韓國事務自動化學會誌
第2卷 第1號, 1995. 6

객체 지향 대수적 명세서 시스템의 설계 및 구현

김 영 란*

1. 서 론

1.1 연구의 배경

오늘날 소프트웨어 시스템에 대한 사용자의 요구사항이 복잡·다양해지고 있고, 또한 소프트웨어 개발 과정중에 사용자 요구사항에 대한 변경이 번번히 요구됨으로 인해, 대규모 소프트웨어에 대한 개발비용 증대 및 개발기간이 장기화되고 있다. 이러한 문제점을 해결하기 위한 방안으로 대두된 개념이 객체 지향이다. 객체 지향 개념은 소프트웨어 생명주기의 전단계에 적용되고 있으며, 특히 분석 및 설계단계에 적용되는 객체 지향 방법론에 관한 연구는 매우 활성화되어 있는 실정이다[1][2]. 반면에 규모가 크고, 복잡하고, 신뢰성 있는 소프트웨어 시스템을 개발하는데 있어서 핵심적인 역할을 수행하는 정형적인 객체 지향 명세화에 관한 연구는 활성화되지 않고 있으며, 객체 지향 시스템의 명세화에 관한 근본적인 문제점들은 아직까지 연구되지 않고 있는 실정이다[3].

소프트웨어에 대한 규격화 연구는 비정형적인 언어가 갖는 비효율적인 추상화, 정밀성의 결여, 애매 모호한 사항들의 존재 가능성 등의 단점을 보완하기 위하여 다양한 방법으로 수행되어 왔다. 그러나 소프트웨어 공학에서 새로운 파라다임으로 제시하는 객체 지향 개념과 객체 지향 분석, 객체 지향 설계, 객체 지향 프로그래밍을 포용할 수 있는 규칙화 방안은 아직까지 제시되고 있지 못한 실정이다.

본 논문에서는 소프트웨어 라이트 사이클의 요구단계 및 설계단계에서 이용될 수 있는 방안으로써, 객체 지향 소프트웨어를 수학적 개념으로 간결하고 명확하게 표현할 수 있는 객체 지향 규격화방안을 제안하였으며, 객체의 설정에 따르는 객체의 정확한 성질을 규격화하는 객체 지향 대수적 시스템을 구현하였다. 이러한 연구를 수행함으로써 기대되는 효과는 다음과 같다. 첫째, 객체 지향 소프트웨어 표준화에 대한 기초적인 이론이 정립된다. 둘째, 신기술인 객체 지향 기법을 통신용 소프트웨어에 적용함으로써 외국기관(ITU·T)과의 이해의 범용성을 추구할 수 있다. 세째, 객체 지향, 개념을 적용함으로써 사용자와 개발자간의 인터페이스가 향상되어 사용자의 요구사항 변경에 신속히 대처할 수 있다. 네째, 객체 또는 클래스 단위들로 구성된 소프트웨어 라이브러리에 있는 관련 정보의 속성을 객체 지향 규격화 기법에 의해 명확하게 제시해줌으로써, 소프트웨어의 재사용성을 향상시키고 소프트웨어 개발 기간을 단축시키게 된다. 결국, 기능 수정 및 보완이 용이해짐으로써 소프트웨어 생산성을 향상시킬 수 있다.

* 충청전문대학 사무자동화과

1.2 연구의 내용 및 범위

본 논문의 내용은 크게 객체 지향 규격화 연구를 수행하는데 관련된 개념 조사와 객체 지향 소프트웨어 명세화 시스템 설계 및 구현으로 구성된다. 관련 연구로는 추상화 개념, 소프트웨어 명세화 개념, 객체 지향 개념을 조사하였으며, 객체 지향 소프트웨어 명세화 시스템은 본 논문에서 제시된 객체 지향 규격화 기법을 기반으로 하여 구현하였다.

연구의 범위는 가장 널리 활용되고 있는 방법론 중의 하나인 Rumbaugh의 OMT방법론을 구성하는 객체 모델, 기능 모델, 동적 모델 중에서도 기능 모델을 명세화하는 자료 흐름도에 관한 규격화 연구로 제한하였다. 왜냐하면, 객체 모델을 명세화하는 객체 다이어그램과 동적 모델을 명세화하는 상태 다이어그램은 구문 뿐만 아니라 시맨틱스가 정형화되어 있는데 반해서, 기능 모델을 명세화하는 자료 흐름도의 구문은 정형화되어 있지만, 시맨틱스 측면에서 볼 때 정형적인 면이 부족하다는 단점을 갖고 있기 때문이다.

2. 관련 연구

2.1 추상화 개념

대규모의 소프트웨어를 개발하는데 대두되는 중요한 문제는 세부사항(details) 및 개념적 복잡도(conceptual complexity)를 줄이는 것이다. 복잡도를 줄이기 위한 효율적인 방법으로 분해(decomposition)와 추상화(abstraction)개념이 있다[5][6].

분해란 하나의 타스크를 여러개의 부타스크로 분리(factoring)하는 방법으로서, 다음과 같은 사항을 만족하도록 수행되어야 한다. 첫째, 각 부타스크는 동일한 수준의 구체성(datailness)을 갖어야 한다. 둘째, 각 부타스크는 독립적으로 해결이 가능해야 한다. 세째, 각 부타스크에 대한 해결책들이 결합되어 본래의 문제를 해결해야 한다.

추상화란 구체성에 대한 수준을 달리하면서 적절한 분해가 이루어지도록 지원하는 방법으로서, 타스크에 대해 중요하지 않은 세부항목은 삭제하고 필수적인 특성과 속성만을 강조하는 개념이다.

소프트웨어 설계 단계에서 추상화는 구조적 특징, 데이타 흐름, 데이타 구조가 설정될 때까지 구조상의 고려사항 및 알고리즘상의 고려사항을 연관시키면서, 시스템의 특성을 구성한다.

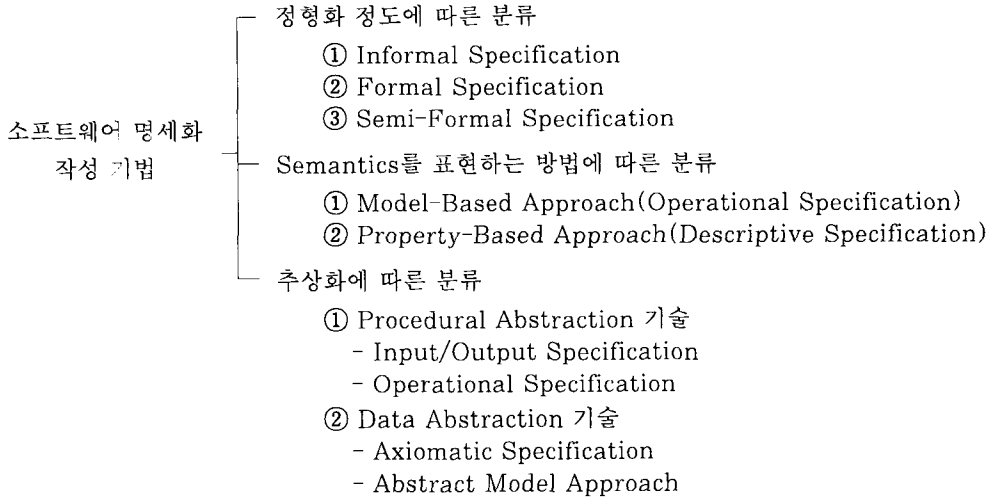
소프트웨어를 설계하는데 있어서 복잡도를 제어하는 대표적인 매카니즘인 추상화 방법으로서, 절차적 추상화(Procedural abstraction)와 데이타 추상화(Data abstraction)가 있다[8][13].

2.2 소프트웨어 명세화 개념

일반적으로 시스템 명세서는 생산자와 소비자간의 의견일치 사항에 대한 문장이지만, 경우에 따라서 구현자와 사용자가 다르고 명세서의 개념이 다르다[9].

명세서와 구현간의 관계는 "무엇"과 "어떻게"간의 관계로 설명될 수 있다. 즉 구현자는 시스템이 어떻게 수행되는가를 결정하고, 명세서 작성자는 시스템이 무엇을 수행해야 하는가를 결정한다.

소프트웨어 명세서를 기술하는 기법은 정형화 정도에 의한 분류기준, 표현방법에 의한 분류 기준, 추상화 개념에 의한 분류 기준에 따라 세분화된다. 형식에 의한 분류 기준에 따르면 비형식적 명세화 기법과 정형적 명세 기법으로 분류되고, 표현 방법에 의한 분류 기준에 따르면 기능적 명세 기법과 기술적 명세기법으로 분류되고, 추상화 개념에 의한 분류 기준에 따르면 절차적 추상화 명세기법과 데이타 추상화 명세기법으로 분류된다.



2.3 객체 지향 개념

객체 지향 개념은 최초의 객체 중심 프로그래밍 언어라 할 수 있는 Simula에서 소개되었고, 개념 정립은 대화식이며 디스플레이 중심으로 구현된 Smalltalk 언어에서 이루어졌다.

소프트웨어 개발에 있어서 객체 지향이란 데이타 구조(data structure)와 행위(behavior)의 특성을 포함하는 객체(object)들의 집합으로 소프트웨어를 구성하는 소프트웨어 개발 기법을 의미한다[16].

이 절에서는 객체 지향의 기본 개념인 객체(object), 클래스(class), 상속(inheritance), 정보 은닉(information hiding/encapsulation), 다형성(polymorphism)을 간략히 기술하였다.

2.3.1 객체(Object)

객체란 필요한 데이타 구조와 그 위에서 수행하는 함수들을 가진 하나의 소프트웨어 모듈로서, 캡슐화와 자료의 추상화로 설명된다. 데이타 구조는 데이타 화일, 디렉토리, 하드웨어 자원 등에 해당되는데, 이들 데이타 구조는 작업중에 그 상태가 변화될 수 있다.

각 객체는 캡슐화되어 있기 때문에, 외부적으로 보여지는 객체의 행위는 구체적이 아닌 추상적인 특성을 지니게 된다. 객체는 내부의 상태와 그 객체가 수행하는 연산의 구현에 대한 사항들을 외부적으로 나타내지 않는다. 객체의 외부적인 행위를 연산이라 하고, 객체 내부의 데이타와 연산들 및 다른 객체들과의 대화를 가능하게 하는 인터페이스 연산들을 메소드라고 한다. 객체를 사용하는 사용자는 객체의 연산이 어떤 메소드로 구현되었는가와는 상관없이 추상적으로 정의된 연산을 통하여 객체를 사용할 수 있고, 또한 객체의 내부적인 상태를 변화시킬 수도 있다. 이러한 특성을 자료 추상화라 하며, 이는 데이타 형태에 대한 이해를 높이고, 프로그램의 유지보수나 개량을 용이하게 하는데 매우 중요한 개념이다.

2.3.2 클래스(Class)

클래스는 하나 이상의 유사한 객체들을 묶어서 하나의 공통된 특성으로 표현한 것이며, 상대적으로 객체는 클래스로부터 만들어진 인스턴스(instance)라고 정의한다. 클래스로부터 만들어진 객체들은 모두 같은 연산을 갖고 있으며, 메소드에 대한 프로그램 코드는 공유되고 있다. 따라서, 한 클래스에서부터 만들어진 객체들은 모두 동일한 행위를 하게 된다.

다음 그림 2.1은 Car 클래스를 정의한 예로써, 클래스의 정의는 대개 클래스 명칭, 메타클래스, 상위클래스, 애트리뷰트, 셀렉터(selector), 그리고 메소드로 이루어진다[20]. 메타클래스에서는 그 클래스의 클래스를 지정하고, 상위클래스에서는 클래스가 상속받은 클래스들을 나열한다. 애트리뷰트의 정의는 객체의 상태를 저장할 변수들의 정의로 이루어지는데 변수에는 클래스 변수와 인스턴스 변수가 있다. 클래스 변수는 클래스에 저장되는 변수로 그 클래스에 속하는 모든 인스턴스에 의해 공유되는 반면에, 인스턴스 변수는 각 인스턴스에 저장되고 그 실객체의 상태에 대한 정보를 가지고 있는 변수이다. 셀렉터의 정의는 외부 인터페이스를 정의하는 부분으로 셀렉터와 이에 대응되는 메소드들의 명칭으로 구성된다. 즉 객체는 메세지를 전달받으면 그 메세지의 셀렉터가 가리키는 메소드를 실행한다. 단, C++과 같은 몇몇 객체 지향 언어에서는 메소드의 명칭이 바로 그 메소드와 연관된 셀렉터가 되기 때문에 셀렉터를 별도로 정의하지 않는다.

ClassName	Car
NetaClass	Class
SuperClass	(Vehicle)
ClassVariables	
	TotalCarCount
InstanceVariables	
	owner
	engine
	totalweight
Methods	
	Drive
	Park

그림 2.1 Car 클래스의 정의

2.3.3 상속(Inheritance)

상속은 한 클래스에서 정의된 메소드들을 다른 클래스가 공유하도록 함으로써, 정보를 개념적으로 조직화하는 기능을 제공한다. 여기서 상속을 받는 클래스를 하위 클래스(subclass)라 하고, 상속을 주고 클래스를 상위 클래스(superclass)라 한다. 이는 클래스들 간의 계층구조(class hierarchy)로 나타내는데, 하위 클래스는 상위 클래스의 속성인 지역변수들과 메소들들을 상속받고, 새로운 지역변수 및 메소드들을 정의할 수 있다.

다음 그림 2.2는 객체 지향 시스템 내의 클래스 계층 구조의 예로써, 클래스를 하나의 노드로 나타내고 상속관계를 간선(edge)로 나타냈다. 이와 같은 단일 상속(single inheritance)의 경우는 트리구조로 나타난다.

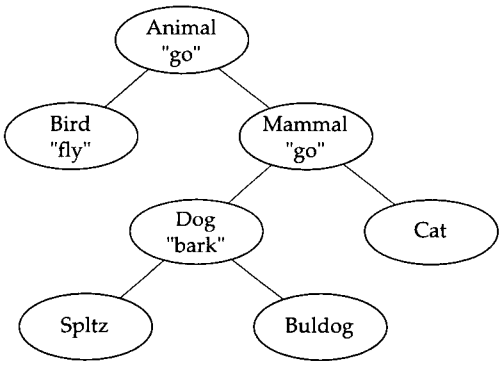


그림 2.2 Animal 클래스 계층 구조

다중 상속(multiple inheritance)은 단일 상속과는 달리 하나의 클래스가 여러 개의 클래스를 상위클래스로 할 수 있는 상속 구조이다.

상속은 프로그램의 재사용(reusability)을 보다 수월하게 한다. 방대한 소프트웨어를 작성할 경우에, 이러한 코드 재사용은 소프트웨어 작성 비용를 많이 줄일 수 있다. 한번 잘 정의된 클래스들은 클래스 라이브러리에 보관해 두고, 이후 다른 프로그램 작성시에 가져다 쓸 수 있고, 또한 연관성 있는 프로그램에서는 새로운 클래스들과 계층 관계화하여 상속을 통한 코드 재사용을 할 수 있다.

2.3.4 정보은닉(Information Hiding/ Encapsulation)

정보은닉은 객체의 근본적인 특성들에 도움을 주지 않는 모든 세부사항을 숨기는 과정이다. 추상화와 정보은닉은 보충적인 개념으로서, 추상화는 객체의 외부 모양에 중점을 두고, 정보은닉은 추상화의 행위가 수행되는 객체의 내부 특성을 나타내지 않는 것이다.

정보은닉의 잇점은 객체 내부의 은닉된 데이타 구조가 변하더라도 주변 개체들에게 영향을 주지 않는다는 것이다. 이는 객체의 은닉된 데이타 구조가 외부에 공개되지 않기 때문이다.

2.3.5 다형성(Polymorphism)

다형성은 실제 메시지가 어떤 데이타 타입으로 어떻게 구현되었는가를 모르더라도, 원하는 결과를 얻기 위해서 일반적으로 정의된 메세지를 사용할 수 있도록 해주는 기법이다. 즉, 매개변수에 의해 표현되는 자료형들을 여러가지로 정의해서 사용하기 위한 기술방법이다. 다형성의 가장 단순한 형태가 중복표현(overloading)으로서, 동일한 이름을 여러가지 의미로 확장하여 사용할 수 있도록 하는 방법이다.

다형 함수는 여러 종류 타입의 매개변수들 모두 처리해 줄 수 있는 함수이다. 객체 지향에서는 다형성을 사용하여 같은 메세지를 상위 및 하위 클래스에 보낼 수 있다.

3. 객체 지향 대수적 명세서의 규격화 방안

3.1 객체 지향 소프트웨어의 명세화 메카니즘

객체 지향 명세서는 객체 지향 분해와 추상화 개념을 정형적으로 기술하는 것으로서, 객체 지향 소프트웨어에 대한 명세서를 구성하는 방법은 그림 3.1에서 보는 바와 같이 크게 두가지로 구분된다.

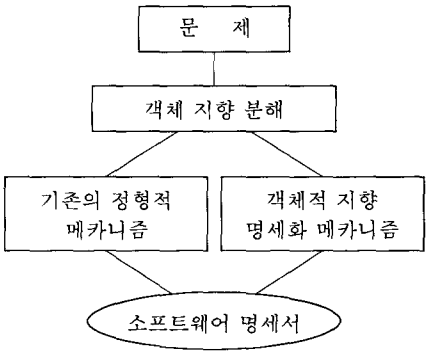


그림 3.1 객체 지향 소프트웨어 명세서에 대한 접근방법

첫번째 방법은 기존의 정형적 명세화 메카니즘을 활용하는 방법으로서, 이 기법은 설계자가 기존의 명세서와 동일한 명세언어를 사용하여 작성된 새로운 명세서간에 일치선을 유지할 수 있다는 장점을 작고 있는 반면에, 기존의 정형적 명세화 메카니즘 기능적 분해를 기술하도록 설계되었기 때문에 객체 지향 분해 기법에 의해 분석된 시스템의 특성을 제대로 반영하지 못하는 단점이 있다. 두번째 방법은 기존의 객체 지향 소프트웨어 방법론을 정형화하는 접근 방법으로서, 대표적인 예가 OMT와 Booch 방법에 대한 정형화이다.

본 논문에서는 가장 널리 활용되고 있는 객체 지향 방법론인 Booch 방법론과 OMT 방법론 중에서도 OMT 방법론에 대한 규격화 연구를 수행하였다. 특히 OMT 방법론에서 기능적 모델을 표현하는 자료 흐름도의 시멘틱스를 정형화하는 방법을 제안하였고, 제안된 방법을 실험하기 위한 객체 지향 대수적 명세화 시스템(OOASS : Object-Oriented Algebraic Specification System)을 설계하였다.

3.2 자료 흐름도의 정형화된 시맨틱스 구성 방안

자료 흐름도는 시스템의 프로세서와 이러한 프로세스들을 연결하는 데이타를 네트워크로 표현하는 구조적 분석의 주된 도구이며 시스템을 여러 개의 프로세서로 분리하는데 사용된다. 이것으로 시스템이 어떤 일을 하는지는 알 수 있지만 어떻게 하는지는 알 수가 없다. 자료 흐름도는 시스템의 프로세서를 보다 상세히 보여주기 위해 하향식 접근법으로 사용된다. 이러한 자료 흐름도는 배우고 이해하기가 쉽다는 장점 때문에 학계 뿐만아니라 연구소에서도 널리 사용되고 있다. 반면에, 요구명세에 대한 정형적인 면이 부족하여 실제 산업계에서는 적용이 쉽지 않은 것이 사실이고, 이러한 약점을 보완하기 위하여 기존의 요구분석 기법에 많은 확장이 이루어져 왔다.

자료 흐름도와 페트리 네트의 그래픽 표현은 매우 유사하다. 페트리 네트는 1960년대에 독일학자 Carl Adam PETRI에 의해 고안된 도구로써, 이 모델은 비결정적이며, 비동기적이고, 동시적인 시스템을 분석하고 기술하기 위한 모델링 도구로써 널리 사용되어져 왔다.

본 논문에서는 이와 같은 페트리 네트의 장점을 기반으로 하여, OMT 방법론의 기능적 모델을 표현하는 자료 흐름도의 정형적인 시맨틱스를 구성하였다.

3.3 자료 흐름도를 페트리 네트로 변환

본 논문에서는 자료 흐름도의 정형적인 시맨틱스를 구성하기 위해서 자료 흐름도를 페트리 네트로 전환하였으며, 전환 방법은 자료 흐름도의 프로세스를 페트리 페트의 전이로 대응시키고 자료 흐름도의 자료 흐름은 페트리 네트의 장소로 대응시켰다. 이와 같은 전환을 수행함으로써 수학적 개념을 내포하는 네트리 네트를 대수적 기법으로 표현함으로써 자료 흐름도의 비정형성 문제를 해결할 수 있으며, 또한 페트리 네트의 그래픽 표현을 대수적 표현으로 나타냄으로써 명세서를 분석하는데 필요한 이론적인 개념을 제공할 수 있게 된다.

자료 흐름도를 페트리 네트로 전환하는데 있어서, 자료 흐름이 여러 개인 경우에 자료 흐름간의 관계로 인한 문제점이 대두된다. 예를 들면 임의의 프로세스 P_i로 입력되는 자료 흐름이 2개인 경우에 프로세스 P_i가 작동하기 위한 조건은 여러가지가 있을 수 있다. 즉, 모든 자료 흐름으로 부터 자료가 이동되어야만이 프로세스 P_i가 작동하는 경우가 있을 수도 있고, 아니면 둘중에 어느 한 자료 흐름으로 부터 자료가 이동되어지면 프로세스 P_i가 작동하는 경우가 있을 수 있다. 이러한 자료 흐름의 애매 모호성을 해결하기 위해서, 본 연구에서는 조합 논리를 갖는 자료 흐름도를 제시하였다. 조합 논리는 자료 흐름도의 프로세스 작동의 순서에 대한 모호성을 통제할 수 있는 논리로서, AND와 OR 논리로 구성된다. AND 논리는 "*" 기호로 나타내고 OR 논리는 "+" 기호로 나타낸다. 이러한 조합 논리를 페트리 네트로 전환하는 규칙은 다음 그림 3.2와 같이 구성하였다.

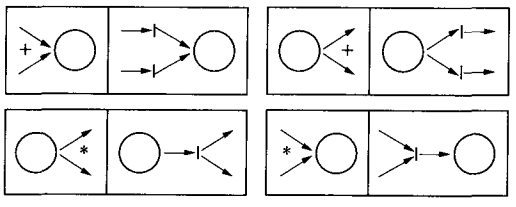


그림 3.2 조합 논리를 페트리 네트로 전환하는 규칙

이러한 전환 규칙을 적용하여 주문서 작성 자료 흐름도를 표현하면, 다음 그림 3.3과 같다.

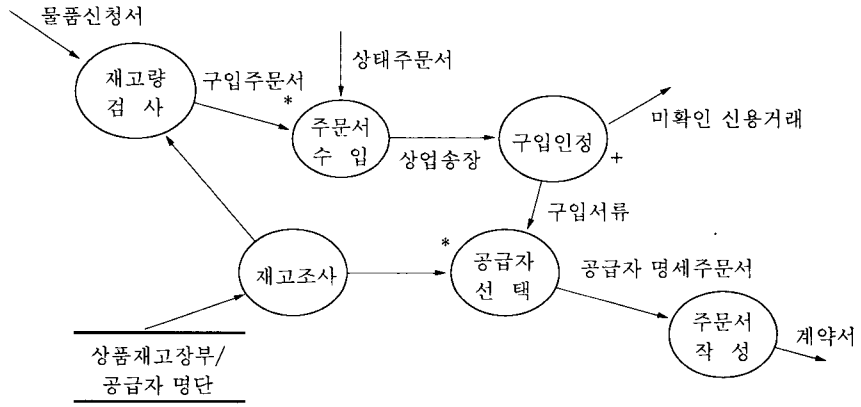


그림 3.3 상세화된 주문서 작성 자료 흐름도

여기서 기호 "*"와 "+"는 조합 논리를 의미하는 것으로서, 공급자를 선정하는 프로세스는 재고 조사 프로세스로부터 출력되는 데이타와 구매 승인 프로세스로부터 출력되는 데이타를 동시에 입력으로 받아들인 후 실행을 한다. 즉 재고조사 프로세스와 구매 승인 프로세스 중에서 어느 하나의 결과가 출력되지 않은 상태라면 공급자 선정 프로세스는 작동할 수 없음을 나타낸다.

AND와 OR 조합 논리를 포함한 자료 흐름도를 그림 3.2의 전환 규칙에 의해서 전환된 페트리 네트는 다음 그림 3.4와 같다.

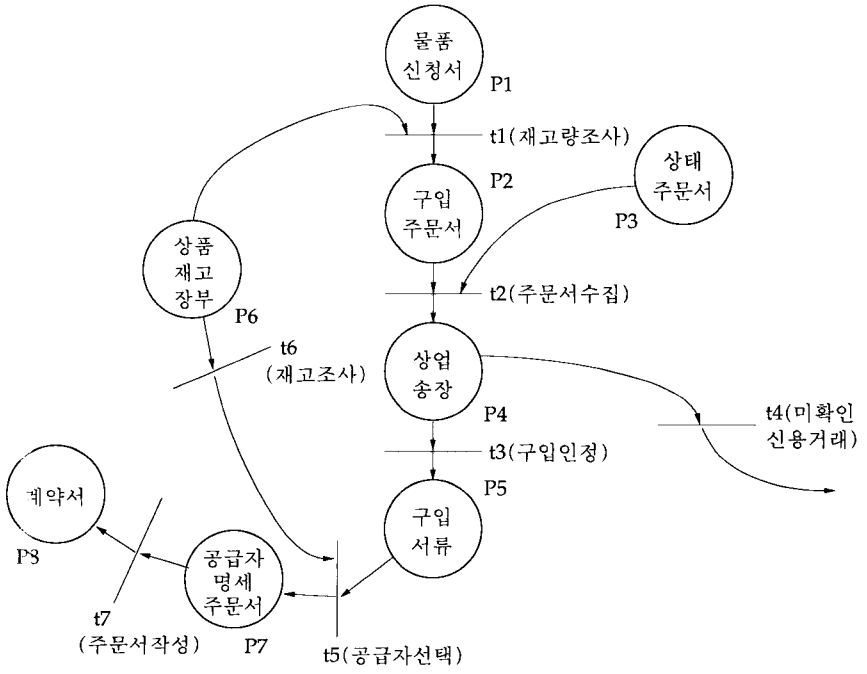


그림 3.4 주문서 작성 자료 흐름도를 전환한 페트리 네트

3.4 객체 지향 대수적 명세서의 템플리트 구성

본 논문에서는 객체 지향 개념을 가장 정형화된 형태인 대수적인 방법으로 기술하기 위한 템플리트를 그림 3.5와 같이 구성하였다. 템플리트는 기본적인 명세서 단위로서, parameters, exports, functions, imports, variables, 그리고 equations 등의 6부분으로 구성된다.

```
Module          /* 모듈 이름을 기술 */
begin
  parameters    /* 파라메터명, carrier sets 정의 */
      ⋮
  exports       /* 다른 모듈에서만 사용할 수 있는 carrier sets와
      ⋮            function을 기술 */
  functions     /* 현재 모듈에서만 사용할 수 있는 hidden function을 기술 */
      ⋮
  imports       /* 다른 모듈에서 선언된 것을 현재 모듈에서 사용할 수 있는
      ⋮            carrier sets을 기술 */
  variables     /* 변수 기술 */
      ⋮
  equations     /* 조건 방정식 또는 무조건 방정식을 기술 */
      ⋮
end
EndModule
```

그림 3.5 객체 지향 대수적 명세서의 템플리트

Parameters 부분은 현재의 모듈이 다른 구문에서 좀 더 일반적으로 이용될 수 있도록 기술되는 부분으로서, 대수적 함수인 sort와 function 선언문으로 구성되는 signature 부분이라고도 한다. 이와 같이 모듈의 signature를 형성함으로써 명세서의 정형성을 갖게 된다. Imports와 Exports 부분은 클래스의 가시성 리스트(visibility list)를 설정하는 부분으로서, exports에서 선언된 sorts와 function은 다른 모듈에서 visible할 수 있다. 따라서 exports와 imports는 객체 지향 개념 중의 상속성을 반영하는 부분이다. Functions 부분에서는 현재 모듈에서만 사용할 수 있는 hid-den 함수를 정의함으로써, 객체 지향 개념 중의 정보 은닉을 반영한다. Equations 부분에는 함수들의 기능을 대수적 연산자를 이용하여 방정식으로 기술하기 때문에 작성되는 명세서가 정형화된 성질을 갖도록 하는 부분이다.

4. 시스템 설계 및 구현

본 논문에서는 도형적으로 표현된 페트리 네트가 대수적으로 표현될 수 있다는 장점과 가장 정형화된 명세화 기법 대수적 명세화 기법이라는 점을 기반으로 하여 객체 지향 대수적 명세 언어를 설계하였고, 자료 흐름도에 대한 객체 지향 대수적 명세서를 생성하는 객체 지향 대수적 명세화 시스템(Object-Oriented Algebraic

Specification System : OOASS)을 개발하였다. 이 시스템의 구성절차는 그림 4.1과 같다.

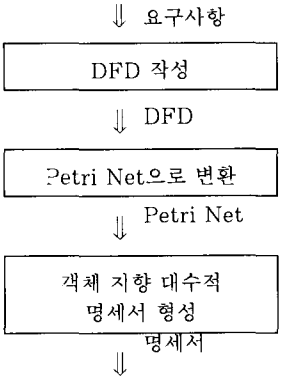


그림 4.1 OOASS의 구성 절차

4.1 시스템의 구성도

OOASS의 전반적인 구성도는 그림 4.2와 같다. 사용자는 요구명세를 편집하는 도구인 명세 편집기를 통하여 자료 흐름도를 작성하며, 이때 기술되는 자료 흐름의 이름, 저장소 이름, 그리고 프로세스 이름은 자료사전에 등록된다. 자료사전에 등록된 정보는 페트리 네트 변환기에 의해 참조되고, 변환된 페트리 네트의 그래픽 표현에 기술된다. 페트리 네트 변환기는 Rule-base Ⅰ에 등록되어 있는 규칙에 의해서 자료 흐름도를 페트리 네트로 변환하고, 객체 지향 대수적 명세서 출력기는 템플리트 자료 사전에 등록된 정보와 Rule-base Ⅱ에 등록되어 있는 규칙을 기반으로 하여 생성되는 명세서를 출력하고, 또한 이 명세서 모듈이 재사용되도록 하기 위해서 소프트웨어 라이브러리에 등록하는 절차를 수행한다.

객체 지향 명세화 시스템 구성도는 다음과 같이 정의된다.

OOASS = 〈사용자 인터페이스, 명세 편집기,
Petri Net 변환기,
명세서 출력기, 정보베이스〉

- 사용자 인터페이스

  객체 지향 명세화 시스템의 사용을 편리하게 하기 위해 윈도우 방식으로 설계됨

- 명세 편집기

  요구 명세서를 작성하기 위한 자료 흐름도 편집 도구로서 자료 흐름도의 기본 구성요소와 편집 메뉴를 버튼 형식으로 구성함

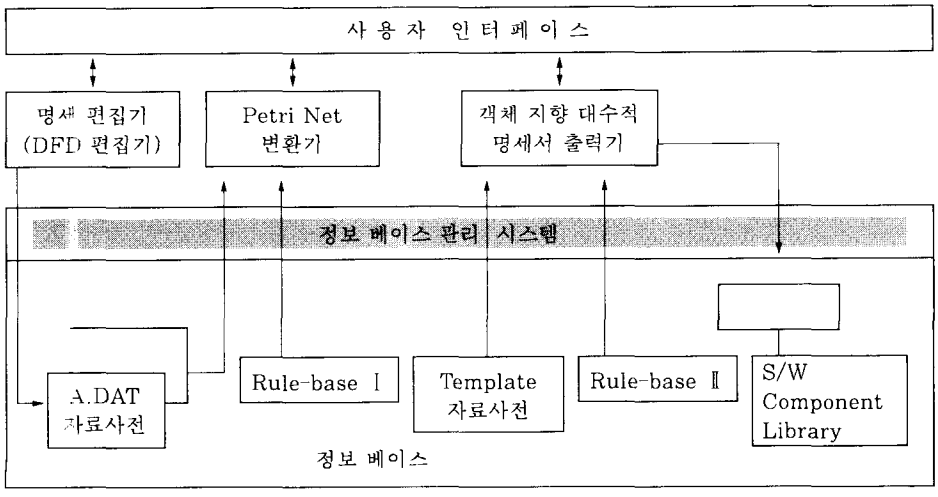


그림 4.2 객체 지향 대수적 명세서 시스템의 구성도

● 페트리 네트 변환기

자료 흐름도를 페트리 네트의 그래픽 표현으로 변환하는 도구로서 페트리 네트의 기본 구성 요소와 편집 메뉴를 버튼 형식으로 구성함

● 명세서 출력기

객체 지향 대수적 명세서의 템플리트를 기반으로 하여 페트리 네트의 대수적 개념을 명세화하고, 작성된 명세서를 출력하거나 소프트웨어 컴포넌트 라이브러리에 등록하는 기능을 수행함

● 정보 베이스

명세 편집기, 페트리 네트 변환기의 수행에 필요한 자료사전 및 Rule-base를 저장하고 있으며, 또한 객체 지향 대수적 명세서 출력기에 의해 생성되는 모듈 명세서를 저장하는 소프트웨어 컴포넌트 라이브러리로 구성됨

4.2 화면 설계

OOASS의 초기화면은 다음 그림 4.3과 같고, OOASS의 기본 메뉴 화면은 그림 4.4와 같다. 그림에서 보는 바와 같이 객체 지향 대수적 명세서 시스템의 기본 메뉴로는 자료 편집기, 페트리 네트 변환기, 명세서 편집기가 있다.

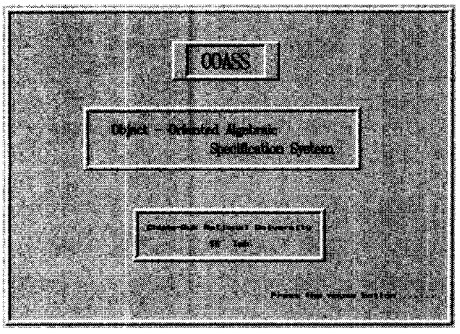


그림 4.3 OOASS의 초기화면

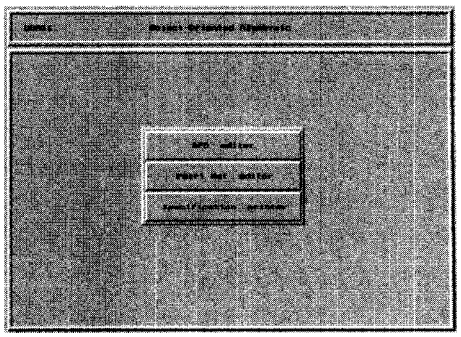


그림 4.4 OOASS의 기본 메뉴 화면

4.3 명세 편집기

사용자는 편집기를 통하여 자료 흐름도 편집 작업과 자료사전에 등록되는 프로세스 이름 및 속성을 입력시킬 수 있다. 명세 편집기는 윈도우 방식과 Pop-up 메뉴 방식을 통해 쉽게 접근할 수 있도록 구성하였으며, 또한 사용자가 쉽게 이용할 수 있도록 하기 위해 버튼과 메뉴 아이콘으로 구성하였다. 명세 편집기는 주 메뉴로써 도면 편집, 자료사전 등록, 도면 출력, 도움말 메뉴를 갖고 있으며, 도면 편집을 수행하는 부메뉴로써 도형 삽입, 도형 삭제, 이름 이동, 도형 이동의 메뉴를 갖고 있다. 다음 그림 4.5는 명세 편집기의 초기화면이고, 그림 4.6은 명세 편집기에서 도면 편집을 하는데 사용되는 Pop-up 방식의 부메뉴가 표시된 화면이다.

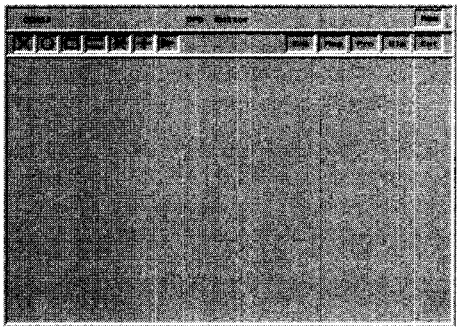


그림 4.5 명세 편집기의 초기 화면

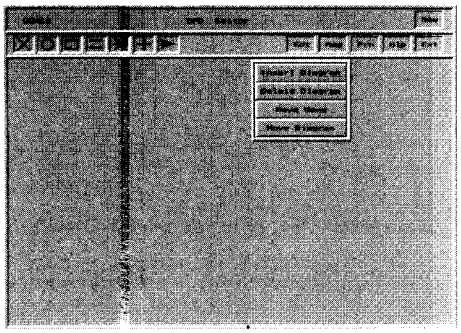
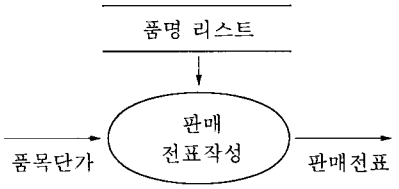


그림 4.6 도면 편집을 위한 Pop-up 방식의 부메뉴 화면

명세 편집기에 의해 작성된 자료 흐름도에는 프로세스명, 자료 흐름상에 기술된 입·출력되는 자료명, 조합 논리(AND, OR), 자료 저장소명 등의 정보가 표현되어 있다. 이러한 정보들은 명세 편집기에 의해 편집됨과 동시에 자료사전에 등록되어야 할 속성들이다. 만약, 자료사전이 없다면 자료 흐름도는 단지 시스템에서 무엇이 진행되고 있는지에 대해 대략적으로 보여주는 그림에 지나지 않는다. 따라서 자료 흐름도의 모든 원소들이 엄격하게 정의되었을 때 명세서가 될 수 있는 것이다 자료사전의 정의 대상으로는 정보 모임의 통로가 되는 자료 흐름, 더 이상 분해할 수 없는 자료 요소, 자료 흐름도의 일시적인 자료저장 등이 대상이 된다. 그림 4.7(b)는 (a)의 자료 흐름도에 수록되어 있는 정보들에 의해 구성되는 자료사전의 내용이다.

자료사전에 기술되는 속성들은 페트리 네트 변환기에 의해 참조될 정보들로서, 프로세스는 p, 자료 흐름은 f, 자료 저장소는 s, 단말은 t로 구분하였다. 명세 편집기를 사용하여 그림 4.7(a)에 있는 자료 흐름도를 편집하고 자료사전에 항목 및 속성을 등록하는 화면은 다음과 같다. 먼저, 사용자는 판매 전표 작성의 프로세스를 편집하기 위해 원 아이콘을 클릭한 후, 원하는 화면 위치로 커서를 이동시키고 마우스로 클릭하면, 그림 4.8에서 보는 바와 같이 화면에 프로세스를 나타내는 원이 나타난다. 이 상태에서 명세 편집기의 주메뉴인 자료사전 등록(Reg)을 선택하면 그림 4.9에서 보는 바와 같이 화면 하단에 항목명과 속성을 입력하기 위한 작은 윈도우가 열린다. 사용자는 프로세스 이름과 속성을 나타내는 문자를 입력하고 RETURN 키를 누르면 그림 4.10과 같은 화면이 구성된다.



(a) 자료 흐름도

항 목 명	속 성
품목단가	f
판매전표작성	p
판매전표	f
품명 리스트	s

(b) 자료 사전

그림 4.7 자료 흐름도와 자료 사전

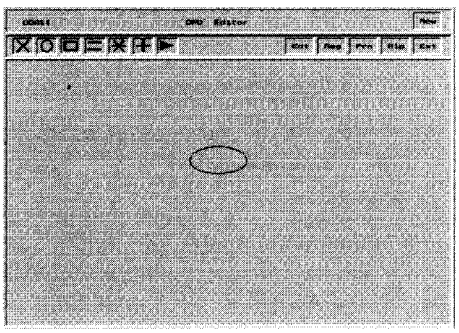


그림 4.8 자료 흐름도의 구성 요소를 클릭한 화면

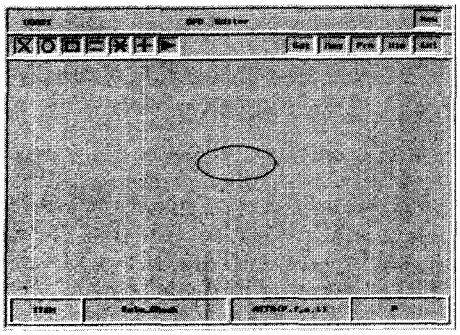


그림 4.9 자료 사전에 항목과 속성을 등록하는 화면

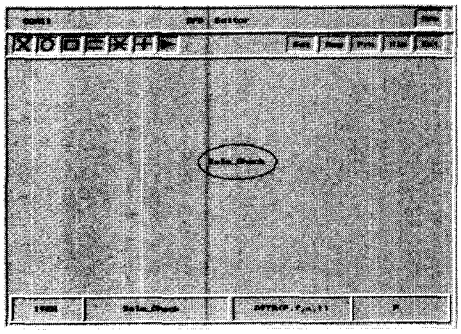


그림 4.10 Sales-Check 프로세스 편집 화면

명세 편집기를 통해서 그림 4.7(a)의 자료 흐름도를 작성한 것은 다음 그림 4.11과 같고, 자료 흐름도 작성을 하면서 자료사전에 등록된 정보 리스트 화일의 리스트는 그림 4.12와 같다.

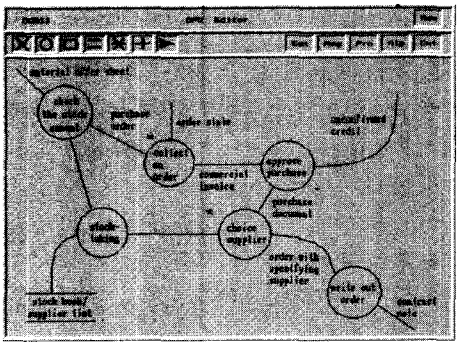


그림 4.11 그림 4.7(a)의 자료 흐름도 편집 화면

```
a:\>type item.dat
waterial offer sheet                    f
check the stock amount                  P
purchase order                          f
order state                             f
collect an order                        P
commercial invoice                      f
approve purchase                        P
unconfirmed credit                      f
purchase document                       f
choice supplier                         P
order with specifying supplier          f
write out order                         P
contract note                           f
stock-taking                            P
stock book/supplier list                s
a:\>
```

그림 4.12 자료사전에 등록된 정보 리스트

4.4 알고리즘

객체 지향 대수적 명세서 시스템의 수행 알고리즘은 다음과 같다.

```
main()
{
   OOASS의 초기 화면 모듈();

   do {
          switch {
                 case 1 : DFD 에디터();
                 case 2 : Petri Net 변환기();
          }
   }while
}

DFE 에디터()
{
   switch {
          case 1 : 편집 항목 선택 모듈_d();
          case 2 : 등록 모듈_d()
          case 3 : 출력 모듈_d();
          case 4 : 도움말 모듈_d();
          case 5 : 새화면 설정 모듈();
          case 6 : exit;
          case 9 : 그림 편집 모듈_d();
   }
}

편집 항목 선택 모듈_d()
{
   switch {
          case 1 : 다이어그램 삽입 모듈();
          case 2 : 다이어그램 삭제 모듈();
```

```
        case 3 : 항목 이동 모듈();
        case 4 : 다이어그램 이동 모듈();
    }
}

그림 편집 모듈_d()
{
    switch {
        case 1 : 자료 흐름선 표시 모듈();
        case 2 : 프로세스 표시 모듈();
        case 3 : 단말기 표시 모듈();
        case 4 : 자료 저장소 표시 모듈();
        case 5 : ANC-조합 논리 기호 표시 모듈();
        case 6 : OR-조합 논리 기호 표시 모듈();
        case 7 : 화살표 표시 모듈();
    }
}

Petri Net 변환기()
{
    switch {
        case 1 : 편집 항목 선택 모듈_p();
        case 2 : 출력 모듈_p();
        case 3 : 도움말 모듈_p();
        case 4 : exit;
        case 5 : 그림 편집 모듈_p();
    }
}

그림 편집 모듈_p()
{
    switch {
        case 1 : 장소 표시 모듈();
        case 2 : 전이 표시 모듈();
        case 3 : 아이크 표시 모듈();
        case 4 : 화살표 표시 모듈();
    }
}
```

5. 결 론

소프트웨어 공학에서 새로운 파라다임으로 제시하는 객체 지향 개념은 소프트웨어 생명 주기의 전단계에 적용되고 있으며, 특히 객체 지향 분석 및 설계에 관한 연구는 매우 활성화되어 있는 실정이다. 그러나 규모가 크고, 복잡하고, 신뢰성있는 소프트웨어 시스템을 개발하는데 핵심적인 역할을 수행하는 객체 지향 규격화에 관한 연구는 단지 이론 정립의 단계이고, 이에 관한 근본적인 문제점들은 아직까지 연구되지 않고 있는 실정이다.

본 논문에서는 객체의 설정에 따르는 객체의 정확한 성질을 규격화하는데 목적을 두고 있다. 즉, 기존의 소프트웨어 규격화 방법론 중에서 데이타 추상화 개념을 가장 효율적으로 반영할 수 있는 대수적 규격화 방법론을 개반으로 하여 객체지향의 핵심적인 개념인 객체 지향 소프트웨어의 규격화 방안을 제시하였다.

본 논문의 내용은 객체 지향 규격화를 수행하는데 관련된 연구로서, 추상화 개념, 소프트웨어 명세화 개념, 객체 지향 개념, 객체 지향 방법론등을 조사하였고, 객체 지향 규격화 기법을 적용한 객체 지향 대수적 명세화 시스템을 구현하였다. 객체 지향 소프트웨어 명세화 시스템은 널리 활용되고 있는 객체 지향 방법론 중의 하나인 OMT 방법론을 대상으로 하였다. 왜냐하면, OMT 방법론을 구성하는 객체 모델과 동적 모델은 구문 및 시맨틱스가 정형적인데 반해서, 기능적 모델을 표현하는 자료 흐름도의 시맨틱스는 정형적인 면이 부족하다는 단점을 갖고 있기 때문이다.

객체 지향 규격화 방안은 크게 두 단계로 구성하였다. 첫번째 단계에서는 자료 흐름도를 수학적 개념을 내포하고 있는 페트리 네트로 변환하였다. 본 논문에서는 이러한 변환을 수행하는데 대두되는 자료 흐름의 모호성 문제점을 해결하기 위해 조합 논리 기호 개념을 사용하였고, 조합 논리 기호가 표시된 자료 흐름도를 페트리 네트로 변환하는 규칙을 제시하였다. 두번째 단계에서는 페트리 네트의 대수적 개념을 기술하기 위한 객체 지향 대수적 명세서의 템플리트를 정의하였다. 템플리트는 상속성, 정보은닉등의 객체 지향 개념을 반영할 수 있도록 하기 위해 parameters, exports, imports, functions, vari-

ables, 그리고 equations 부분으로 구성하였다.

객체 지향 규격화 연구를 수행함으로써 기대되는 효과는 다음과 같다. 첫째, 객체와 클래스 개념에 의해 이미 개발된 소프트웨어를 규격화함으로써 소프트웨어의 재사용성을 향상시킬 수 있고, 둘째, 객체 지향 개념을 적용함으로써 사용자와 개발자간의 인터페이스를 향상시켜 사용자의 빈번한 요구사항 변경에 신속히 대처할 수 있다. 세째, 신기술인 객체 지향 기법을 통신용 소프트웨어에 적용함으로써 외국기관(ITU.T)과의 이해의 범용성을 추구할 수 있다.

본 논문의 활용방안으로는 제시된 객체 지향 규격화 기법을 객체 지향 SDL'92 환경을 구축하는데 기초 자료로 활용하고, 또한 기존의 통신용 소프트웨어를 객체 지향 소프트웨어로 변환하는 CASE 시스템 구축의 기초 자료로 활용되어져야 할 것이다.

향후 연구방향으로는 설계된 명세 편집기의 기능을 확장한 명세 분석기 및 페트리 네트 변환기의 기능을 확장한 페트리 네트 분석기에 관한 연구가 수행되어져야 한다.

참 고 문 헌

[1] G.Booch, "Object-Oriented Development", IEEE Trans. on Software Eng., Vol. SE-12, No.2, 1988.

[2] G. Booch, Object-Oriented Design with Applications, Benjamin / Cummings, 1991.

[3] 김수동, "객체 지향 소프트웨어 구성을 위한 Formal Specification의 연구".

[4] J.A. Bergstra, J. Heering and P. Klint, "Algebraic Specification", ACM Press, 1989.

[5] J. V. Guttag, "Notes on Type Abstraction", IEEE Trans. on Software Eng., Vol. SE-6, No. 1, Jan., 1980, pp.13-23.

[6] B. Liskov and J. Gutag, Abstract and specification in Programming Development, Mcgraw-Hill, 1986.

[7] 이준석, 지동해, "객체지향 개발방법론 고찰", 주간 기술동향, 한국전자통신연구소, 1992.

[8] J. Gannon, et al, "Data Abstraction Implementation, Specification and Testing", ACM TOPLAS, Vol. 3, Jul., 1981, pp. 211-223.

[9] C. Ghezzi, M. Jazayeri and D. Mandrioli, Fundamentals of Software Engineering, Prentice-Hall International Editions, 1991.

[10] C. V. Ramamoorthy, A. Praksh, W. T. Tasi and Y. Usuda. "Software engineering : Problems and Perspectives", IEEE Computer, Oct., 1984, pp. 191-209.

[11] V. S. Alagar and K. Periyasamy, "A Methodology for deriving an object-oriented design from functional specification", Software Engineering Journal, July, 1992, pp. 247-263.

[12] N. Gehini, "Specifications : Formal and Informal - A Case Study", Practice and Experience, Vol. 12, 1982, pp. 433-444.

[13] B. H. Liskov and V. Berzins, "An Appraisal of Program Specifications", in Software Specification Techniques, N. Gehani and A. Mcgettrick(eds.), Adison-Wesley, 1986, pp. 13-23.

[14] D. L. Parnas, "A Technique for Software Module Specification with Examples", in Software Specification Techniques, N. GehaniA. Mcgettrick(eds.), Addison-Wesley, 1986, pp. 75-88.

[15] B. Liskov and S. Zilles, "Specification Techniques for Data Abstractions", IEEE Trans. on SE, Vol. SE-1, No. 1, March, 1975, pp.7-19.

[16] Rumbaugh, Js., et al, "Object-Oriented Modeling and Design", Prentic Hall, 1991.

[17] M. Fowler, "A Comparison of Object-Oriented Analysis and Design Method", July, December, 1992.

[18] P. Coad and E. Yourdon, Object-Oriented Analysis, Second Edition, Prentice-Hall International Inc., 1991.

[19] R. S. Pressman, Software Engineering, Third Edition, McGraw-Hill International Editions, 1992.

[20] S. Shlaer and S. J. Mellor, Object-Oriented Systems Analysis : Modeling the World in Data, Prentice-Hall, 1989.

[21] R. Braek and φystein Haugen, Engineering Real Time Systems With An Objcet-Oriented Methodo-logy based on SDL, SISU, Oslo, Trondheim, june, 1992.

[22] K. R. Kim, H. W. Yun, and Y. S. Koo, "Construction a Formal Semantics of DFD Using and algebraic specification Technique", ICCTA'94, Aug, 1994.

[23] S. D. Kim, Formal Specification in Object-Oriented Software Development, Ph. D. Dissertation, IOWA, May, 1991.

[24] Borland C++ for Windows(Library Reference), Borland International, Inc., 1993.

[25] A. V. Aho, R. Sethi, and J. D. Ullman, Principles of Compiler Design.

[26] CCITT Specification and Descrip-tion Language(SDL), COM X-R 17-E, rev. 25.5., 1992.

□ 著者紹介

김 영 란

충북대학교 전산통계학과 졸업(이학사)
충북대학교 대학원 전자계산학과 졸업(이학석사)
충북대학교 대학원 전자계산학과 박사과정 수료
현재 한국정보과학회 종신회원
충청전문대학 사무자동화과 전임강사