# 객체지향 분석 및 설계를 지원하는 모델러의 구조

*김영진 · *김대헌 · *유철중 · *장옥배 · **정효택 · **양영종 · **이삼덕
*전북대학교 컴퓨터과학과 소프트웨어공학연구실
**한국전자통신연구원 컴퓨터소프트웨어기술연구소

# An Architecture of Modeler Supporting Object-Oriented Analysis and Design

*Y. J. Kim, *D. H. Kim, *C. J. Yoo, *O. B. Chang
**H. T. Jung, **Y. J. Yang, **S. D. Lee
*SE Lab., Dept. of Computer Science, Chonbuk National University
**CSTL, ETRI

## 요 약

기존의 개발 방법이 절차지향 방법에서 객체지향 방법으로 변화해감에 따라 구현시의 중요성보다는 분석 및 설계 단계의 중요성이 높아지고 있고 이를 지원하는 여러 가지 모델링 도구들이 개발되었다. 그러나 이러한 도구는 사용자를 위한 기능에 더 많은 비중을 두어 도구 자체를 구조적으로 정확한 객체지향 방법을 적용하지 못하였다. 본 연구는 인터넷/인트라넷 환경에서 프레임워크를 기반으로 소프트웨어를 개발하기 위한 도구 중에서 모델러에 관한 모듈을 개발하는 것으로서, 도구 자체를 MVC 기반의 객체지향 개념을 적용하여 개발하고 있고, 플랫폼에 독립적인 Java 언어를 이용하여 개발하고 있기 때문에 이와 유사한 OMT 에디터(Java version)를 분석하여 문제점을 개선함으로써 UML 표기법을 사용할 수 있는 모델러를 설계 및 구현하였다. 본 논문은 이러한 모델러를 개발하기 이전의 도구의 구조에 관한 기초 연구로서 위임형 이벤트 모델을 사용한 컨트롤러의 독립성을 이용한 이벤트 처리 기법을 적용하였으며, 여러 개의 뷰(폼)사이의 메시지 전달을 위하여 Agent 패턴이라는 자체 설계 패턴을 개발함으로써 도구 자체를 객체지향적으로 구조화하였다. 이러한 객체지향적 설계 및 구현은 사용자의 요구가 변경되고 도구 자체의 기능확장이 요구될 경우에 빠르고 쉽게 이를 반영할 수 있다는 장점을 가지고 있다.

## 1. 개 요

지난 수년간 소프트웨어 공학은 매우 빠르게 변하고 있는 하드웨어의 발전과 사용자의 요구를 충분히 만족하기 위해 다양한 프로그래밍 기법과 방법론이 많은 발전을 하였다. 그중에서도 특히, 생산성과 재사용성을 위한 많은 프로그래밍 기법이 소개되었고 소프트웨어를 빠르게 개발할 수 있는 RAD 도구(Rapid Application Development Tools)도 소개되고 있다. 프로그래밍 기법으로는 프로시저가 주체가 되어 객체인 데이터를 포함함으로써 프로시저를 실행하는 피동적인 프로그래밍 방법에서 객체가 주체가 되어 객체와 객체 사이의 메시지를 전달하는 방법으로 하는 능동적인 프로그래밍 방법으로 변화되었다[1]. 사용자 중심으로 개발된 소프트웨어로서 각종 언어를 지원하는 비주얼 도구를 살펴보면, 이는 구현단계에서 사용자가 빠르고 쉽게 프로그래밍을 할 수 있도록 하였다. 이러한 변화는 복잡한 시스템의 구현과 이해를 용이하게 할 뿐만 아니라 기존의 개발자 중심에서 사용자 중심으로 개발 방법이 변화되고 있음을 나타내고 있다[2].

초기에 객체지향을 이용한 프로그래밍 기법에 있어서 적합한 분석 및 설계 방법이 없었기 때문에 개발자는 여전히 기존의 방법으로 분석 및 설계를 하였다. 이에 대한 해결책으로 여러 가지 객체지향 방법에 적합한 분석 및 설계 방법이 연구되었으나 이를 적용시키기에는 많은 어려움이 있었다[3]. 현재는 사용자가 객체지향 방법으로 분석 및 설계를 쉽게 하기 위한 모델링 지원도구로서 Rational Rose와 OMT 에디터(Java version) 등이 있으므로 이를 이용하여 편리한 개발을 할 수 있다[4].

그러나, 이러한 도구들은 너무 많은 기능과 도구 자체가 구현 단계에서 완전한 객체지향 개념을 사용하지 못하여 도구의 확장 및 유지보수를 어렵게 하고 있다. 이러한 문제점을 해결하기 위해서는 사용자의 요구사항을 정확하게 수용할 수 있는 객체지향 모델링 도구로서 요구사항이 변경될 때 이를 설계 단계에서 쉽게 변경할 수 있도록 도구 자체를 구조적으로 체계화해야 하며 정확한 적용방법이 필요하다.

따라서 본 논문에서는 사용자의 요구사항을 정확하게 수용할 수 있는 객체지향 모델링 도구로서 Model-View-Controller(이하

본 논문은 '99년도 한국전자통신연구원 컴퓨터소프트웨어기술연구소의 위탁연구과제(과제명: 프레임워크의 비주얼 인터페이스 기법 및 브라우저 구현) 수행결과물의 일부입니다

MVC)구조 기반의 OMT 에디터 구조를 개선한 모델링 도구 측면의 구체적인 객체지향 적용 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장의 관련연구에서는 OMT 에디터와 MVC 구조를 분석하고, 3장에서는 MVC 기반의 모델러 구조를 이벤트 처리 방식의 개선과 Agent 패턴이라는 자체 설계 패턴을 사용한 적용 기법을 제안한다. 4장에서는 결론 및 향후 연구과제를 논한다.

## 2. 관련연구

본 연구과제는 Java 언어를 이용하여 객체지향 분석 및 설계를 지원하는 UML 기반의 모델러를 개발하기 때문에 이에 관한 관련연구로서 기존의 관련된 동일한 개발 언어를 사용하면서 가장 효과적으로 객체지향적인 설계를 포함하고 있는 OMT 에디터를 분석하고 그 기반이 되는 MVC 구조에 관하여 분석하였다.

### 2.1 OMT 에디터

OMT 에디터는 객체지향 데이터 모델링을 분석시에 데이터의 변환 관계를 나타내고 있는 프로세스, 프로세스에서 동작하는 데이터, 그리고 시간에 따라 변환하는 데이터의 상태를 제어하는 기능들로 구분하는 J.Rumbaugh의 OMT 방법론을 기초로 하고 있다. 이들은 각각 기능 모델(Functional Model), 객체 모델(Object Model), 동적 모델(Dynamic Model)로 설계된다[5]. 그러나 프로그래밍 구현시에 객체는 정적 특성과 동적 특성으로 구분되며, 정적인 특성은 데이터로 표현되고, 동적인 특성은 함수 또는 메소드로 표현되기 때문에 OMT에디터의 비주얼 인터페이스 기법에는 기능 모델링이 별도로 구현되지 않았다.

OMT 에디터의 객체 모델과 동적 모델 에디터의 구조는 MVC 패턴을 사용하여 정확하게 설명할 수 있다.

2.1.1 MVC 구조

Smalltalk-80에서 제안된 MVC 구조는 객체지향 구조에서 많이 연구 및 응용되고 있는 구조로서 입력, 처리, 출력을 Model, View, Controller의 세가지 구성요소로 분리하여 유지 보수와 확장 가능성을 용이하게 한다[6]. MVC의 본래 목적은 사용자에게 디스플레이 되어지는 부분과 사용자가 그것을 제어하는 부분으로부터 애플리케이션을 분리하여 보다 간단한 사용자 인터페이스를 개발하는 것이다. 이러한 MVC 구조에서 각 객체의 역할은 다음과 같다.

- ◆ 모델(Model) - 프로그램의 핵심 기능과 데이터를 가지고 있으며, 공통적인 정보를 관리한다
- ◆ 뷰(View) - 처리된 결과를 사용자에게 디스플레이 한다. 뷰의 디스플레이를 위한 결과는 모델로부터 변경 통보를 받았을 경우 이를 반영해야 하므로 모델이 어떤 변화가 일어나고 있는지를 관찰자 입장에서 알아야 한다.
- ◆ 컨트롤러(Controller) - 이벤트가 발생하였을 경우 데이터 처리를 위해서 모델에게 변경된 사실을 통보하며, 처리된 결과는 뷰에 반영된다. 그러나 모든 사용자의 외부 입력이 모델을 경유하는 것이 아니라 모델의 처리가 필요없이 바로 뷰에 디스플레이를 요청하여 직접적인 상호작용을 하는 경우도 있다.

MVC 구조를 사용하는 OMT의 객체 모델과 동적 모델 에디터는 데이터(모델부분)가 인터페이스로부터 분리되어 있고 제어부분(컨트롤러)이 뷰로부터 분리되어 있기 때문에 OMT 에디터를 이용하여 사용자가 세부적인 정보를 일 필요없이 도구에서 제공하는 데이터 구조를 포함하는 클래스 구조를 만들 수 있도록 한다. 이 외에도 OMT에디터는 여러 가지 패턴을 사용하고 있다. OMT 에디터에서 사용된 패턴의 활용에 대하여 간략하게 살펴보면 다음과 같다.

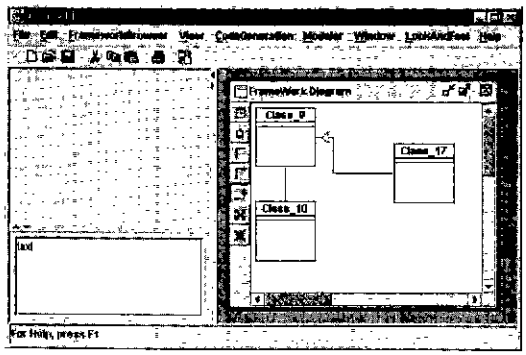
- ◆ Observer 패턴 - 윈도에서 보여주고자 하는 모델에서 발생하는 변화의 트랙을 유지
- ◆ Visitor 패턴 - 생성되어진 모델을 저장
- ◆ Chain of Responsibility - 모델링 영역에서 발생되는 이벤트에 대한 메시지의 처리
- ◆ Iterator 패턴 - 연관된 객체들에 대한 탐색 및 접근
- ◆ Bridge 패턴 - 추상화 클래스를 구현하여 동일한 파일 다이얼로그를 윈도 시스템 또는 유닉스 시스템에서 기능은 같고 모양은 플랫폼에 맞게 생성

지금까지 살펴본 OMT 에디터는 비록 여러 가지 패턴을 사용하여 객체지향적으로 잘 설계되어져 있긴 하지만, 구조적인 측면에서 초기 개발 당시에 JDK1.0.2 버전으로 개발되었기 때문에 상속형 이벤트 모델을 사용하고 있어서 MVC 구조의 뷰와 컨트롤러 부분을 완전히 분리하지 못하였다는 문제점이 있다.

## 3. MVC 기반의 모델러 구조

본 논문은 인터넷/인트라넷 환경에서 프레임워크 기반의 소프트웨어를 분석 및 설계하기 위한 모델링 도구(이하 JFree)의 모델러라고 하는 모듈을 개발하는 과정에서 도구 자체를 보다 완전한 객체지향적으로 설계함으로써 확장성과 재사용성을 높이고자 하는데 주안점을 두었다. JFree의 구조는 OMT 에디터를 기초로 개발되고 있기 때문에 MVC 구조를 기본 구조로 채택하였으며, JDK1.2 버전으로 개발하고 컨트롤러의 독립성을 보장하기 위하여 위임형 이벤트 모델을 사용하고 서로 다른 뷰 사이의 메시지 전달을 위하여 Agent 패턴이라는 자체 설계 패턴을 사용함으로써 구조를 명확히 분리하였다.

[그림 1]은 이러한 구조를 사용한 JFree의 실행화면을 나타낸다.

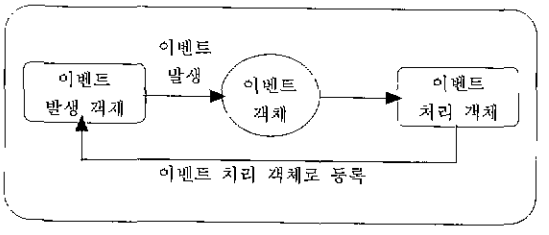


[그림 1] 모델러 화면 구성도

### 3.1 위임형 이벤트 모델을 적용한 컨트롤러의 독립성

기존의 상속형 이벤트 모델(Inheritance Event Model)은 JDK1.0의 이벤트 모델로서 Component 클래스로부터 상속받은 이벤트 처리 메소드에 의존하는 방식으로 이벤트를 발생한 객체에서 실행 이벤트가 처리되지 않으면 상위 클래스로 이벤트가 전파된다. 이러한 이벤트 모델은 뷰의 컴포넌트 객체가 구현시에 이벤트 객체를 포함하여야 하므로 이벤트 처리 부분을 독립시킬 수 없게되어 재사용성과 유지보수를 어렵게 만든다.

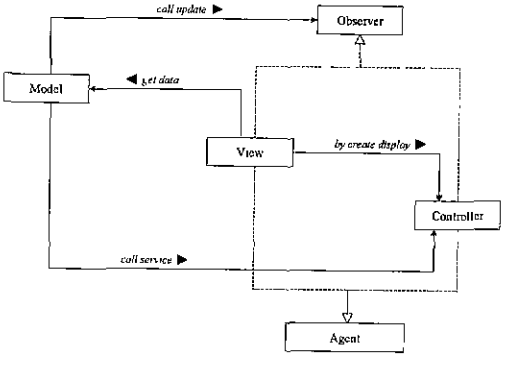
본 연구에서 적용한 위임형 이벤트 모델(Delegation Event Model)은 [그림 2]와 같이 이벤트 객체를 이벤트 발생 객체가 생성하여 이를 이벤트 처리 객체(리스너)에게 위임하는 방식의 이벤트 모델이다. 위임형 이벤트 모델을 사용할 경우에 이벤트 객체의 인스턴스를 이용하여 기능을 추가하는 클래스를 작성함으로써 클래스의 독립적인 기능확장과 재사용성, 그리고 유지보수를 용이하게 할 수 있다.




[그림 2] 위임형 이벤트 모델

### 3.2 Agent 패턴을 사용한 뷰 사이의 메시지 전달

MVC 구조의 각각의 요소(모델, 뷰, 컨트롤러)에서 자료구조는 Observer 패턴을 사용할 수 있지만, 여러개의 뷰가 존재할 경우 뷰 사이의 메시지를 전달할 수 있는 구조는 현재 명확히 구분이 되고 있지 않은 실정이다. 따라서 본 연구에서는 자체 설계 패턴인 Agent 패턴을 사용하여 여러 개의 뷰 사이에 발생하는 메시지를 전달함으로써 도구 전체가 일관된 개발을 할 수 있도록 하였다.




[그림 3] Agent 패턴을 적용한 MVC 구조

Agent 패턴은 [그림 3]과 같이 다수의 뷰와 컨트롤러 사이의 직접적인 통신을 할 경우에 컨트롤러로 하여금 서로 다른 뷰에게 메시지를 전달하는 과정을 명확히 구조화한 패턴이다.

Agent 패턴의 핵심 모듈은 AgentObservableT라는 인터페이스로 되어 있으며 [그림 4]와 같이 두 개의 메소드만을 정의하고 이를 구현하여 기능확장을 할 수 있도록 하였다

```
package jfree.data;

public interface AgentObservableT {
    public void set(Object source, String command, Object
                    arg);
    public Object get(Object source, String command, Object
                    arg);
```

[그림 4] AgentObservableT 인터페이스

본 연구에서 설계한 MVC 기반의 프레임워크 다이어그램 모델링을 지원하는 JFree는 사용자의 편리한 개발을 위한 비주얼 모델링 도구로서 여러개의 뷰(폼)를 사용하고 있다. 각각의 뷰들은 메시지를 전달함으로써 각각의 상태를 다른 뷰에게 전달한다.

## 4. 결론 및 향후 연구과제

본 논문에서는 도구 자체를 보다 완전한 객체지향 형태로 개발하기 위하여 플랫폼에 독립적인 Java 언어를 사용하고 있는 OMT 에디터의 구조를 분석하고, 이를 보완하여 MVC 구조 기반의 인터넷/인트라넷 개발환경하에서 소프트웨어를 개발하기 위한 분석 및 설계를 시각적으로 지원할 수 있는 모델러의 구조를 설계 및 구현하였다. 현재 연구과제로서 수행되고 있는 모델러의 구조는 앞으로의 객체지향형 도구를 개발할 때 재사용 가능하며 사용자의 요구사항이 변경될 때 도구 자체를 다양하고 쉽게 변경할 수 있을 것으로 기대된다.

향후 연구과제로는 본 논문에서 제시하고 있는 패턴 이외에도 다른 패턴을 이용하여 도구의 구조를 체계화하며 아직 개발중인 부분에 대하여 지속적인 보완을 할 예정이다.

## 참고 문헌

[1] David M. Arnow and Gerald Weiss, *Instruction to Programming Using Java™*, Addison-Wesley, 1998.

[2] Sherman R. Alpert, Kyle Brown and Bobby Woolf, *The Design Patterns Smalltalk Companion*, Addison-Wesley, 1997.

[3] Grady Booch, James Rumbaugh and Ivar Jacobson, *The Unified Modeling Language User Guide*, Addison-Wesley, 1998.

[4] Terry Quatrani, *Visual Modeling with Rational Rose and UML*, Addison-Wesley, 1998.

[5] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, *William Lorensen, Object-Oriented Modeling and Design*, Prentice-Hall International, 1991.

[6] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad and Michael Stal, *A System of Patterns*, John Wiley & Sons, 1996