객체지향 분석 지원을 위한 모델링 기법 정의 및 툴에 관한 연구

김 행 곤†

요 약

기존 객체지향 모델링 및 설계 기법들을 다양한 관점에서 비교해 본 결과, 객체지향 분석과 객체지향 설계에 대한 경계와 모델링 개념 표현에 대한 표기법 이해에 어려움이 있음을 알 수 있었다. 본 논문에서는 객체지향 분석의 중요성을 인식하여 필수적인 객체지향 분석 특징과 이해하기 쉬운 표기법의 객체지향 분석을 지원하는 분석 지향 모델링(AOM:Analysis Oriented Modeling)을 정의한다. AOM은 시스템 관점과 클래스 관점을 가지며 outside-in 방식으로 문제 영역에 접근한다. 다음으로, 다이어그램 편집기, 다이어그램 분석기, 코드 생성기, 로더, 저장소, 저장기로 구성된 AOM 지원 CASE 툴을 설계한다. 문제 영역 분석 후, 기존 CASE 툴에 예제 모델을 실행시키고 C++ 템플릿을 생성한다.

A Study on the Definition and Tool of Modeling Technique for Supporting Object-Oriented Analysis

Haeng-Kon Kim†

ABSTRACT

Many object-oriented modeling and design techniques are a great forward step for software development. We have found difficulties to understand in the boundary between object-oriented analysis(OOA) and object-oriented design(OOD) and notations for representing the modeling concepts after comparing with a number of distinct views. In this paper, we focus and recognize the importance of OOA as a result. We define AOM(Analysis Oriented Modeling) to support OOA with essential analysis features and understandable notations. AOM approaches problem domain in outside in way that has system view and class view. Next, we design AOM supporting CASE tool that consists of diagram editor, diagram analyzer, code generator, loader, repository and saver. After analyzing problem domain, we execute example models with existing CASE tool and generate C++ template.

1. 서 론

기존 절차적 개발 방법론들은 사용자의 요구사항 변경에 능동적으로 대처할 수 없으며 분석에서 설계로의 변환이 자연스럽지 못하여 이 과정에서 정보 손실의 우려가 있으며 소프트웨어 재사용과 유지보수에 대한 지원이 부족하고 분석과 설계에 있어 경계가 불분명하여 일관되지 못하기 때문에 대규모의 복잡한 소프트웨어 개발시 많은 문제점이 제기되고 있다. 양질의 소프트웨어를 대량으로 개발하고 재사용과 유지보수를 지원할 수 있는 새로운 소프트웨어 개발 방법론이 요구됨에 따

† 종신회원 : 대구효성가톨릭대학교 컴퓨터공학과 부교수
논문접수 : 1997년 2월 31일, 심사완료 : 1997년 10월 14일

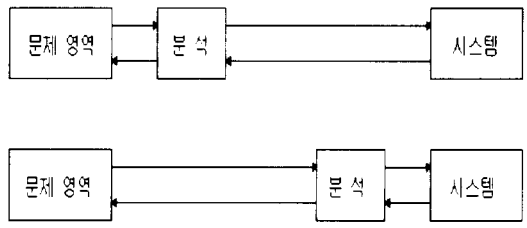
라 객체지향 방법론(object-oriented methodology)이 제안되었다[1]. 소프트웨어의 품질 향상과 생명주기의 전 과정에 걸쳐 소요되는 비용과 노력의 절감을 위해서 체계적인 기초 조사와 사용자의 요구사항을 명확히 하고 비용과 일정 등에 관한 치밀한 계획을 수립하며 요구되어지는 기능을 효과적으로 구현할 수 있는 시스템 분석이 선행되어야 한다. 따라서, 본 논문에서는 분석 지원을 위해 필요한 필수적인 객체지향 개념을 정의하여 이를 이해하기 쉬운 표기법으로 모델링하는 기법을 제안하고 이를 지원하는 툴을 설계한다. 전체 시스템을 시스템 관점과 클래스 관점으로 나누어, 시스템 관점에서는 분석의 첫 단계로 문제 기술서를 작성하여 시스템의 요구사항을 정의한 후 시스템 모델, 시나리오, 시스템 상태 모델, 시스템 메세지 모델을 작성하여 문제 영역에 대한 충분한 분석을 거친 후 시스템 관점에서의 분석 결과를 시스템 명세서로 작성한다. 클래스 관점에서는 시스템을 구성하는 각 클래스를 식별하여 그 관계를 정의함으로 시스템의 정적이고 구조적으로 표현하는 클래스 모델을 작성하고 구현될 시스템 내의 각 클래스와 클래스간의 메세지의 흐름을 클래스 메세지 모델로 나타낸다. 또한, 작성된 모델을 CASE 툴에서 실행 시켜보고 시스템 모델에 대한 템플릿으로 C++ 원시 코드를 작성한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로 객체지향 분석 방법론에 대해 전반적으로 살펴보고 3장에서는 제안된 분석 지향 모델링 AOM(Analysis Oriented Modeling)을 정의한 다음, 지원 툴을 설계하고 AOM에 의한 호 교환 시스템(Call Switching System)을 모델링한다. 4장에서는 실행 예 및 평가를 거친 후 마지막으로 5장에서 결론 및 향후 연구 과제로 맺는다.

2. 관련 연구

2.1 객체지향 분석 방법론

문제 영역에 대한 이해와 요구사항에 대한 이해를 사용자가 확인하기 위해서는 반복적으로 수행되어야 하며 개발 환경에 맞게 사용자 요구사항의 변경을 유도할 수도 있다. 개발자는 변화에 융통성 있게 정책과 표기법을 구성하여야 함과 동시에 안정된 상태를 유지할 수 있도록 유도하여야 한다. 분석을 크게 목표지향 분석과 문제지향 분석으로 분류하여 도식화하면 (그림 2-1)과 같다.



(그림 2-1) 문제지향 분석 / 목표지향 분석
(Fig. 2-1) Problem oriented analysis / Target oriented analysis

분석은 먼저, 사용자의 요구에 의한 개념을 바탕으로 문제 해결을 위해 사용자 관점에서 실세계 객체와 행위를 분석한 후, 모델링 기법을 이용하여 정형적 방법으로 요구사항을 추출한다[3]. 그러나, 문제 영역과 컴퓨터 영역의 객체들은 서로 상이한 목적을 가지며 추상화의 정도가 다름에도 불구하고 동일한 객체지향 개념과 표기법을 사용하고 있다.

분석 과정이 중요성을 더해가는 이유 중 하나는 분석 과정에서 오류를 발견하고 수정하는 것이 시스템이 완성되었을 때 수정하는 것보다 경제적이기 때문이다. 분석 과정에서 오류가 발생하면 설계 및 구현 과정까지 파급되어 오류 검출이 어렵고 비용이 많이 들기 때문에 분석 과정에 보다 많은 고려가 요구된다.

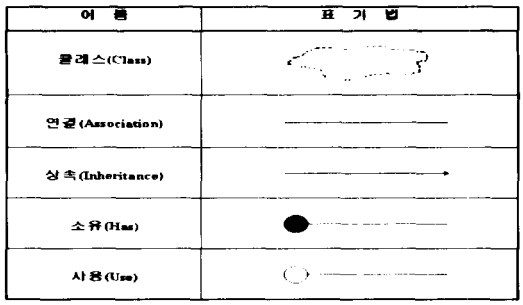
2.2 기존의 객체지향 분석 방법론

2.2.1 OOAD(Object Oriented Analysis and Design)

Booch가 제안한 객체지향 분석 및 설계 기법[4]은 전체 시스템의 가시화와 실시간 처리에 유용하며 설계를 위한 문서화 기법을 강조하고 있다. 그러나 설계 중심 방법론으로 분석 과정이 취약하고 모델의 구현에 종속적인 개념들을 많이 포함하고 있다. 또한 설계 과정이 너무나 경험에 의존적이며 반복적이므로 규모가 큰 프로젝트 수행시 과정이 복잡하고 구현 언어에 제한되고 객체, 클래스, 메세지, 속성 등의 정의 방법이 취약하다. (그림 2-2)은 OOAD에서 제공되는 표기법이다.

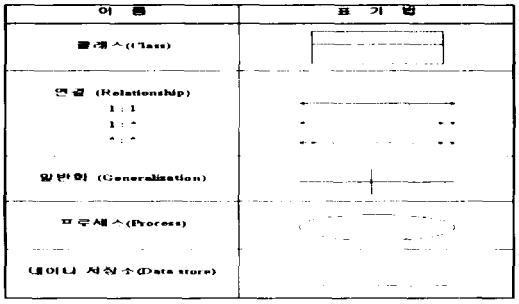
2.2.2 OMT(Object Modeling Technique)

객체 모델링 기법[5]은 Rumbaugh가 제안한 방법론으로 모델의 일관성 결여와 모델을 구축하는 전이 과



(그림 2-2) OOAD 표기법
(Fig. 2-2) Notations of OOAD

정이 자연스럽지 못하다. 객체 모델과 동적 모델이 객체 중심의 분석인 반면, 기능 모델은 시스템 전체를 기능적으로 다시 분할하는 과정이다(그림 2-3).



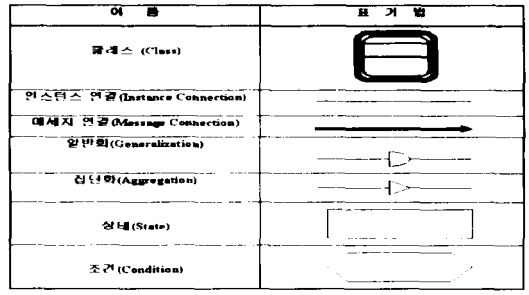
(그림 2-3) OLC 표기법
(Fig 2-3) Notations of OLC

2.2.3 OLC(Object Life Cycle)

정보 모델링을 기본으로 분석에 중점을 둔 Shlaer/Mellor의 방법론[6]은 모델링만을 지원하는 방법이다. 모델 작성 중 일관성이 결여되거나 새로운 사항이 발견되면 앞 단계의 모형에 수정이 이루어지는 반복, 순환적 과정을 가정한다. OLC는 대형 프로젝트 개발시에 시스템 분해성이 우수하며 실시간 처리와 시뮬레이션에 적합하다. 그러나 객체들의 반복 처리가 부족하고 설계 및 구현의 세부사항이 제시되지 않으며 표기법이 혼란스럽다(그림 2-3).

2.2.4 OOA(Object Oriented Analysis)

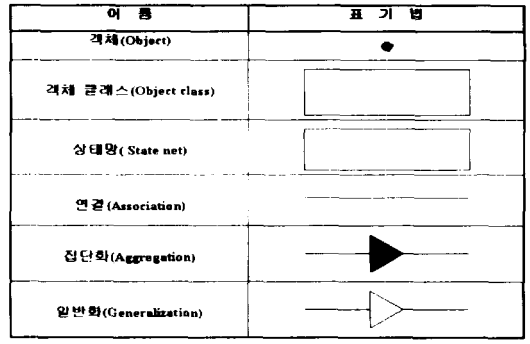
정보 모델링, 지식 공학, 객체지향 프로그래밍 개념 중 가장 우수한 개념을 혼합하여 제안된 Coad/Yourdon의 객체지향 분석 방법론[7]은 정보 모델링에서 속성, 인스턴스간의 관계성, 일반화/상세화 개념을, 지식 공학과 객체지향 프로그래밍 언어에서 속성의 캡슐화, 독자적 서비스 제공, 메세지를 통한 객체 상호간의 통신, 상속 개념을 도입하였다. 지원하는 개념이나 다이어그램이 적은 반면, 하나의 다이어그램으로 문제를 쉽게 파악할 수 있는 장점이 있다. 그러나, 객체간의 통신 지원 개념이 취약하고 규모가 작은 개발에 더 적절하다(그림 2-4).



(그림 2-4) OOA 표기법
(Fig.2-4) Notations of OOA

2.2.5 OSA(Object-oriented System Analysis)

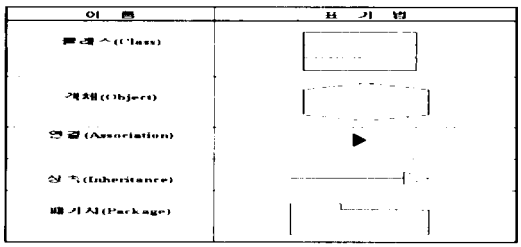
객체지향 시스템 분석[8]은 Embley가 제안한 것으로 분석 특징 지원이 우수하나 표기법에 있어 다른 방법론들에서 공통적으로 사용된 기호가 다른 의미로 사용됨으로 혼란을 가져오고 전반적으로 복잡하여 지원된 분석적 특징을 효과적으로 나타낼 수 없으며 전체 모델을 이해하기 어렵다(그림 2-5).



(그림 2-5) OSA 표기법
(Fig.2-5) Notations of OSA

2.2.6 Unified Method

Unified는 OMT, OOAD, OOSE(Object Oriented Software Engineering)를 중심으로 다른 여러 방법론의 장점을 통합한 방법론이다. 개발 중인 객체지향 시스템의 결과들을 명시하고 가시화하며 문서화함으로 사용자의 경험을 바탕으로 객체지향 분석과 설계의 정의 영역에서 표준화를 시도하였다[9,10].



(그림 2-6) Unified 표기법
(Fig. 2-6) Notations of Unified

3. 분석 지향 모델링 (AOM : Analysis Oriented Modeling)

3.1 개요

본 논문에서 제안하고 있는 분석 지향 모델링 AOM(Analysis Oriented Modeling)은 문제 영역을 객체지향 개념으로 분석하여 그 결과를 다양한 관점에 따라 이해하기 쉬운 표기법을 사용하여 모델링함으로써 효과적인 프레임워크를 제공할 수 있다. AOM은 (그림 3-1)과 같이 정제된 8 단계의 개발 과정을 점진적이고 반복적으로 수행한다. 분석 과정에서는 문제 영역을 다양한 관점으로 시스템을 분석하는 것이 이상적이다. 따라서, AOM은 다양한 분석의 관점을 지원하기 위하여 크게 시스템 관점과 클래스 관점으로 나누어 분석을 시도한다. 시스템 관점에서는 주어진 문제 영역에서 구현될 시스템과 상호작용하는 시스템을 정의한다. 시스템간의 경계를 명확하게 상호작용하는 인터페이스를 식별하고 식별된 시스템에 대하여 상태 변화와 메시지의 흐름을 나타냄으로 주변 환경을 고려한 시스템에 대한 정보를 얻을 수 있다. 클래스 관점은 구현하고자 하는 시스템을 중심으로 구성하고 있는 내부 클래스들을 식별함으로 그들간의 관계를 정의하고 클래스간의 메시지 흐름을 나타냄으로 보다 상세하게 분석할 수 있다[12].

(그림 3-1) 개발 단계
(Fig. 3-1) Development steps

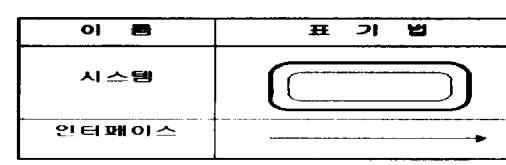
단계 1 : 문제 기술서

문제 기술서는 시스템 개발 과정에서 최초로 명세화되는 문서로 전 과정에서 기초적이고 중요한 자료로 사용된다. 실세계 시스템을 이해 가능한 형태로 모델하기 위해 먼저 사용자의 요구사항을 조사하고 분석하여 논리적으로 기술하는데 있어서 실세계의 중요한 특징을 추상화하여야 하고 세부적인 사항은 고려되지 않는다. 또, 설계 방법에 대한 제한이나 구현에 대한 결정 사항 없이 구현되어야 하는 대상을 설명되며 분석의 결과는 설계에 대한 준비 과정으로 이용될 수 있다.

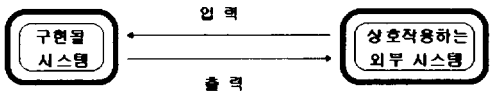
AOM의 문제 기술서는 분석의 초기 단계로 설계 및 구현의 결정 사항이 배제되고 분석 특징만을 포함한다.

단계 2 : 시스템 모델

시스템 모델은 문제 기술서의 정보를 시각적으로 간단 명료하게 표현하기 위한 것으로 하드웨어적 요소와 소프트웨어적 요소를 구분하고 시스템의 경계를 분명히 하며 시스템과 상호작용하는 외부와의 인터페이스를 정의한다. 따라서, 시스템 모델은 사용자와 개발자간의 이해와 대화를 용이하게 하고 시스템의 상세한 내부 사항은 블랙 박스화된 상태에서 외부와의 입출력만을 나타낸다. (그림 3-2)와 (그림 3-3)은 각각 시스템 모델



(그림 3-2) 시스템 모델에 대한 표기법
(Fig. 3-2) Notations of system model



(그림 3-3) 시스템 모델
(Fig. 3-3) System model

을 구성하는 표기법과 시스템 모델의 예를 나타내고 있다.

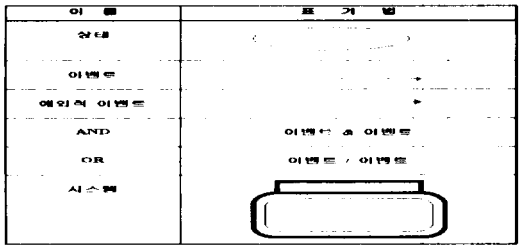
단계 3 : 시나리오

시나리오는 문제 기술서로부터 획득된 정보를 텍스트 형식으로 순차적으로 기술한 것으로 이벤트에 따른 시스템 내 상태의 변화를 나타내며 시스템 상태 모델을 위한 기본 자료로 이용된다.

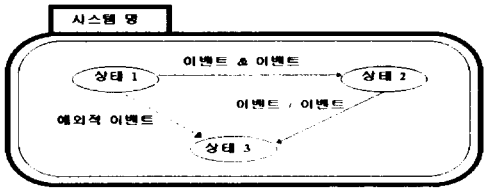
단계 4 : 시스템 상태 모델

독립적으로 각 시스템 내부에서 발생할 수 있는 이벤트에 따른 상태 변화를 모델링하는 것이다. 대규모의 복잡한 시스템일 경우, 시스템 이름을 쉽게 식별할 수 있도록 시스템의 좌측 상단에 시스템 이름을 표기한다.

시스템 상태 모델에서는 하나의 상태에서 다른 상태로 전이하기 위해서는 반드시 하나 이상의 이벤트가 발생하여야 한다. 이 때, 두 가지 이상의 이벤트가 발생



(그림 3-4) 시스템 상태 모델에 대한 표기법
(Fig. 3-4) Notations of system state model

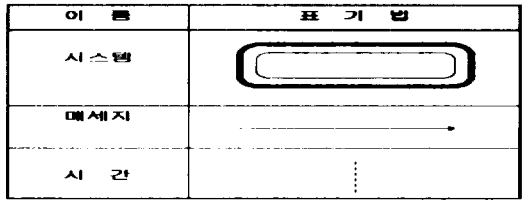


(그림 3-5) 시스템 상태 모델
(Fig. 3-5) System state model

하여야 전이가 일어나는 AND 조건일 경우, 기호 '&'로 나타내고 여러 이벤트 중 선택적으로 하나만 발생하여도 상태가 전이되는 OR 조건일 경우에는 기호 '/'로 나타낼 수 있다. (그림 3-4)와 (그림 3-5)는 시스템 상태 모델에 대한 표기법과 시스템 상태 모델이다.

단계 5 : 시스템 메시지 모델

시스템 메세지 모델은 시스템의 상호작용을 메세지의 흐름에 따라 표현된 모델로서 시스템 모델에서 식별된 시스템과 상호 작용하는 외부 시스템을 이중 사각형으로 나타내고 이를 중심으로 각각 시간을 의미하는 세로의 점선과 메시지의 흐름을 의미하는 화살표로 표현한다. (그림 3-6)과 (그림 3-7)은 각각 시스템 메시지 모델에 대한 표기법과 시스템 메시지 모델을 나타내고 있다.



(그림 3-6) 시스템 메시지 모델에 대한 표기법
(Fig. 3-6) Notations of system message model

(그림 3-7) 시스템 메시지 모델
(Fig. 3-7) System message model

단계 6 : 시스템 명세서

시스템 명세서는 시스템 모델, 시나리오, 시스템 상태 모델, 시스템 메시지 모델 등에서 알려진 정보를 추출하여 정보를 문서화하는 단계이다. (그림 3-8)은 AOM의 시스템 명세서를 나타내고 있다.

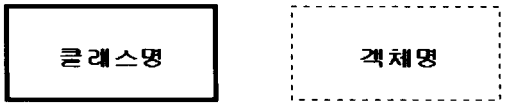
항 목	내 용
1. 이 름	
2. 상호작용하는 외부 시스템	
3. 입 력	
4. 출 력	
5. 설 명	

(그림 3-8) 시스템 명세서
(Fig. 3-8) System specification

단계 7 : 클래스 모델

클래스 모델은 시스템을 정적이고 구조적인 관점으로 표현하기 위한 것으로 구현될 시스템을 구성하고 있는 각 클래스를 식별한다. 식별된 클래스들간의 유기적인 관계를 나타내기 위해 AOM에서는 다중성을 가진 연결, 일반화/상세화, 집단화를 지원한다.

AOM에서는 클래스만을 식별하여 전체 모델에서 알기 쉽고 일관되게 나타내고 있다. AOM의 클래스에 대한 표기법으로는 가장 일반적으로 통용되고 있는 실선의 사각형을 채택하였으며, 사각형 내부에 식별된 클래스의 이름을 기입하고 클래스의 인스턴스인 객체는 점선의 사각형으로 나타낸다(그림 3-9).

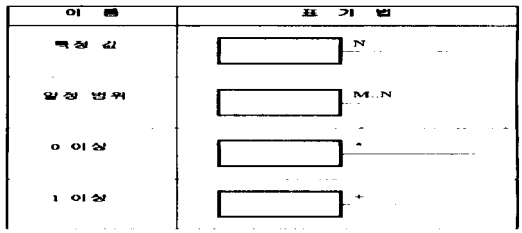


(그림 3-9) 클래스와 객체에 대한 표기법
(Fig. 3-9) Notation of class and object

식별된 클래스들 사이의 유기적인 관계를 연결, 일반화/상세화, 집단화로 구분하여 나타낸다.

객체지향에 어느 정도 경험이 있는 사람이라 해도 개념적으로 일반화/상세화와 집단화는 이해가 어려우며 혼란을 일으키기 쉽다. 따라서, 사용자와의 대화에 있어서도 사용자들이 쉽게 의미를 파악할 수 있는 표기법을 사용하는 것이 바람직하다[3].

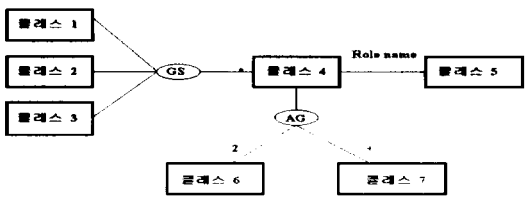
AOM에서는 두 관계성을 쉽게 식별할 수 있게 하기 위하여 일반화/상세화에 대한 표기법으로 Generalization/Specification의 약자 'GS'를 원 내부에 기입하고 집단화는 일반화/상세화와 동일하게 AGgregation의 약자인 'AG'를 원 내부에 기입함으로써 두 관계성 사이의 혼돈을 줄이고 사용자가 쉽게 이해할 수 있게 하였다. 연결을 비롯한 모든 관계성은 링크된 클래스에서 허용될 수 있는 객체의 수인 다중성을 이용하여 많은 의미를 나타낼 수 있다. OSA에서의 다중성은 사용자에게 지나치게 자유를 허용하므로 오히려 부담을 가지게 된다. 따라서, AOM에서는 사용자가 상황에 맞게 적절히 선택하여 사용할 수 있도록 (그림 3-10)과 같이 4가지로 분류하였다. 먼저, 다중성이 특정 값일 경우, 클래스와 관계성의 연결 부분에 특정값 N으로 나타내고 일정 범위일 경우에는 M ... N으로 나타낸다. 다중성이 0 이상의 값일 경우 기호 '*'로, 1 이상일 경우에는 기호 '+'로 표현한다. (그림 3-11)은 AOM의 관계성에 대한 AOM의 표기법을 나타내고 (그림 3-12)는 클래스, 관계성, 다중성을 이용한 클래스 모델의 예이다.



(그림 3-10) 다중성에 대한 표기법
(Fig. 3-10) Notation of multiplicity

이 름	표 기 법
연 결(Association)	Role name
일반화/상세화 (Generalization/ Specification)	GS
집단화 (Aggregation)	AG

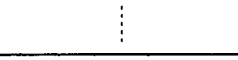
(그림 3-11) 관계성에 대한 표기법
(Fig. 3-11) Notation of relationship



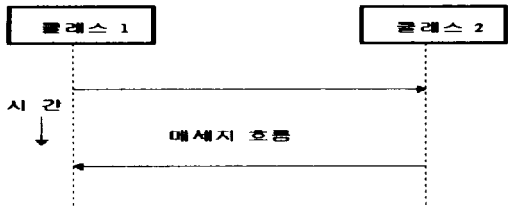
(그림 3-12) 클래스 모델
(Fig. 3-12) Class model

단계 8 : 클래스 메시지 모델

AOM의 마지막 단계로 클래스 메세지 모델은 구현될 시스템을 구성하는 여러 클래스들간에 일어날 수 있는 메시지의 흐름을 모델링한 것으로 그 형식은 시스템 메시지 모델과 유사하다. (그림 3-13)과 (그림 3-14)는 각각 클래스 메시지 모델에 대한 표기법과 클래스 메시지 모델이다.

이 름	표 기 법
클래스	
메세지	
시간	

(그림 3-13) 클래스 메시지 모델에 대한 표기법
(Fig. 3-13) Notation of class message model



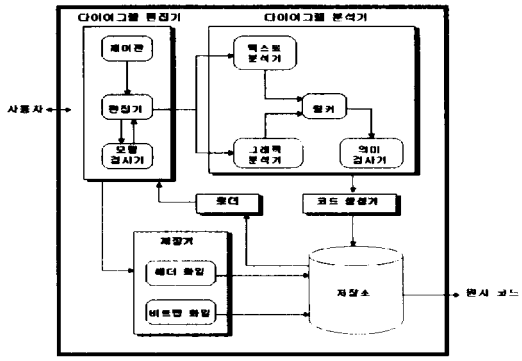
(그림 3-14) 클래스 메시지 모델
(Fig. 3-14) Class message model

3.2 AOM 지원 툴의 설계

본 논문에서 제안하는 AOM 지원 툴은 (그림 3-15)와 같이 소프트웨어 시스템의 개발시 분석 과정에서 이루어지는 모델링에 대하여 필요한 모델의 작성과 작성된 모델의 관리를 지원하도록 설계되었다. 각 모델별로 허용된 표기법만을 선택할 수 있도록 제공하므로 모델링 상에서 발생할 수 있는 오류를 최소화 시킬 수 있다. 또한, 도메인을 전체적으로 개략 분석에서 시작하여 점차적으로 각 부분의 상세 부분으로 접근함으로써 도메인 분석에 있어 핵심을 유지할 수 있다. 다이어그램 편집기와 다이어그램 분석기, 저장소와 저장기, 로더, 코드 생성기 등으로 구성된다.

(1) 다이어그램 편집기

모델을 작성하거나 작성된 다이어그램에 대하여 수



(그림 3-15) 구성도
(Fig. 3-15) Configuration

정, 삭제할 수 있으며 관련된 텍스트를 입력할 수 있다. 일련의 작업이 완료되면 표기법의 위치나 이름의 중복 등 필수 항목에 대하여 검사 결과, 오류가 발생하면 사용자에게 오류 메시지를 전달한다.

(2) 다이어그램 분석기

텍스트와 다이어그램의 각각 관련된 사항들에 대한 분석이 이루어진다. 클래스 명이나 두 클래스간 관계 중복, 연결되지 않은 클래스, 허용되지 않은 관계성 등을 중점적으로 분석한다. 분석이 끝나면 링커를 통해 관련된 텍스트와 다이어그램이 매핑되고 의미 검사기는 매핑 결과에 대하여 전체적인 의미를 검사한다.

(3) 저장소

저장소는 툴의 가장 중요한 부분으로 재사용 가능한 형태로 개발자들이 저장소의 정보를 새로운 정보를 만들 수 있도록 한다. 일관성과 무결성이 보장된 모델에 관한 정보와 모델을 구성하는 표기법에 관한 정보는 사용자가 편집기를 사용하여 모델링 시에 입력되는 것으로 모델에 관한 정보로는 모델의 종류와 모델의 실체를 나타내는 이름, 모델의 작성 유무 등의 정보를 저장하며 표기법에 관한 정보로는 표기법의 이름, 위치, 관련된 표기법과의 관계 등을 저장한다.

로더는 다이어그램에 대한 화일을 편집기로 불러오고 저장소는 다이어그램 이미지를 비트맵 화일로, 다이어그램 메타 정보를 헤더 화일로 저장한다. 코드 생성기는 다이어그램 분석기 결과를 해당하는 C++ 템플릿으로 생성한다.

3.3 AOM에 의한 모델링

분석 지향 모델링 AOM으로 "호 교환 시스템(CSS: Call Switching System)"을 사례 연구로 하여 각 단계별로 모델링하여 본다[12].

단계 1 : 문제 기술서

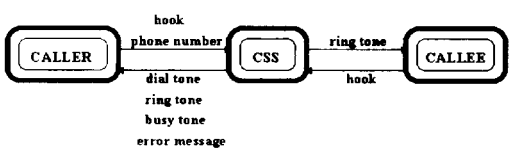
사용자와 개발자의 대화를 통해 추출된 사용자 요구사항을 문서화한 CSS의 문제 기술서는(그림 3-16)과 같다.

> 일정 지역내의 사용자에 대하여 한 사용자(CALLER)가 다른 사용자(CALLEE)에게 전화를 걸고자 한다. CALLER가 수화기를 들면(off hook) 호 교환 시스템(Call Switching System)은 CALLER에게 이미 알고 있는 CALLEE의 전화번호를 입력하여도 된다는 의미로 dial tone을 전송한다. 일정 시간내에 CALLER는 전화번호를 입력하여야 한다. 만약, 일정 시간동안 입력이 완료되지 않으면 CSS는 전화번호 입력에 대한 메시지를 CALLER에게 전송한다. CSS는 입력된 전화번호를 분석하여 정상적이면 CALLEE의 상태를 조사하고 오류가 발생하면 이에 대한 메시지를 전송한다. CALLEE의 상태가 idle이면 CSS는 CALLER와 CALLEE에게 ring tone을 전송하고 CALLEE의 상태가 busy이면 CALLER에게 busy tone을 전송한다. CALLEE가 수화기를 들면(off hook) CALLER와 CALLEE는 연결되며(connect) 통화가 이루어진다. 통화가 끝나면 CALLER, CALLEE가 모두 수화기를 놓고(on hook) 연결이 해제된다(disconnect).

(그림 3-16) CSS 문제 기술서
(Fig. 3-16) problem statement of CSS

단계 2 : 시스템 모델

문제 기술서에서 추출된 시스템 CSS와 상호작용하는 CALLER와 CALLEE를 중심으로 모델링한 CSS의 시스템 모델은 (그림 3-17)과 같다.



(그림 3-17) CSS 시스템 모델
(Fig. 3-17) System model of CSS

단계 3 : 시나리오

시스템 상태 모델을 작성하기 위하여 문제기술서로부터 하나의 시스템 내에서 일어날 수 있는 상태의 변화를 순차적으로 기술한다. (그림 3-18), (그림 3-19), (그림 3-20)은 각각 문제 기술서에서 추출된 CALLER, CALLEE, CSS의 시나리오이다.

1. 수화기를 든다.
2. 전화번호 입력 허용 신호인 dial tone을 받는다.
3. 이미 알고 있는 전화번호를 입력한다.
4. 일정 시간동안 입력을 완료하지 못하면 error message를 받게 된다.
5. 입력된 전화번호의 분석 결과에 따라 ring tone, busy tone, error message를 받는다.
6. 연결이 된다.
7. 통화가 끝나면 수화기를 놓는다.
8. 연결이 해제된다.

(그림 3-18) CALLER 시나리오
(Fig. 3-18) Scenario of CALLER

1. ring tone을 받는다.
2. 수화기를 든다.
3. 연결된다.
4. 통화가 끝나면 수화기를 놓는다.
5. 연결이 해제된다.

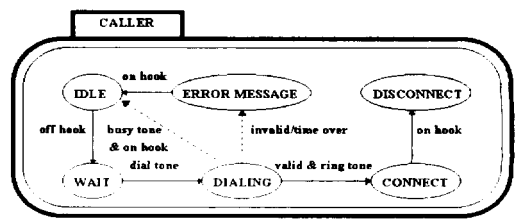
(그림 3-19) CALLEE 시나리오
(Fig. 3-19) Scenario of CALLEE

1. CALLER의 off hook을 인식한다.
2. CALLER에게 dial tone을 전송한다.
3. CALLER로부터 입력되는 전화번호를 인식한다.
4. 일정 시간이 지나도 입력이 완료되지 않으면 CALLER에게 error message를 전송한다.
5. 입력된 전화번호를 분석한다.
6. 분석 결과 전화번호에 오류가 없으면 CALLEE의 상태를 조사한다.
7. 오류가 발생이면 error message를 CALLER에게 전송한다.
8. CALLEE의 상태에 따라 busy tone, ring tone을 전송 한다.
9. CALLER와 CALLEE를 연결한다.
10. CALLER와 CALLEE의 on hook을 인식한다.
11. 연결을 해제한다.

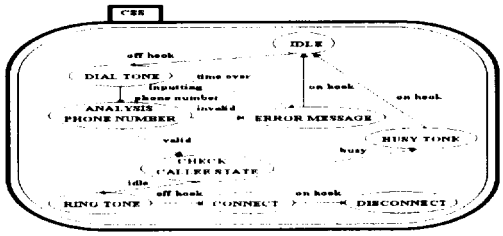
(그림 3-20) CSS 시나리오
(Fig. 3-20) Scenario of CSS

단계 4 : 시스템 상태 모델

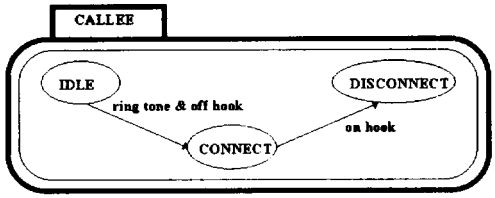
각각의 시나리오에 의해 작성된 시스템 상태 모델은 (그림3-21), (그림 3-22), (그림 3-23)과 같다.



(그림 3-21) CALLER 시스템 상태 모델
(Fig. 3-21) System state model of CALLER



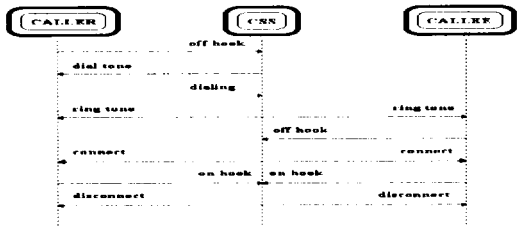
(그림 3-22) CSS 시스템 상태 모델
(Fig. 3-22) System state model of CSS



(그림 3-23) CALLEE 시스템 상태 모델
(Fig. 3-23) system state model of CALLEE

단계 5 : 시스템 메세지 모델

구현될 시스템 CSS와 상호 작용하는 CALLER, CALLEE에 대한 메세지의 흐름을 모델링한 시스템 메세지 모델은 (그림 3-24)과 같이 나타낼 수 있다.



(그림 3-24) CSS 시스템 메세지 모델
(Fig. 3-24) System message model of CSS

단계 6 : 시스템 명세서

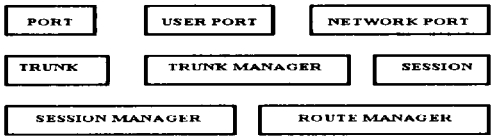
시스템 관점에서의 마지막 단계로 시스템 모델, 시스템 상태 모델, 시스템 메세지 모델에서 알려진 정보를 (그림 3-25)와 같이 명세서로 작성한다.

단계 7 : 클래스 모델

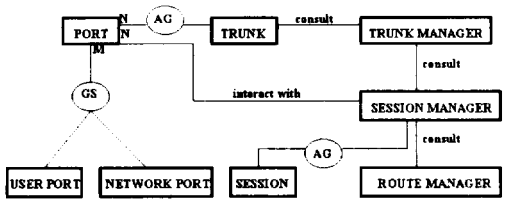
구현될 시스템 CSS를 구성하는 클래스를 추출하고 (그림 3-26), 그들간의 관계성과 다중성을 정의하여 다음의 클래스 모델을 얻을 수 있다(그림 3-27).

항 목	내 용
1. 이 름	CSS
2. 상호작용하는 외부 시스템	CALLER CALLEE
3. 입 력	hook phone number
4. 출 력	dial tone ring tone busy tone error message
5. 설 명	일정 지역 내 두 사용자의 호 교환 시스템

(그림 3-25) CSS 시스템 명세서
(Fig. 3-25) System specification of CSS



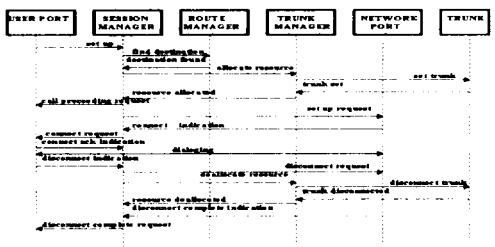
(그림 3-26) 추출된 클래스
(Fig. 3-26) Extracted classes



(그림 3-27) CSS 클래스 모델
(Fig. 3-27) Class model of CSS

단계 8 : 클래스 메세지 모델

구현될 시스템 CSS를 구성하는 클래스들 간의 메세지 흐름을 나타내면 다음 (그림 3-28)과 같다.

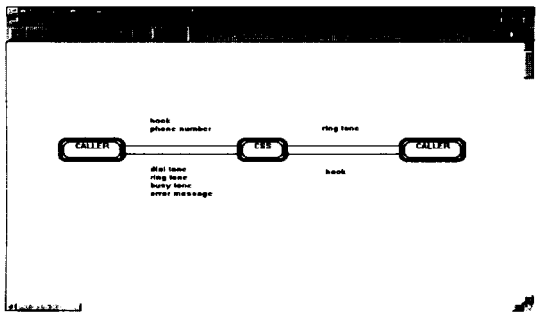


(그림 3-28) CSS 클래스 메세지 모델
(Fig. 3-28) Class message model of CSS

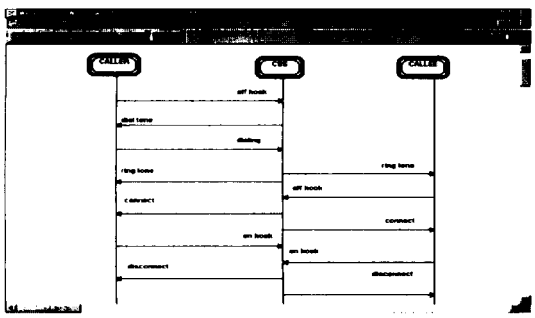
4. 실행 예 및 평가

4.1 실행 예

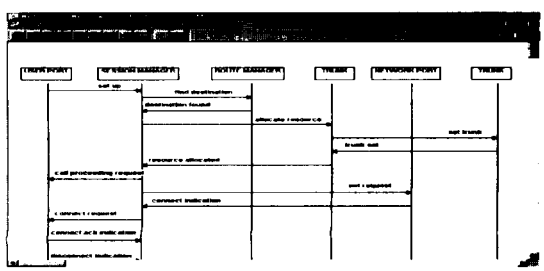
AOM에 의해 호 교환 시스템을 모델링한 결과로 생성된 시스템 모델, 시스템 메시지 모델, 클래스 메시지 모델은 모델링 도구인 With Class 3.0에서 실행시킨 결과로 (그림 4-1), (그림 4-2), (그림 4-3)과 같다.



(그림 4-1) 시스템 메시지 모델
(Fig. 4-1) System message mode



(그림 4-2) 시스템 모델
(Fig. 4-2) System model



(그림 4-3) 클래스 메시지 모델
(Fig. 4-3) Class message model

CSS의 시스템 모델에 대한 템플릿를 C++ 코드로 작성하면 다음 (그림 4-4) (4-5)와 같다.

```
# include "CALLER.h"
// Default construct alternative to compiler provided
default constructor
// Association object data member pointers initialized to
null association object
CALLER :: CALLER () : hook ()
{// Initialization of array of 1:M association objects to
null association objects}
// Operator = Assignemt Operator alternative to compiler
provided operator
 CALLER& CALLER :: operator = (const CALLER&
aCALLER)
{ if (this == &aCALLER) return * this;
   int i = 0;
   hook = aCALLER.hook;
   phone number = aCALLEE.phone number;
   return * this; }
```

(그림 4-4) CALLER 템플릿
(Fig. 4-4) Template of CALLER

```
#include "CSS.h"
// Default constructor alternative to compiler provided
defailt constructor
// Association object data member pointers initialized to
null association object
CSS :: CSS ( ) : dialtone (), ring tone(), busy tone(),
error message()
{// Initialization of array of 1:M association objects to
null association objects }
// Operator = Assignment Operator alternative to
compiler provided operator =
CSS& CSS :: operator = (const CSS& aCSS)
{ if (this == &aCSS) return * this;
  int i = 0;
  dial tone = aCSS.dial tone;
  ring tone = aCSS.ring tone;
  busy tone = aCSS.busy tone;
  error message = aCSS.error essage;
  return *this; }
```

(그림 4-5) CSS 템플릿
(Fig. 4-5) Template of CSS

4.2 평가

분석 지향 모델링 AOM을 정의하고 사례 연구로 호 교환 시스템을 모델링하는 과정에서 접근 방법, 지원하는 다양한 관점, 표기법의 이해성, 설계 및 구현 특징의 지원 여부, 관계성과 예외 지원 등의 평가 항목으로 다른 방법론들과 비교 평가해 보았다.

〈표 4-1〉 평가 결과
〈Table 4-1〉 Result of evaluation

방법론 / 평가 항목		OOAD	OMT	OOA	OSA	Unified Method
접근 방법	Outside-in	Bottom-up				
시스템 관점	○	×	×	×	×	○

클래스 관점	○	△	△	△	○	○
표기법 이해성	용이	복잡	일관성 부족	단순	복잡	단순
속성	×	△	△	○	×	○
오퍼레이션	×	△	△	○	×	○
프로세스	×	△	△	×	×	○
관계성	연결 일반화 집단화	연결 상속 소유 사용	연결 일반화 집단화	인스턴스 / 메시지 연결 일반화 집단화	연결 일반화 집단화	연결 일반화 집단화/ 상속 사용
예외	△	×	×	×	△	○

5. 결론 및 향후 연구 과제

기존의 객체지향 분석 방법론들은 문제 해결을 위해 다루어야 할 실세계의 객체들을 컴퓨터로 사상시킨 소프트웨어 객체들이 실세계의 객체 행동을 반영하고 해결함으로 소프트웨어의 생산성과 유지보수에 효율적이였으나 개발 과정에 있어 분석과 설계에 대한 경계가 불분명하며 설계 및 구현 특징이 반영됨으로 순수한 분석이 이루어지고 있지 않음을 알 수 있었다. 객체지향 분석 지원을 위한 모델링 기법 AOM(Analysis Oriented Modeling)은 분석과 설계의 경계를 명확히 하기 위하여 필수적인 객체지향 개념만을 도입하여 일관성 있고 알기 쉬운 표기법으로 정의하고 이를 지원하는 툴을 설계하였으며 사례 연구 "호 교환 시스템(CSS : Call Switching System)"을 통해 모델링 하였다. AOM은 문제 영역에 대한 다양한 관점을 지원하기 위하여 Outside-in 접근 방식으로 시스템 관점에서부터 분석을 시작하여 클래스 관점으로 확장해 나아감으로 시스템 식별이 용이하고 완성도 높은 분석이 이루어질 수 있다. 시스템 외부에서부터 먼저, 사용자의 요구사항을 문서화한 문제 기술서를 작성하고 이를 기초로 시스템 모델, 시나리오, 시스템 상태 모델, 시스템 메시지 모델을 통해 시스템 관점으로 분석하여 전체 문제 영역에서 구현될 시스템과 상호 작용하는 외부 시스템을 식별하여 인터페이스를 정의하고 각 시스템 내에서 일어날 수 있는 상태의 변화를 모델링하고 시스템과 시스템 간의 메시지 흐름을 알기 쉽게 간단한 표기법을 사용하여 모델링하였다. 클래스 관점은 구현될 시스템에 대하여 클래스를 추출하여 다중성을 가진 클래스들간의 유기적인 관계를 정의하고 클래스들간에 일어날 수 있는 메시지 흐름을 나타내는 클래스 모델과 클래스 메시지 모델로 이루어진다. AOM에 의해 생성된 모델을 CASE에서 실행시키고 시스템 모델에서 식별된 시스템에 대하여 C++ 코드 형식의 템플릿을 작성하였다.

향후 연구 과제로는 다양한 실세계 영역에 대하여 AOM의 8단계에 따라 적용하여 보고 제안된 AOM에 대한 메타 모델과 정형화를 위한 형식 언어를 정의하며 설계된 AOM 지원 툴의 구현 등이 있다.

참 고 문 헌

[1] G. Hoydalsvick, G. Sindre, "On the purpose of Object-Oriented Analysis", OOPSLA, pp. 240-255, 1993.

[2] D. Embley, R. Jackson, S. Woodfield, "OO Systems Analysis : Is It or Isn't It?", IEEE SOFTWARE, pp.19-32, July 1995.

[3] 이경환, 객체모델링 방법론, 중앙대학교출판부, 1996.

[4] G.Booch, Object-Oriented Analysis and Design with Application, Benjamin/Cummings, 1994.

[5] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen, Object-Oriented Modeling and Design, Prentice Hall, 1991.

[6] S.Shlaer, S.Mellor, Object Life Cycle Modeling The World in States, Yourdon Press, 1991.

[7] P. Coad, E. Yourdon, Object-Oriented Anal-yis, Prentice Hall, 1991.

[8] D.Embley, B.Kurtz, S.Woodfield, Object Oriented System Analysis : A Model Driven Approach, Yourdon Press, 1992.

[9] J. Rumbaugh, Unification of Object-Oriented Method, Rational Software Corporation, 1996.

[10] J. Rumbaugh, "To form a more perfect union : Unifying the OMT and Booch meth-ods", Journal of Object-Oriented Programming, pp.14-18, Jan. 1996.

[11] H.K.Kim, D.S.Han, J.E.Cho, "REdefine of Object-Oriented System Analysis about Nota-tions", High-New Technology and Traditional Industry, pp.354-359, July 1996.

[12] D. Cheung, "ATM Software Analysis and Design", Dr. Dobb's Journal, pp.70-76, Oct. 1996.

김 행 곤

1985년 중앙대학교 전자계산학과 졸업(학사)
1987년 중앙대학교 대학원 전자계산학과 졸업(이학석사)
1991년 중앙대학교 대학원 전자계산학과 졸업(공학박사)
1978년~1979년 미 항공우주국 객원 연구원
1987년~1989년 AT&T 객원 연구원
1990년~현재 대구효성가톨릭대학교 컴퓨터공학과 부교수
관심분야 : 객체지향시스템 설계, 사용자 인터페이스, 소프트웨어 재공학, 유지보수 자동화툴, CASE