM은 Booch, Objectory, OMT에 기반을 두고 있으며, 또한 다른 방법들까지도 포함하는 단일 모델링 방법이다. UM에서는 형식 메타 모델(A Formal Metamodel), 시각적 표기법(A Graphical Notation), 사용례(A Set of Idioms of Usage) 등을 정의하고 있다. 메타 모델은 모델의 모델로서 명확하고, 표준화된 문법 및 의미를 제공한다. 이러한 메타 모델을 정의함으로써 기존의 방법론들과의 호환성을 유지할 수 있으며, 공통의 언어로서의 역할을 할 수도 있다. 시각적 표기법은 여러 다이어그램 및 시각적 표기법에 대해 정의하고 있다. UM에서 현재 제공하는 시각적 표기는 Use-Case, Class, State, Message-Trace, Object-Message, Module, Platform 다이어그램 및 Operation Specification 등이다. 또한 사용례에서는 개발자나 도구에서 쓰일 수 있는 여러 사용례를 포함한다.

UM에서 개발 공정은 기본적으로 Architecture-Driven, Incremental, Iterative Development Process를 지원하고 있다. 하지만 특정 개발 공정을 정의하지는 않는다. 개발 공정은 문제 영역 및 분야의 특성에 따라 많은 차이가 있을 수 있으므로 개발자의 특성에 맞도록 특화 시킬 수 있도록 하기 위해서 이다.

결국 UM은 객체지향 소프트웨어 개발에 필요한 Artifacts에 대한 표준화를 시도하고 있는 것이다. 이러한 측면에서 볼 때 UM은 객체지향 연구 개발에 있어 상당히 긍정적으로 평가되며, Rational Software사의 상업적 성공에 힘입어 앞으로의 표준화로서 정착되리라 예상된다.

5. 객체지향 방법론의 미래 동향

우리는 위에서 몇 개의 방법론들을 살펴보았지만 이외에도 다른 많은 방법이 존재한다. 객체지향 패러다임에 근거한 방법들은 각각의 적용 분야 및 특징들을 가지고 광범위하게 연구

개발되고 있다. 또한 시스템 개발자들은 많은 관심을 가지고 객체지향 방법을 적용하기 위해 노력하고 있다. 하지만 이러한 노력과 관심에 비해 실제 적용 사례는 그리 많지 않은 상황이다. 현재로서는 방법론 자체가 실무에 적용되어지는데 있어서 경험이나 지침이 부족한 상태이며, 통일된 표기법이 없으며, 안정성 또한 확실히 증명되지 못하고 있기 때문이다.

그럼에도 불구하고 객체지향 방법론이 미래의 소프트웨어 개발 패러다임으로서 제시되고 있는 이유는 객체지향 방법론이 기존의 방법론이 해결하지 못하는 근본적인 문제, 즉 일관된 추적성(Seamless Traceability)이나 재사용성(Reusability) 등을 해결하고 있기 때문이다. 상업적 성공이 기술적 우월성을 나타낸다고는 볼 수는 없지만, 객체지향 방법론은 멀티미디어, 사용자 인터페이스나 Client-Server 시스템을 기반으로 상업적인 분야에서 미래의 소프트웨어 개발 방법론으로서 적극적으로 활용되기 시작하고 있다. 또한 객체지향 패러다임은 단순히 소프트웨어 개발 모델로서 뿐 아니라, 세상을 바라보고 분석하는데 적용할 수 있는 사고의 방법으로서, 제시된 여러 인간사고 모델[11, 12, 13]들과 그 맥락을 같이하고 있다. 이러한 일반성(Generality) 및 자연성(Naturalness)으로 인해 객체지향 패러다임은 여러 분야에서의 이론적 연구도 활발히 진행되고 있다.

결론적으로 컴퓨터적 사고 중심의 구조적 방법론이 이미 한계로 지적되고 있는 상황에 와 있음을 볼 때 수년 후 객체지향 중심의 방법론이 시스템 구축을 비롯한 문제해결에 있어 완전히 기존의 방법론을 대치할 것으로 보여진다. 객체지향 방법론은 현재 및 미래의 기술로서 연구 개발되고 있다.

참고문헌

[1] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen , Object-Oriented Modeling and Design, Prentice-Hall, Englewood Cliffs, New Jersey, 1991.

[2] Ivar Jacobson, Object-Oriented Software Engineering A Use Case Driven Approach, The Addison-Wesley Publishing Company Inc., 1992.

[3] Grady Booch, Object-Oriented Analysis And Design with Applications, The Benjamin / Cummings Publishing Company Inc., 1994.

[4] R. Wirfs-Brock, B. Wilkerson, and L.wiener, Designing Object-Oriented Software, Prentice-Hall, Englewood Cliffs, New Jersey, 1990.

[5] P. Coad and E. Yourdon , OOA -- Object - Oriented Analysis, 2nd Edition, Prentice-Hall, Englewood Cliffs, New Jersey, 1990.

[6] P. Coad and E. Yourdon , OOD -- Object - Oriented Design, Prentice-Hall, Englewood Cliffs, New Jersey, 1991.

[7] J.Martin and J.J. Odell, Object-Oriented Analysis and Design, Prentice-Hall, Englewood Cliffs, New Jersey, 1992.

[8] S. Shlaer and S.J. Mellor, Object-Oriented Systems Analysis : Modeling the World. In Data, Yourdon Press : Prentice-Hall, Englewood Cliffs, New Jersey, 1988.

[9] S. Shlaer and S.J. Mellor, Object-Oriented Systems Analysis : Modeling the World. In States. Yourdon Press : Prentice Hall, Englewood Cliffs, New Jersey, 1992.

[10] Grady Booch and J. Rumbaugh, Unified Method for Object-Oriented Development : version 0.8, Rational Software Co., 1995.

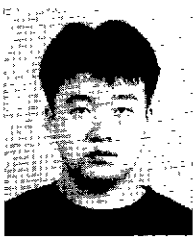
[11] Johnson-Laird P.N. Mental models. In Foundations of Cognitive Science(Posner M.I., ed.), pp.469-93, Cambridge, MA : MIT Press, 1989.

[12] Carroll J.M. and Olson J.M. Mental models in human-computer interaction. In Handbook of Human-Computer Interaction (HELANDER M.,ed.), pp.45-65, Amsterdam : North-Holland, 1988.

[13] Rogers Y., Rutherford A. and Bibby P., eds. Models in the Mind : Theory, Perspective and Application, London : Academic Press, 1992.

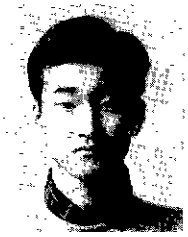
최 성 운

1985 한국외국어대학교 무역학과 학사 졸업
1987 Oregon State University 전산학과 석사 졸업
1992 Oregon State University 전산학과 박사 졸업
1993~현재 명지대학교 컴퓨터공학과 조교수
관심분야 : 객체지향 소프트웨어 공학, 인간 컴퓨터 인터페이스



도 흥 석

1995 명지대학교 컴퓨터공학과 학사 졸업
1995~현재 명지대학교 대학원 컴퓨터공학과 석사 과정
관심분야 : Object-Oriented Methodology, Case Tool



계 원 경

1995 명지대학교 컴퓨터공학과 학사 졸업
1995~현재 명지대학교 대학원 컴퓨터공학과 석사 과정
관심분야 : 객체지향 소프트웨어 개발 방법론, Case Tool

● Call for Papers ●

- 행 사 명 : 10th International Workshop on Testing of Communicating Systems
- 대회일자 : 1997년 9월 1~3일
- 개최장소 : 서울
- 논문마감 : 1997년 3월 10일
- 주 최 : 한국통신
- 연락 및 문의처 : 한국통신 김명철 · 강성원

㉾137-792 서울시 서초구 우면동 17 한국통신 연구개발본부
Tel : 02-526-5180 Fax : 02-526-5567
E-mail : {mckim, kangsw}@sava.kotel.co.kr