「經營情報硏究」 第1號
大韓經營情報學會
1997年 6月

객체모델링기법에 의한 객체지향 모델베이스 설계

정 대 율*

〈目 次〉

Ⅰ. 연구동기

오늘날 기업 내에서 의사결정모델이 안정적으로 사용되고, 또한 공유가능한 자원이라는 인식이 점차 확산되자 기업 전체의 정보관리라는 측면에서 총체적 모델링자원의 통제 중요성이 강조되고 있다. 효과적인 모델링자원의 통제를 위해서는 체계적인 모델베이스 개발에 의한 모델관리가 필요하다. 모델링 환경에서도 데이터 자원의 관리문제와 유사한 상황이 발생한다. 즉, 모델의 ①중복성, ②불일치성과 무결성(integrity) 결여, ③공유의

* 동명전문대학 경영정보과

결여, ④표준화의 미흡, ⑤모델독립성의 결여 등과 같은 문제가 발생할 수 있다. 이러한 문제를 효과적으로 처리하기 위해서는 체계적인 모델베이스 개발이 필요하다(D. R. Dolk 1986, pp.73-80).

지금까지 모델베이스에 대한 대부분의 연구들은 단일 유형의 논리적 모델스키마 표현에 대한 연구에 초점을 두어 왔으며, 조직 전체적인 입장에서 모델베이스를 조직화하는 방법에 관한 연구는 부족하였다(Muhanna 1992, p. 359). 즉, 단일 유형의 모델 해를 구하기 위해 이 모델을 특정 모델링언어로 표현하는 데 연구의 초점을 두어왔다. 또한 조직 내에서는 각 부서마다 각각 독자적인 모델링 패러디임을 지원하는 다양한 형태의 모델링 언어가 사용되어져 왔다. 이로 인하여 개발된 모델이 분산관리됨은 물론이고 모델간의 호환성 결여와 모델의 중복개발로 인한 개발비용의 낭비를 가져왔다. 따라서 조직내의 각 부문마다 각자 특특한 모델링 패러디임을 적용하여 이미 개발되어온 여러 유형의 모델에 대하여 동일한 모델관리 매카니즘을 적용할 수 있는 방법에 대한 연구가 필요하다.(허순영 1994, p. 43).

이 문제를 해결하기 위해서는 다양한 모델링 언어로 개발된 모델들을 통합하여 하나의 모델베이스에 저장하여 관리할 필요가 있다. 이 때 통합된 모델베이스는 물리적 측면에서 통합된 모델베이스가 아니라 조직내의 여러 부서에 흩어져 있는 모델들을 개념적, 논리적 수준에서 통합하여 하나의 관리메커니즘을 적용할 수 있게 함을 의미한다.

따라서 본 연구에서는 객체지향 모델베이스 개념을 도입하여, 모델베이스를 구성하는 구성요소들을 타입화하고 이들 모델구성요소들 간의 참조적 무결성을 확보할 수 있는 모델베이스를 설계하고자 한다. 이를 위하여 본 연구에서는 모델베이스 설계에 있어 메타-모델링(meta-modeling) 개념을 도입하였다. 메타수준의 모델베이스(메타-모델베이스라 부름) 설계는 여러 모델링 패러다임에서 개발된 다양한 형태의 모델들을 하나의 통일된 틀 속에 넣을 수 있게 해줄 뿐만 아니라 DSS(Decision Support Systems)를 구성하는 이질적인 구성요소(의사결정모델, 데이터, 해산출 앨고리즘, 모델링 지식 등)들을 체계적으로 조직화할 수 있게 한다. 이것은 DSS의 모델베이스를 구성하는 여러 가지의 객체들에 대하여 단일화된 관리방법으로 그 접근이 가능하게 해준다.

이러한 목적을 달성하기 위해서 본 연구에서 기본적으로 도입된 기법은 Rumbaugh 등(1991)이 제시한 객체모델링기법(Object Modeling Technique : OMT)이다. DSS의 모델베이스 설계에 있어 객체모델링기법의 도입은 모델베이스 설계를 위한 다양한 표현법을 제공해준다.

Ⅱ. 선행연구

모델관리시스템(Model Management Systems : MMS)이란 모델베이스 내의 모델을 효과적으로 관리하는 기능 즉, 모델의 개발 및 저장, 조작 및 접근, 통제하는 것을 체계적으로 관리하는 소프트웨어의 집합체이다(Chung & O'Keefe 1992, p.137). 모델관리시스템이란 용어를 최초로 사용한 사람은 Will(1975)로 알려져 있다. 그 이후 약 20여년 동안 경영과학 및 정보시스템 분야에서 모델관리시스템에 대한 연구가 이루어져 와 이제는 의사결정지원시스템 분야의 중요한 연구영역으로 자리를 차지하고 있다.

Chung & O'Keefe(1992)는 모델관리에 대한 기본적인 접근법으로 ①모델/모델링의 인식 접근법 ②데이터-모델 유추 접근법 ③지식중심의 접근법 ④모델표현 접근법 ⑤기타 접근법으로 분류하고 있으며, Bharadwaj 등(1992)은 ①대수적 모델링 언어 ②데이터베이스 접근법 ③그래프중심의 접근법 ④지식중심 접근법으로 분류하고 있다. 또한 Blanning(1993)은 ①모델베이스구조(네트웍 또는 관계형 데이터베이스구조와 대비됨) ②모델베이스처리와 인공지능응용 ③MMS의 조직적 환경과 기여라는 측면에서 모델관리의 주요 주제를 분류하고 있다.

〈표 2-1〉 모델베이스 개발을 위한 주요 연구

프레임웍		연 구 자	년도	모델링언어 (모델관리시스템)
대수적 모델링언어		Harverly Systems	1976	OMNI
		Bisschop & Meeraus	1982	GAMS
		Schrage	1987	LINDO
		Fourer, et.al.	1990	AMPL
		MathPro, Inc.	1990	MathPro
데이터베이스 지향	실체-관계	Blanning	1986	E-RD
		Chen, Y.S.	1988	EE-RD
		Choobineth	1991b	ERLMD
	네트웍형	Dolk	1986a	GXMP
	관계형	Blanning	1985, 1987	MQL, ETQL
		Choobineth	1991a	SQLMP

프레임웍		연 구 자	년도	모델링언어 (모델관리시스템)
지식 베이스 지향	논리중심	Bonczek et.al.	1981	(GDSS)
		Dutta & Basu	1984	PCL
		Sivasankara & Jarke	1985	(ACS)
		Murphy & Stohr	1986	nn
		Liang	1988	(TIMMS)
		Krishnan	1989,1991	PM*(PDM)
		Ma et. al.	1989	(LPFROM)
		Liu et. al.	1990	(AIMM)
		Bhargava & Kimbrough	1993	L↑
	프레임중심	Elam et. al.	1980	SI-Net
		Dolk & Konsynski	1984	Model Abstraction
		Binbasioglu & Jarke	1986	(LP Formulator)
		Applecate et. al.	1986	(SPMMS)
		Fedorowicz & Williams	1986	(IDSS)
		Pracht	1990	ModelVisualization
그래프 중심		Liang	1986	Model-graph
		Jones	1990,1991	(GBMS)
		Glover et. al.	1990	NETFORM
		Geoffrion	1992	SML
		Collaud & Boltuck	1994	(gLPS)
객체지향 / 시스템		Lazimy	1991	EERA
		Dempster & Ireland	1991	nn
		Potter, et.al.	1992	KDM/KDL
		Hong et.. al.	1993	UML(Lu)
		Huh	1993	nn
		Muhanna	1993,1994	MDL(SYMMS)

* nn : no named

모델베이스 개발에 대한 기존의 연구들을 분류하기 위해서는 위에서 본 바와 같이 여러 가지의 기준이 제시될 수 있으나, 본 연구에서는 Bharadwaj 등의 분류기준에 기초하여 모델베이스 개발을 위한 프레임웍을 다음과 같이 분류한다.

① 수리적 모델(특히 LP모델)을 대수식으로 보고, 이들 대수식을 해산출기(solver)로 풀어 모델의 해를 구하려는 대수적 모델링언어 프레임웍.

② 모델을 데이터로 보고 데이터베이스기술과 통합하려는 실체-관계모델(E-R모델) 프레임웍과 관계모델 프레임웍.

③ 모델을 지식의 집합체로 보고 지식표현의 기법을 모델관리에 도입하려는 논리중심 프레임웍과 프레임중심 프레임웍.

④ 수리적 모델을 그래프로 이용하여 표현하고자 하는 그래프중심 프레임웍.

⑤ 모델을 하나의 객체로 보고 객체지향 개념을 도입한 객체지향 프레임웍.

이들 프레임웍별로 대표적인 연구와 모델표현을 위해 사용된 모델링 언어 또는 언어실행을 위한 모델관리시스템을 요약하면 (표 2-1)과 같다. 그리고 이들 모델베이스 개발을 위해 제시된 연구 프레임웍들의 주요 장·단점을 요약하면 (표 2-2)와 같다. 이들 연구 프레임웍들 중 어느 것이 가장 이상적이라 할 수는 없으나, 각각의 장·단점이 있기 때문에 서로 보완적이라 할 수 있을 것이다. 그런데 최근의 관심은 객체지향 모델관리 프레임웍의 개발에 쏠리고 있다. 그 이유는 객체지향 개념의 도입은 지금까지 연구된 각종 프레임웍의 장점을 최대한 수용할 수 있기 때문이다.

본 연구는 모델베이스 설계에 있어 객체모델링기법을 사용하므로 객체지향 프레임웍의 분류에 속한다. 객체모델링기법을 모델베이스 설계에 사용할 경우 일반화(generalization), 집합(aggregation), 타입화(typing) 등과 같은 객체지향의 여러 개념들을 적용할 수 있다.

〈표 2-2〉 모델관리에서 주요 프레임웍의 장단점

프레임웍	장 점	단 점
대수적 모델링 언어	* 고도의 추상적 모델표현에 의한 조작 용이성.	* 주로 LP문제에 국한되어 일반성이결여됨. * 모델독립성에 대한 고려가 약함. * 개념적 모듈화를 지원하지 못함.
관계모델	* 데이터와 모델베이스의 통합능력 향상. * 관계형 DBMS가 제공하는 뛰어난 관리와 통제능력 이용. * 관계질의 언어의 사용. * 과거의 DBMS 경험을 MMS에 적용 * 다양한 데이터원천의 제공 * 응용영역과 표현법에 독립적임.	* 모델은 데이터보다 복잡하고 동적인 구조를 가지므로 모델의 저장과 관리에 데이터관리기법의 적용이 어려운 부분도 있음. * 의사결정처리에는 부적합. * 모델에 대한 다중 관점 결여
E-R모델	* 모델실체들 간의 관계표현 가능. * 모델간의 입출력관계표현. * DB와 모델베이스의 개념적 통합. * 사용자와의 효과적인 의사소통.	* 모델의 논리적 수준의 스키마표현 법이 없음. * 모델의 동적 구조를 표현할 수 없음
논리중심	* 강력한 탐색 및 선택기능. * AI문제에서 성공적으로 수행	* 대량의 데이터/모델취급이 곤란. * 문제가 크질 경우 탐색공간의 증대 * 개념관계의 상실.
프레임중심	* 복잡한 의사결정이나 모델 영역의 표현이 용이. * 모델특성들의 다중표현이 가능. * 프레임 내에서 문제/모델특성을 나타내는 표현들을 저장하기 위해 술어연산공식논리의 사용 가능	* 문제상황에 비추어 프레임이 먼저 정의되어야 함. * 설계시 정의되지 않은 유용한 대체적인 해의 선택에 제약을 받음.
그래프중심	* 강력한 분류기능. * 문제-모델간의 개념적 관계표현용이 * 그래프문법의 사용 가능	* 여러 수준의 논리에 대한 직접적인 지원결여. * 복잡한 모델이나 의사결정영역의 취급곤란.
객체지향	* 계승개념에 의한 모델의 재사용성 강화. * 모델 및 모델구성요소의 타입화와 집합개념을 이용한 모델합성의 용이 * 의미론적 충분성을 기할 수 있는 모델링 구성자의 제공	* 고도의 추상적인 모델표현 결여 * SQL과 같은 논리적 근거에 의한 질의처리기능 미흡.

Ⅲ. 객체모델링기법과 모델관리

1. 객체모델

객체(object)는 한 실체(entity)의 데이터구조와 행위(data structure and behavior)를 동시에 결합한 것으로 데이터의 특성과 절차(procedure)의 특성을 동시에 결합한 실체로 정의할 수 있다. 객체지향접근법에서 요구되는 객체의 주요 특성으로는 정체식별성(identity), 분류(classification), 多形性(polymorphism), 그리고 계승(inheritance) 등을 들 수 있다.

이러한 객체의 특성을 나타내는 모델이 객체모델(object model)이며 객체지향 접근법의 요체가 된다. 객체모델은 시스템내의 객체구조, 즉 객체의 정체식별성(identity), 타 객체와의 관계, 객체의 속성, 그리고 객체의 오퍼레이션 등을 記述한다. 객체모델을 구축하는 목적은 문제해결에 중요한 현실세계의 개념들을 포착하는데 있다. 객체모델은 객체 클래스를 포함한 객체다이어그램(object diagram)으로 표현되어진다. 객체다이어그램에서 클래스는 공통구조와 행위를 공유한 계층(hierarchies)으로 정렬되며, 다른 클래스와 연관되어진다.

모델표현을 위한 노력과 수준, 모델의 취급가능성과 이해의 용이성은 현실문제를 얼마나 추상화하느냐에 따라 결정된다. 추상화(abstraction)는 복잡성을 극복하는 가장 기본적인 방법의 하나로서 실체의 본질적이고 고유한 측면에 중점을 두어 문제에 직결되는 요소들만을 주요 고려대상으로 하는 것을 말한다.

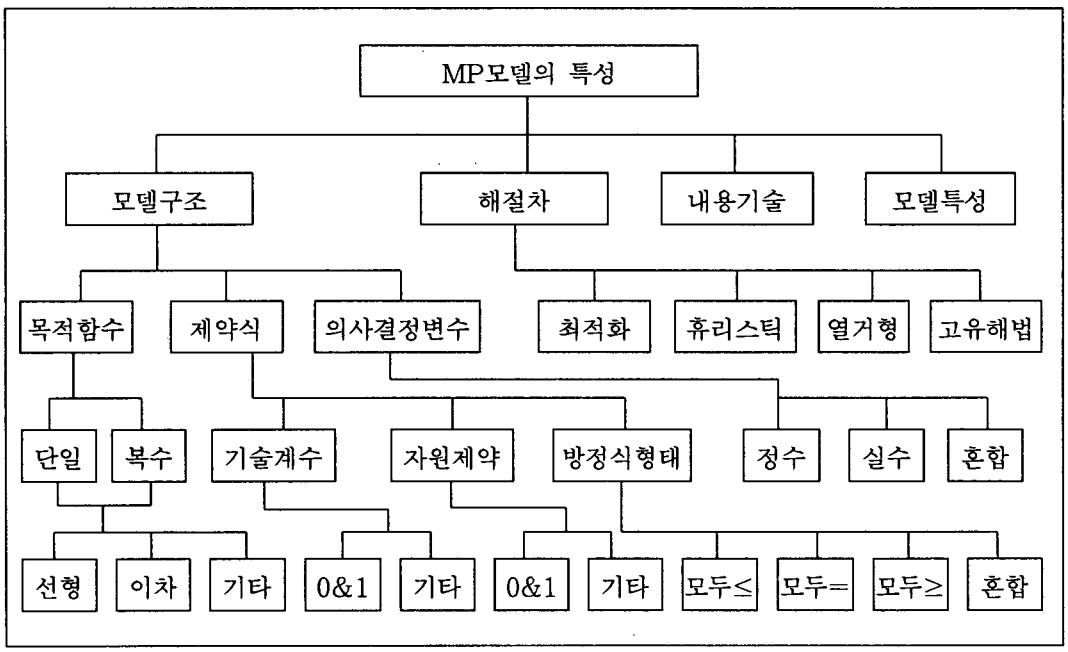
컴퓨터 기반의 모델링환경 측면에서 볼 때, 모델표현을 위해서는 두 가지 측면의 추상화를 들 수 있다. 첫 번째는 개념적 모델링과 논리적 모델링이다. 의사결정자의 의사결정 요구를 정확히 반영하여 문제를 모델화하기 위해서는 컴퓨터 상의 구현문제와는 독립적으로 개념적 수준의 모델링이 필요하며, 이러한 개념적 모델을 컴퓨터 상에서 실행 가능하도록 하는 논리적 수준의 모델링이 요구된다.

두 번째는 모델베이스 내의 모델클래스들 간의 추상화이다. 모델클래스는 모델링 패러디임에서 사용하는 가정에 따라 그 추상화 수준에 차이가 나며, 문제영역의 수준에 따라서도 그 추상화 정도를 달리할 수 있다. 객체모델은 이들 모델클래스들 간의 추상화 관계를 용이하게 식별할 수 있게 한다.

객체모델링을 위한 기본 개념들로는 분류(classification)와 클래스(class), 연관관계(association), 집합(aggregation), 일반화(generalization) 등이 있다. 본 연구에서는 이러한 개념들이 모델베이스 추상화에 어떻게 사용될 수 있는가를 보이고자 한다.

2. 분류와 클래스

분류(classification)는 지식을 체계화하는 수단이다. 객체지향적 개념에서 객체간의 동질성을 인식하는 것은 추상화를 통한 단순한 모델링을 가능하게 한다. 그러나 분류의 왕도는 없다. 즉, "완전한" 클래스구조와 "올바른" 객체집합과 같은 것을 있을 수 없다. 따라서 창의적인 활동을 통해 객체들을 모으는 방법을 규정하는 메커니즘 뿐만 아니라 일반화된 추상화를 고안해야만 한다. 일반적으로 분류시 공통된 구조를 지니고 공통된 행위를 보여주는 것끼리 그룹화하게 된다. 예를 들어 수리계획(MP)모델을 분류화하는 데 (그림 3-1)과 같이 네 가지 특성(①모델구조 ②해절차 ③내용기술 ④모델특성)에 따라 분류할 수 있다.



〈그림 3-1〉 수리계획(MP)모델의 분류

좋은 분류는 일반화(generalization)와 구체화(specialization), 그리고 클래스간의 집합계층(aggregation hierarchies)을 식별하는 데 도움을 준다. 객체간의 공통된 상호작용 패턴을 인식하므로서 실행을 용이하게 하는 메커니즘을 발견할 수 있게 한다. 또한 분류는 모듈화(modularization)에 관한 의사결정을 용이하게 한다.

객체클래스(object class)는 유사한 특성(속성), 공통된 행위(오퍼레이션), 他 객체에 대한 공통된 관계, 공통된 의미론(semantics)을 지닌 객체들의 그룹을 記述한다. 사람, 회사, 동물, 프로세스(process) 등은 모두 객체클래스의 예이다. 객체와 객체클래스는 종종 문제기술시 명사의 형태를 지닌다.

객체다이어그램(object diagrams)은 객체클래스와 이들 간의 관계를 모델링하기 위한 공식적인 그래프적 표현법을 제공해준다. 따라서 객체다이어그램은 추상적인 모델링과 실제 프로그램의 설계에 유용하다.

한 클래스내의 객체에 적용되거나 객체에 의해 적용될 수 있는 기능이나 변환을 오퍼레이션(operation)이라 한다. 예를 들어 의사결정변수 삽입, 파라메트 삽입, 해산출기 연결과 같은 것은 모델클래스에서의 오퍼레이션이다. 한 클래스에 대한 오퍼레이션의 구체적인 실행을 메소드(method)라 한다.

객체모델링기법에서 객체클래스는 3단 박스로 표기하며, 제 1단에는 클래스명, 제 2단에는 클래스속성, 제 3단에는 클래스 오퍼레이션을 기입한다. 그리고 객체인스턴스는 둥근 박스로 표현한다.

3. 연관관계와 집합

연관관계(association)는 객체와 클래스간의 관계를 확립하는 가장 기본적인 방법이다. 링크(link)는 객체인스턴스들 간의 물리적 연결이나 개념적 연결을 나타낸다. 예를 들어 "Joe Smith work-for Simplex company"에서 "work-for"는 링크를 나타낸다. 수학적으로 볼 때 링크는 튜플(tuple), 즉 객체인스턴스의 순서화된 리스트로 정의되어진다. 따라서 링크는 연관관계의 인스턴스이다.

연관관계는 두 객체간의 관계인 이진(binary), 세 객체간의 관계인 삼원(ternary), 또는 더 이상의 고차(higher order)관계를 가질 수 있다. 실제적으로 대부분의 경우 이진관계이거나 한정관계(이진관계의 특수한 형태)이며, 삼원 또는 그 이상의 관계는 흔치 않다.

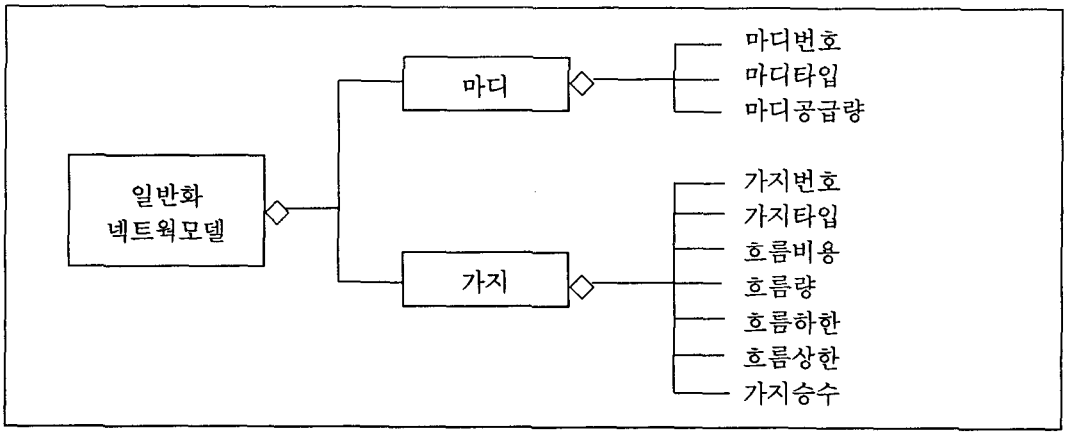
삼원연관관계는 정보를 유실하지 않고는 더 이상 이진관계로 나눌 수 없는 원소단위이다. OMT에서 연관관계는 관련된 객체들 간의 연결선으로 표시하며, 이중 삼원 또는 多元(n-ary)연관관계는 관련된 클래스를 연결하는 선을 가진 다이어몬드형으로 표시한다.

연관관계에서 다중성(multiplicity)은 한 클래스의 어떤 인스턴스에 이와 연관된 클래스의 인스턴스가 얼마나 많이 관련되어 있는지를 나타내는 것이다. 따라서 다중성은 관련된 객체의 수를 제약한다. 연관관계와 링크의 대응관계는 "one-to-one", "one-to-many", "many-to-many" 등이 있다. 일반적으로 다중성 값(multiplicity value)은 "3-5"와 같이 단일구간이거나 "2,4,8"과 같이 불연속구간의 값을 취한다. 객체다이어그램에서 다중성의 표시는 연관화 선의 끝부분에 검은 점이나 구멍이 뚫린 점으로 표시한다. 검은 점은 "many"를 표시하며, 구멍이 뚫린 점은 "zero" 또는 "one"을 나타내는 선택을 표시한다.

집합(aggregation)은 연관관계의 확장된 형태로써 객체들 간의 "a-part-of" 또는 조립-구성요소관계를 나타낸다. 따라서 복합객체를 나타내는 데 유용하다. 집합의 가장 중요한 특성은 移行性(transitivity)이다. 移行性이란 만일 A가 B의 일부분이고, B가 C의 일부분일 경우, A는 C의 일부분을 나타낸다. 또한 집합은 반대칭적(antisymmetric)이다. 즉 A가 B의 일부분이면 B는 A의 일부분이 아니다. 객체다이어그램에서 집합은 작은 다이어몬드(◇)로 나타내며, 동일한 조립품에 속하는 모든 구성품의 집합은 다이어그램상에서 하나의 집합트리(aggregation tree)로 결합할 수 있다.

모델관리에서 연관관계와 집합개념은 원소모델(atomic model)의 구성요소들과의 관계를 보여줄 수 있을 뿐만 아니라 원소모델을 결합한 복합모델(composite model)의 구조를 보여주는 데도 유용하게 사용될 수 있다. 예를 들어 (그림 3-2)는 일반화 네트웍모델(generalized network model)의 구성관계를 보여 주고 있다. 일반화 네트웍모델은 마디(node)와 가지(arc)로 구성되며, 마디는 마디번호, 마디타입(node type), 마디공급량(마디수요량) 등으로 구성되며, 가지는 가지번호, 가지타입(arc type), 흐름비용, 흐름량, 흐름하한, 흐름상한, 가지승수(arc multiplier) 등으로 구성된다.

집합개념이 모델베이스 설계에 적용될 경우 모델을 구성하는 구성요소를 쉽게 식별하게 하며, 원소모델을 결합하여 복합모델을 만드는 경우 그 관계를 쉽게 표현할 수 있다. 즉, 모델의 계층적 조직화를 촉진시켜, 모델구성요소와 원소모델의 재사용을 촉진시키며, 모델베이스 내의 중복성을 배제하여 모델간의 무결성을 자져오게 한다.



〈그림 3-2〉 일반화 네트웍의 구성

4. 일반화와 계승

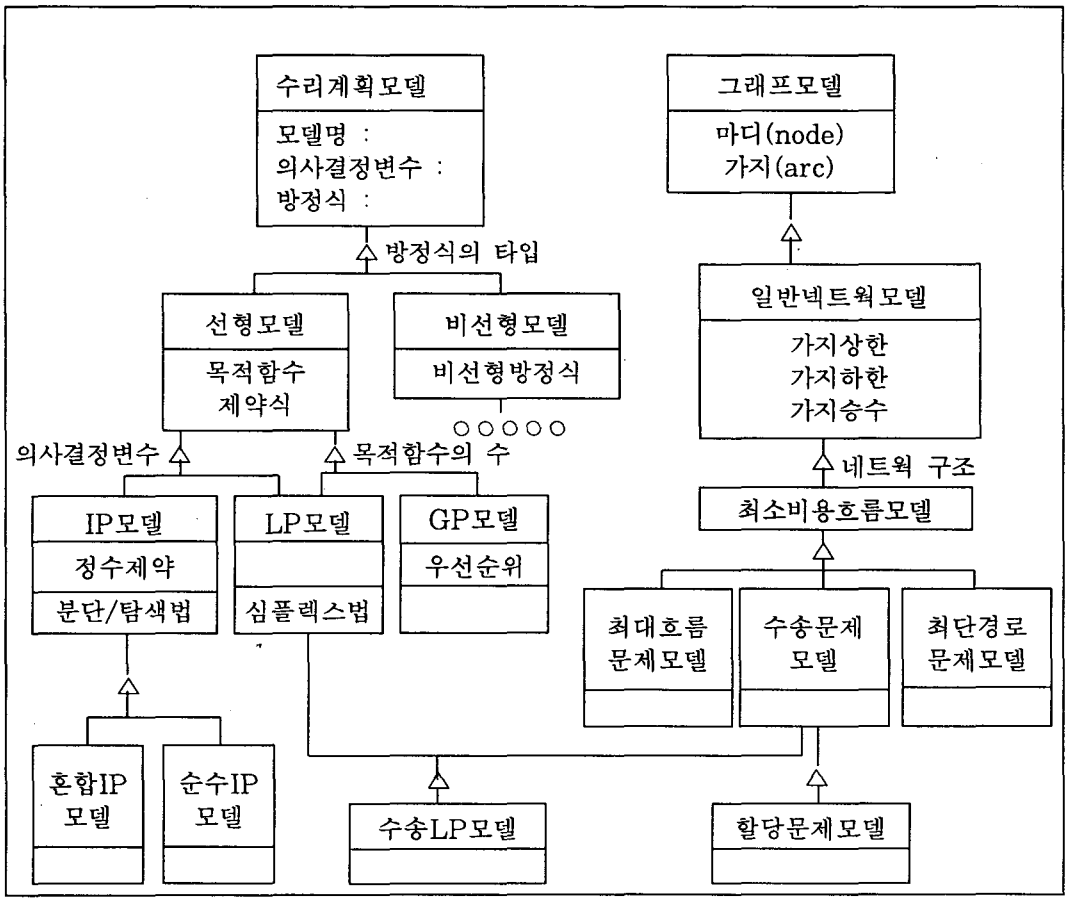
일반화(Generalization)와 계승(inheritance)은 클래스간의 차이를 유지하면서 공통성을 공유할 수 있게 하는 아주 좋은 추상화방법이다. 일반화는 어떤 한 클래스와 이 클래스로부터 특수화된 클래스간의 관계를 나타낸다. 이 때 특수화의 대상이 되는 클래스를 상위클래스(superclass)라 하고 특수화된 각 변형을 하위클래스(subclass)라 한다. 예를 들어 LP모델은 의사결정변수, 선형목적함수, 선형제약식, 파라메타를 갖는 객체이다. LP모델의 특수형인 수송LP모델은 LP모델의 기본적인 특성을 그대로 가지면서 특수한 파라메트구조(제약구조)를 지니며, 이로 인해 특수한 해법(수송심플렉스해법)이 존재하는 모델이다. 따라서 LP모델은 상위클래스이며, 수송LP모델은 하위클래스를 이룬다.

하위클래스 그룹에 공통된 속성과 오퍼레이션은 상위클래스에 귀속되고 각 하위클래스에 의해 공유된다. 각 하위클래스가 상위클래스로부터 특성을 이어받는 것을 계승(inheritance)이라 한다. 예를 들어 수송LP모델클래스는 LP모델클래스의 특수한 형태로서 LP모델의 일반적 속성을 이어 받는다. 일반화 관계를 종종 "is-a" 관계라 부른다. 일반화와 계승은 각 계층수준간에 이행적(transitive)인 성격을 지닌다. 따라서 각 하위클래스는 상위클래스의 모든 특성을 이어받을 뿐만 아니라 그 하위클래스 고유의 속성과 오퍼레이션을 가질 수 있다. OMT에서 일반화를 나타내는 표기법은 상위클래스와 하위클래스를 연결하는 부분에 삼각형(△)으로 표시하며, 상위클래스가 삼각형의 꼭지점에 연결된다. 일반화를 나

타내는 심벌인 삼각형 옆의 글은 구별자(discriminator)를 나타낸다. 구별자는 특정 일반화관계에 의해 추상화되는 객체특성을 가리키는 열거형태의 속성이다

다중계승(multiple inheritance)은 한 클래스가 하나 이상의 상위클래스를 가지며, 모든 상위클래스로부터 특성을 이어받는 경우를 말한다. 따라서 다중계승은 클래스계층을 트리형태로 한정하는 단순계승보다는 보다 더 복잡한 형태의 일반화이다. 다중계승의 장점은 클래스 명세를 보다 용이하게 하고 코드 재사용의 기회를 증가시켜준다. 그러나 다중계승의 단점은 개념적 단순성과 실행적 단순성을 상실하는 것이다.

수리계획모델과 그래프모델의 일반화와 계승관계를 예시하면 (그림 3-3)과 같다. 여기서 수리계획모델은 방정식의 타입에 따라 목적함수와 제약식의 방정식이 모두 일차선형방정식인 경우에는 선형계획모델이 되며, 그렇지 않은 경우에는 비선형계획모델이 된다. 또



〈그림 3-3〉 수리계획모델의 일반화와 상속의 예

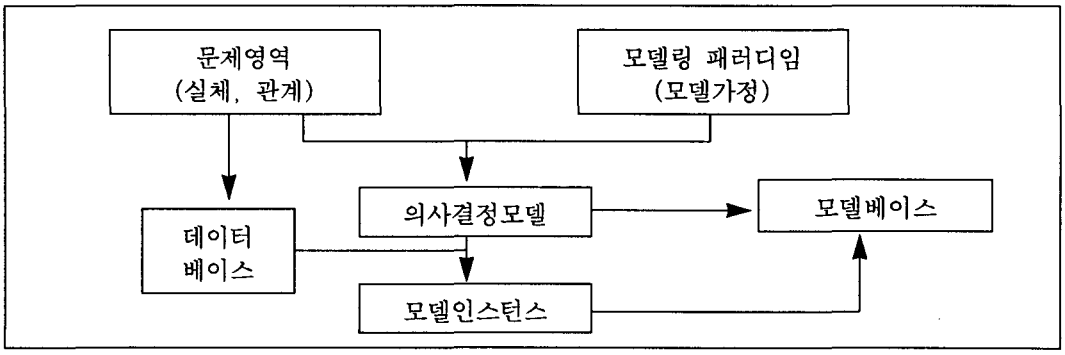
한 네트웍모델들은 그래프모델의 속성을 이어받으며 그 문제구조에따라 다시 세분화할 수 있다. 그리고 수송LP모델은 LP모델의 특수한 형태이며, 네트웍문제의 구조를 가지므로 양쪽의 성질을 다 이어받으므로 다중계승관계를 지닌다. 그리고 최상위의 수리계획모델과 그 다음의 선형모델, 그리고 그래프모델 클래스는 추상클래스(abstract class)이며, LP모델, IP모델, GP모델, 최소비용흐름 네트웍 모델, 수송 네트웍 모델 수송LP모델 등은 구체적인 인스턴스를 가질 수 있으므로 구체클래스(concrete class)이다.

일반화는 개념적 모델링과 실행단계 모두에 유용한 방법이다. 일반화는 클래스를 구조화하고 계속적으로 그 클래스에 대해 유사성을 가진 것을 포착하므로써 모델링을 용이하게 한다. 오퍼레이션의 계승은 실행시 재사용코드를 위한 수단으로 유용하다. 일반화 개념이 모델베이스 설계에 적용될 경우 모델타입 간의 계승관계를 통하여 신속한 모델링을 가능하게 한다.

Ⅳ. 객체지향 모델베이스

1. 모델베이스와 모델클래스

모델링은 문제영역(시스템)에 존재하는 객체와 이들 간의 관계를 특정 모델링 패러디임(예, 선형계획법, 정수계획법, 동적계획법, 대기행렬, 시뮬레이션 등)이 가지는 假定에따라 의사결정모델을 수립하는 것을 말한다(그림 4-1). 따라서 의사결정모델이란 현실세계에 존재하는 시스템을 공식적 언어로 표현한 추상화된 객체이다. 모델화의 대상이



〈그림 4-1〉 모델링의 세계

되는 그 시스템은 다시 여러 하위시스템(subsystem) 또는 구성요소들로 구성되며 이들은 상호 작용한다. 또한 시스템은 외부환경과의 인터페이스를 통하여 다른 시스템과 상호 작용한다. 따라서 모델을 시스템 관점에서 본다면 모델은 외부와의 인터페이스를 나타내는 객체와 모델 구성요소들 간의 상호작용을 나타내는 객체들로 구성된다고 볼 수 있다.

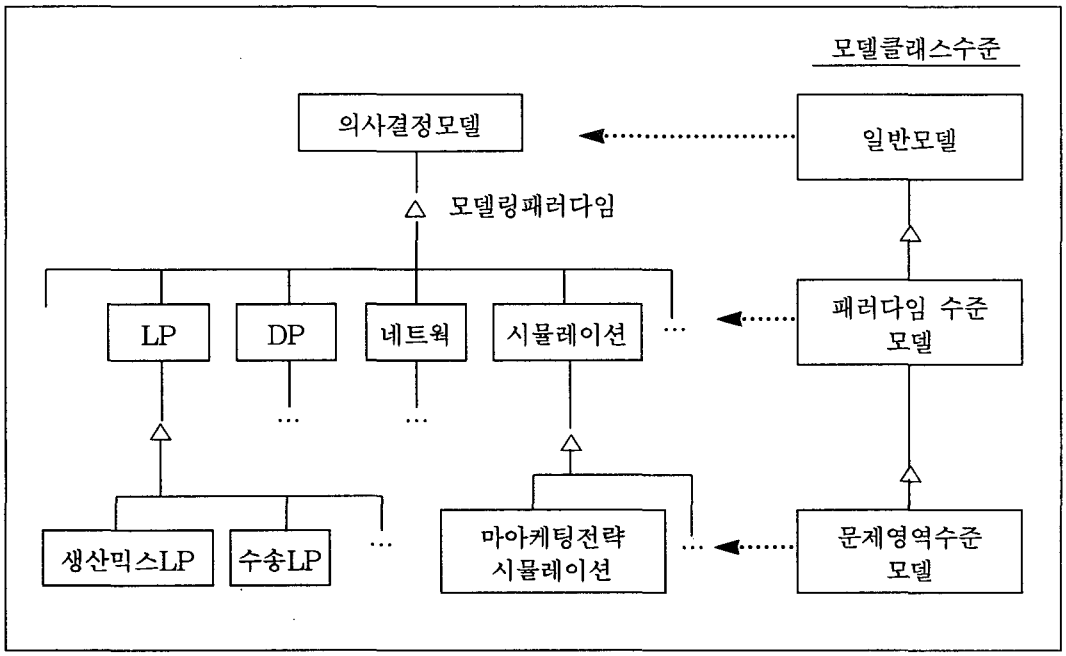
모델베이스는 현실세계에 존재하는 다양한 실체와 이들 간의 관계를 특정 모델링 패러디임에 따라 구조화하여 특정 모델링언어[GAMS(Bisschop & Meeraus 1982), AMPL(Fourer et. al. 1983), LINDO(Schrage 1987), SML(Geoffrion1992) 등]로 표현한 모델들의 집합체이다. 모델베이스 내의 모델들은 그 구조와 행위적 특성에 따라 서로 유사한 것끼리 묶을 수 있다. 이를 모델클래스(model class)라 한다. 따라서 모델클래스는 유사한 구조를 가진 모델들의 집합이며, 이것은 타입화될 수 있다. 모델타입(model type)은 모델 클래스의 특성을 명세하는 객체타입이다. 모델타입은 모델의 구조, 기능, 그리고 다른 모델객체와의 인터페이스 등을 기술한다. 일반적으로 의사결정모델은 구조적 특성(structural properties)과 행위(behavior)로 표현되어진다. 구조적 특성은 그 문제를 구성하는 실체/객체와 이들간의 관계를 표시하며, 행위는 그 시스템의 통태적 특성을 규정하는 메소드(절차 및 함수적 인과관계)로 표시되어진다.

모델링 패러디임마다 각기 다른 가정을 하므로 동일한 문제영역에서 다른 형태의 의사결정모델이 만들어 질 수 있다. 각 모델링 패러디임에 상응하는 모델클래스가 형성되며, 이것 역시 타입화 할 필요가 있다. 예를 들면 선형계획 모델링 패러디임에서는 문제영역 내의 실체와 이들 간의 관계를 선형대수관계로 표현하므로 LP모델 클래스가 만들어진다. 또한 Netform 모델링(Glover et. al. 1990)에서는 문제영역의 실체와 관계를 마디와 가지로 표현한 Netform 모델클래스를 만든다.

2. 추상화 수준에 따른 모델베이스

모델베이스를 구성하는 모델클래스는 그 추상화 수준에 따라 패러디임 수준과 문제영역수준으로 나눌 수 있으며, 문제영역은 다시 특정 조직과는 독립적인 일반문제영역(general problem domain)과 특정 조직에 한정된 조직한정 문제영역(organization-specific problem domain)으로 세분화될 수 있다. (그림 4-2)는 그 추상화 수준에 따

〈그림 4-2〉 모델베이스 개발을 위한 주요 연구



라 모델베이스 내에 존재하는 모델클래스들을 분류한 것이며 이들 간의 계승관계를 보여주고 있다.

따라서 모델베이스를 구성하는 모델클래스는 그 추상화수준에 따라 다음과 같이 나눌 수 있다.

- 일반모델클래스(generic model class)
  : 모든 모델링 패러디임을 수용할 수 있는 모델스키마.
- 패러디임-한정 모델클래스(paradigm-specific model class) : P-모델클래스
  : 특정 모델링 패러디임에서 요구되는 가정들을 수용한 모델스키마
- 영역-한정 모델클래스(domain-specific model class)
  : D-모델클래스 또는 모델템플렛(model template)
  : 특정 영역의 문제를 특정 모델링 패러디임에 따라 모델링한 모델스키마
- 모델인스턴스(model instance)
  : 모델클래스에 구체적인 데이터(파라메터) 값이 부여된 모델

이러한 각 수준의 추상화는 모델베이스 관련자의 특정 관점에 상응된다. 즉, 모델수

립자는 자신이 익숙한 모델링 패러디임에 따라 일반적 문제영역에서 문제를 모델화한 패러디임-한정 모델스키마(paradigm-specific model schema)에 관심을 가질 것이며, 의사결정자는 자신의 조직 내의 특정문제를 위해 제작된 문제영역-한정 모델클래스(모델템플렛)와 모델인스턴스에 관심을 가질 것이다.

일반모델클래스는 모델클래스의 최상위 클래스로써 모든 모델클래스의 특성을 공유하는 추상적 객체클래스이다. 따라서 일반모델타입은 여러 모델링 패러디임에서 개발한 모델을 수용할 수 있게 하는 P-모델클래스의 슈퍼클래스(super-class)이다. 또한 각 P-모델클래스(예, 선형계획 모델, 시뮬레이션 모델, 휴리스틱 모델, 구조적 모델 등)는 특정 문제영역으로 구체화 될 수 있다. 이를 D-모델클래스라 한다(예, 생산믹스 LP모델, 마아케팅전략 시뮬레이션 모델). 이는 객체지향의 일반화와 구체화의 원리를 적용한 것이다. 따라서 모델베이스 내에 특정 모델링 패러디임을 수용할 수 있는 P-모델타입이 있으며, 이로부터 특정 문제영역에 대한 의사결정모델인 모델템플렛을 쉽게 도출할 수 있다. 모델템플렛은 그 조직의 특성에 관한 데이터(모델데이터)와 결합하여 모델인스턴스를 만들 수 있는 구체적인 객체타입이다.

이러한 모델타입분류에 대한 연구로는 Banerjee & Basu(1993)의 연구를 들 수 있다. 이들은 모델추상화 수준을 ① 환경적 수준(environment level,) ② 구조 수준(structure level), ③ 인스턴스 수준(instance level), ④ 해산출기 수준(solver level)으로 나누어 각 모델링 패러디임에서 개발된 모델을 분류하고 있다.

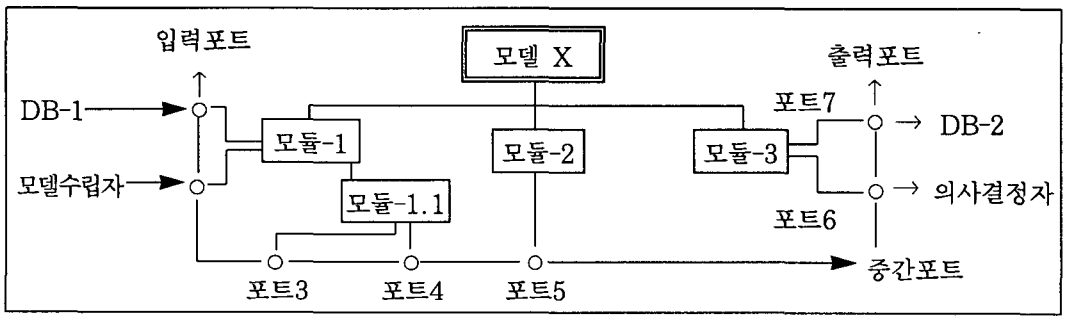
3. 일반모델의 구성요소

일반모델은 현실 세계의 문제에 대한 추상적 표현으로써 모든 모델클래스에 대하여 특성을 계승시킬 수 있는 모델클래스이다. 일반모델에 대한 객체타입이 일반모델타입이다. 모델스키마는 그 관점에 따라 외부명세와 내부명세로 나눌 수 있다. 모델의 외부명세는 그 모델과 관계된 외부환경(모델수립자, 데이터베이스, 의사결정자, 다른 모델)과의 인터페이스를 나타낸다. 내부명세는 모델의 구조적, 행위적 특성을 명세한다. 즉, 모델구성요소들 간의 관계를 나타낸다. 일반모델은 외부 인터페이스로서 포트(port) 집합을 가지며, 포트를 통하여 외부환경과 상호 작용한다. 포트는 포트타입(port type)에 의하여 특징지워진다. 따라서 일반모델타입은 포트(port)라는 연결점들로써 그 기능과 가정을 기술한

다. 포트는 다른 모델객체나 모델의 환경(데이터베이스, 사용자, 모델수립자)간의 인터페이스를 형성한다. 포트에는 입력포트(input ports)와 출력포트(output ports), 그리고 중간포트(mid ports)가 있다. 입력포트와 출력포트는 각각 모델의 외생변수 및 내생변수(또는 computed variables)와 관련 있다. 즉, 입력포트와 출력포트는 외부환경과 관련하여 데이터를 받아들이거나 산출한다. 반면에 중간 포트는 모델의 중간계산결과를 보유하거나, 제약조건을 포함한다. 포트를 분류하는데 있어 (특히 중간포트의 경우) 모델의 의미론에 근거하여 자의적인 판단이 필요하다(허순영 1994, p.45).

포트는 프로그래밍언어의 변수와 같이 타입화된다. 따라서 포트는 그 포트타입(port type)에 의해 특징 지워지며, 그와 관련된 대수적 표현이나 데이터에 대한 정보를 기술하기 위해서 고유한 포트명과 속성집합 및 오퍼레이션을 가진다. 입력과 출력포트는 각각 외부에서 데이터를 받아들이는 역할과 외부로 내보내는 역할을 한다. 외부 인터페이스는 모델의 단순화된 관점을 제공해줌으로써 매우 복잡해 보이는 모델의 세부적인 사항을 사용자가 알아야 하는 부담을 줄여준다.

모델에 대한 내부관점은 모델내부의 세부적인 의미(semantics)를 포착하는 것으로 모델을 구성하는 요소들 간의 구조적 관계(상호작용관계, 함수적 의존관계, 호출순서 등)를 나타낸다. 모델의 구성요소들은 그와 관련된 포트와 함께 모델의 논리적 구성에 따라 보다 상위수준의 객체로 모을 수 있다. 이렇게 모여진 구성체를 모듈(module)이라 하며, 모듈은 다시 그 논리적 구성에 따라 보다 상위수준의 모듈로 조직화 될 수 있다. 일반모델은 이러한 모듈들의 최상위 집합(aggregation)이다. 따라서 포트는 모듈화(modulization)를 통하여 그 추상화 수준(집합수준)을 높일 수 있다. 일반모델의 개념을 도식하면 (그림 4-3)과 같다.



〈그림 4-3〉 일반모델의 개념적 구조

4. 개념적 스키마와 논리적 스키마

모델스키마는 특정 모델링시스템의 문법적 요구와는 독립적인 개념적 스키마(conceptual schema)와 특정 모델링시스템의 문법적 요구에 따라 모델을 표현한 논리적 스키마(logical schema)로 나눌 수 있다. 개념적 스키마는 모델수립자 또는 사용자의 관점을 나타내며, 논리적 스키마는 컴퓨터 시스템의 실행 관점을 나타낸다. 따라서 모델스키마 역시 타입화 할 수 있다. 모델스키마타입은 개념스키마타입(conceptual_schema_type)과 논리스키마타입(logical_schema_type)으로 세분화된다. 개념스키마는 개발자의 관점을 묘사한 개념적 모델(주로 다이어그램이나 그래프 형태를 사용함)을 표현하는 객체타입이며, 논리스키마타입은 컴퓨터 상에서 실행 가능한 모델링언어를 표현하기 위한 객체타입이다.

Ⅴ. 객체지향 모델베이스의 설계

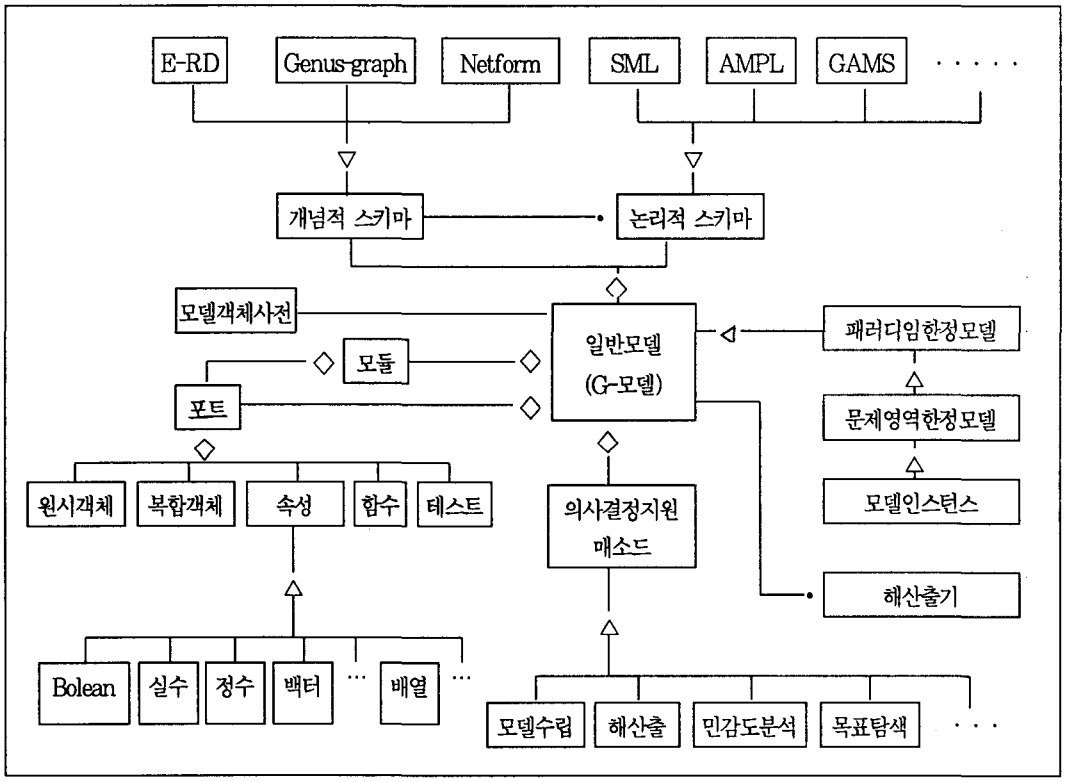
객체지향 모델베이스의 개발을 위해서는 모델베이스를 구성하는 여러 구성요소들을 효과적으로 관련지우고, 이들 구성요소들을 관리하는 데 필요한 객체들을 설계해야 한다. 본 장에서는 이들 구성요소들 간의 개념적 관계를 객체모델로 표현고, 모델베이스 관리에 필요한 일반모델, 모듈, 포트, 객체사전 등의 객체를 설계한다.

1. 메타-모델베이스

모델베이스 내에 존재하는 여러 유형의 객체(일반모델클래스, P-모델클래스, D-모델클래스, 모델인스턴스, 모듈, 포트, 해산출기 등)를 체계적으로 관련 지우고 이를 효과적으로 저장, 관리하기 위해서는 모델베이스를 구성하는 요소들 간의 의미론적 관계를 표현하는 메타수준의 모델베이스를 설계할 필요가 있다. 이러한 모델베이스의 전체적인 구성에 대한 정보를 담은 메타수준의 모델베이스 스키마를 메타-모델베이스(meta-model-base)라 한다.

메타모델베이스는 앞에서 제시된 여러 유형의 모델타입과 이들 객체에 대한 정보를 담은 지식과 데이터들 간의 관계를 개념적으로 표현한다. 메타모델베이스 설계를 위해서 사

용하는 도구는 앞에서 제시된 객체모델링기법의 객체다이어그램이다. 메타-모델베이스를 객체다이어그램으로 표시하면 (그림 5-1)과 같다.



〈그림 5-1〉 메타 모델베이스 설계

모델베이스 내에는 특정 모델링 패러디임과는 독립적인 일반적 모델의 구조를 갖는 일반모델클래스, 모델링 패러디임 수준의 모델을 나타내는 P-모델클래스(예, LP모델), 특정 의사결정영역의 문제를 해결하는 데 사용되는 D-모델클래스 또는 모델템플릿(예, 정유회사의 제품믹스 LP모델) 간에는 일반화와 구체화 관계를 나타낸다.

모델은 하나 또는 그 이상의 모듈 또는 포트로 구성되며, 모듈 역시 하나 또는 그 이상의 포트로 구성된다. 포트는 그 타입에 따라 다시 다음과 같이 나눈다.

• 원시객체(primitive object): 수학적으로 정의되지 않는 요소로써 관련된 값을 갖지 않으며, 모델에서 고유하게 식별될 수 있는 실체(사람, 장소, 사물, 행동, 사건, 질, 상

태 등)이다. 즉, 모델 수립자가 축소 불가능하거나 분석할 수 없는 사물을 표현하는 데 있어 가정 없이 자의적으로 도입되는 실체이다.

• 복합객체(compound object): 이미 정의된 원시객체나 또는 다른 복합객체들을 참조하여 정의된 객체(사물이나 개념)로써 관련된 값을 지니지 않는다. 복합객체는 이산 수학에서는 집합이나 릴레이션의 한 멤버로 표현되어질 수 있다.

• 속성(attribute): 실체(사물이나 개념)의 특성을 나타내는 것으로 일정한 값을 갖는다. 보통 모델의 "계수(파라메터)"나 "의사결정변수"들이 속성에 해당한다. 모델에서 상수나 계수와 같이 그 값이 미리 정해진 속성을 단순 데이터 속성(data attribute)이라 하며, 의사결정자의 행동에 의해서 정해지는 속성을 변수속성(variable attribute) 또는 의사결정속성(decision attribute)이라 한다. 그리고 속성들 간의 함수적 관계를 통하여 도출되는 유도속성(derived attribute)과 성과를 측정하기 위한 성과속성(performance attribute)이 있다. 속성은 단순한 Boolean형, 문자형, 실수형, 정수형에서부터 문자열, 벡터형, 매트릭스형과 같은 복합형태의 데이터타입을 지닌다.

• 함수(function): 특정 속성집합을 다른 속성(유도속성)으로 전환하는 역할을 한다. 따라서 함수는 속성들 간의 호출규칙에 따라 유도속성을 계산하는 절차의 집합이다.

• 제약테스트(constraint test): 속성들의 제약적 관계를 규정하는 것으로 참 또는 거짓의 두 값만을 가진다. 테스트는 속성들을 서로 비교하여 비교된 결과를 되돌려 주는 역할을 한다.

특정 문제영역에서 모델은 하나의 개념적 스키마와 여러 개의 논리적 스키마로 구성된다. 논리적 스키마는 모델을 실행하고자 하는 모델관리시스템의 요구사항에 맞게 모델을 표현한 것이다. 이는 모델에 대한 다양한 관점을 나타내 준다.

모델객체사전(model object dictionary)은 모델베이스 내에 존재하는 여러 가지의 모델객체들에 대한 정보를 담은 사전으로 객체생성시 객체에 대한 정보를 참조하게 한다.

하나의 모델은 여러 개의 해산출기를 가질 수 있다. 예를 들면 할당문제를 해결하기 위해서는 일반 심플렉스 해법(simplex method)과 같은 최적화 앨고리즘을 적용할 수도 있으며, 헝가리해법(Hungarian method)과 같은 휴리스틱 해법을 적용할 수도 있다.

모델을 통한 의사결정지원 방법은 여러 가지의 형태를 지닌다. 이것은 모델링 패러디임에 따라 약간의 차이를 보인다. 예를 들어 최적화 모델링 패러디임의 경우 의사결정지원을 위한 메소드로는 수리적 형태의 모델수립, 해산출 앨고리즘의 실행, 매개변수의 변

화에 따른 민감도 분석 또는 가상질의(What-if)분석, 원하는 결과치를 얻기 위한 목표탐색(goal-seeking) 등의 메소드가 있다. 이들 메소드들은 모델클래스의 메소드로서 이들 만을 모아 하나의 의사결정지원 메소드 클래스를 만들 수 있다.

2. 모델베이스 내의 객체설계

모델베이스 내에 존재하는 모델들을 효과적으로 저장하고 관리하기 위해서는 모델베이스 내의 각 객체타입들(일반모델, 모듈과 포트객체)에 대한 설계가 이루어져야 한다. 먼저 일반모델객체를 설계하면 (그림 5-2)와 같다. 일반모델의 속성으로는 모델에 대한 식별자, 모델에 대한 정보를 담은 모델사전을 참조하는 참조자, 그리고 모델의 구성요소인

```
OBJECT-TYPE Generic_Model HAS
  ATTRIBUTES:
      model-id : model identifier
      model_name : model_dictionary;
      inport : LIST OF Port  // for referential integrity
      outport : LIST OF Port  // for referential integrity
      midport : LIST OF Port  // for referential integrity
      module : LIST OF Module // for referential integrity
      model_schema : LIST OF Model_Schema; // for referential integrity
  OPERATIONS:
      Generic_Model(model_name);  //constructor
      Show_Interface(model_name);
      Show_Internal_Schema(model_schema)
      Solve_Model(model_name, solver_name);
      Type(model_name);
END Generic_Model;
```

〈그림5-2〉 일반모델타입의 표현

모듈과 포트를 참조하는 참조자들로 구성된다. 이것은 모델을 구성하는 요소들 간의 참조적 무결성을 지니기 위해서 필요하다. 그리고 오퍼레이션의 Generic_Model (model_name)은 생성자(constructor)이며, Show_ Interface()는 (그림 4-3)의 바깥부분과 같이 모델의 외부구조, 즉 다른 객체와의 인터페이스를 보여준다. Show Inter-

nal_Schema()는 (그림 4-3)의 안쪽 부분과 같이 모델을 구성하는 모듈과 포트들 간의 의존적 관계(모델의 내부스키마)를 보여준다. 그리고 Solve_Model()은 모델의 해를 구하는 오퍼레이션이며, Type()는 그 모델의 타입을 알려주는 오퍼레이션이다. 만일 모델이 LP모델인 경우 Type(model_name)의 결과는 LP이다. 이들 오퍼레이션은 모든 모델클래스에 대하여 상속할 수 있는 추상오퍼레이션(abstract operation)이다.

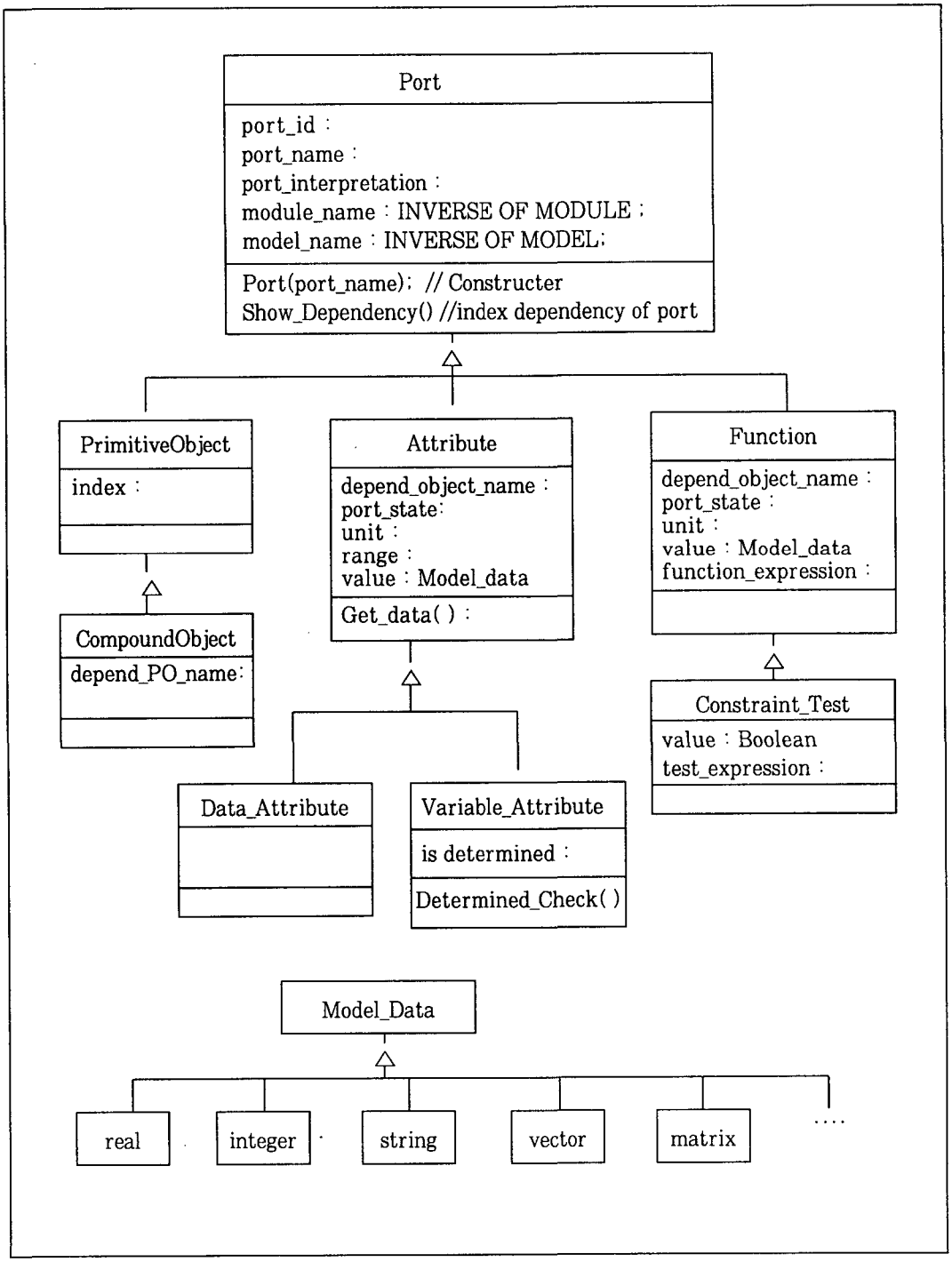
일반모델은 그 모델을 구성하는 모듈들의 집합체이므로 모듈 역시 타입화된다. 따라서 모듈은 포트와 자녀모듈(child module)의 집합으로 모델의 중간 빌딩블록의 역할을 한다. 이러한 모듈객체를 설계하면 (그림 5-3)과 같다. 모듈객체의 속성은 일반모델의 속성

```
OBJECT-TYPE Module HAS
  ATTRIBUTES:
      module_id : module identifier
      model_name : INVERSE of Generic_Model;
      model_schema_name : INVERSE of Model_Schema;
      parent_module_name : INVERSE OF Module
      child_module_name : LIST OF Module
      inport : LIST OF Port;
      outport : LIST OF Port;
      midport : LIST OF Port
  OPERATIONS:
      Module(module_name);  //constructor
      Module(module_name, model_name);
      Module(module_name, port_name);
      Show_Module_Internal(module_name);
  END Module;
```

〈그림 5-3〉 모듈타입의 표현

과 마찬가지로 그 모듈의 식별자, 그 모듈을 포함하고 있는 모델과 구성요소인 하위모듈 또는 포트를 참조하는 참조자들로 구성된다. 그리고 오퍼레이션 Module()은 모듈을 생성하는 생성자이며, Show_Module_Internal()은 모듈의 내부구조를 보여주는 오퍼레이션이다.

포트객체는 데이터베이스나 또는 다른 모델로부터 받아들여진 데이터 값과 모델의 중간 또는 최종 출력 결과 값을 일시적으로 저장하는 기능을 가지므로 (그림 5-4)과 같이



〈그림 5-4〉 포트객체의 설계

객체를 설계할 수 있다. 포트객체는 그 타입을 다시 원시객체, 속성, 함수로 파생할 수 있으며, 원시객체로부터 복합객체로, 속성을 다시 데이터속성과 의사결정속성(변수속성)으로, 함수로부터 제약테스트 객체로 각각 파생할 수 있다.

포트는 포트명과 포트에 대한 간단한 해설을 정의할 수 있게 하며, "model_name"과 "module_name"은 이 포트를 포함하고 있는 모델과 모듈을 참조한다. 이것은 포트와 이 포트를 가진 모델 및 모듈간의 참조적 무결성(referential integrity)을 보증하기 위한 것이다. 속성과 함수포트의 "port_state"는 포트의 입력과 출력상태를 구분하며, "unit"는 데이터 값의 단위를 나타낸다. 그리고 속성의 "range"는 그 데이터 값의 범위를 제약하기 위해서 필요하며, "value"는 포트의 데이터 값을 나타낸다. 원시객체와 복합객체의 "index"는 속성들 간의 함수적 관계표현시 그 속성이 의존하는 객체를 참조할 수 있게 한다. 한편 복합객체, 속성과 함수포드에 있는 "depend_object_name"는 그 포트가 고유하게 의존하는 객체를 나타내기 위해서 필요하다. 함수포트의 "function_expression"과 제약테스트포트의 "test_expression"은 각각 함수적 관계와 제약관계를 대수식으로 표현할 수 있게 한다.

포트의 오퍼레이션 Show_Dependency()는 포트객체들 간의 의존관계를 보여주며, 속성포트의 Get_Data()는 포트의 데이터 값을 받아들이는 역할을 한다. 그리고 변수속성포트의 Determined_Check()는 그 속성의 값이 결정되어졌는가를 체크한다. 속성포트에 저장되는 모델데이터타입은 단일의 실수 값에서부터 매트릭스까지 다양한 포맷의 수치값 뿐만 아니라 문자열도 포함한다.

```
OBJECT-TYPE Model_Schema HAS
  ATTRIBUTES:
      model_schema_name : string;
      comment : string;
      model_name: INVERSE OF Generic_Model;//for referential integrity.
      c_s_name : LIST OF Conceptual_Schema;
      l_s_name : LIST OF Logical_Schema;
  METHODS;
      Model_Schema();          //constructer
      Show_Schema();  //show model's internal structure
      Convert_Schema(c_s_name, l_s_name)
  END Model_Schema;
```

〈그림 5-5〉 모델스키마타입의 표현

모델스키마타입은 모델의 내부구조를 명세하는 것으로 개념스키마타입(conceptual_schema_type)과 논리스키마타입(logical_schema_type)으로 나눌 수 있다. 모델스키마타입은 특정 모델링 패러디임에서 모델을 표현하는 형태와 같이 모델을 사용자에게 보여주는 역할을 한다. 이들 스키마타입을 표현하면 (그림 5-5)와 같다. Show_Schema()는 모델의 스키마를 보여주는 오퍼레이션이며, (그림 5-5)의 오퍼레이션 Convert_Schema(c_s_name, l_s_name)는 모델의 개념적 스키마를 논리적 스키마로 전환하는 오퍼레이션이다.

(그림 5-6)의 모델객체사전은 모델베이스 내에 존재하는 모델에 대한 정보를 관계형 테이블로 만들어 저장관리할 수 있다. 이는 모델수립자 또는 의사결정자가 원하는 모델이 모델베이스 내에 있는가를 상담하는 데 사용될 수 있다. 모델객체사전은 모델타입, 응용영역, 경영계층, 모델개발자, 해산출 앨고리즘 등과 같은 다양한 측면에서 원하는 모델을 쉽게 찾을 수 있게 한다.

```
OBJECT-TYPE Model_Dictionary IS-A Relational_Table;
  ATTRIBUTES:
      model_id   :
      model_type :
      application_domain :
      management_level :
      model_developer :
      solver_algorithm :
      interpretation :
      OPERATION:
  Generate_Table();
      Read(); Write(); Open(); Close();
      Append_record();  Insert_record(); Delete_record();  Recall_record();
  END Model_Dictionary;
```

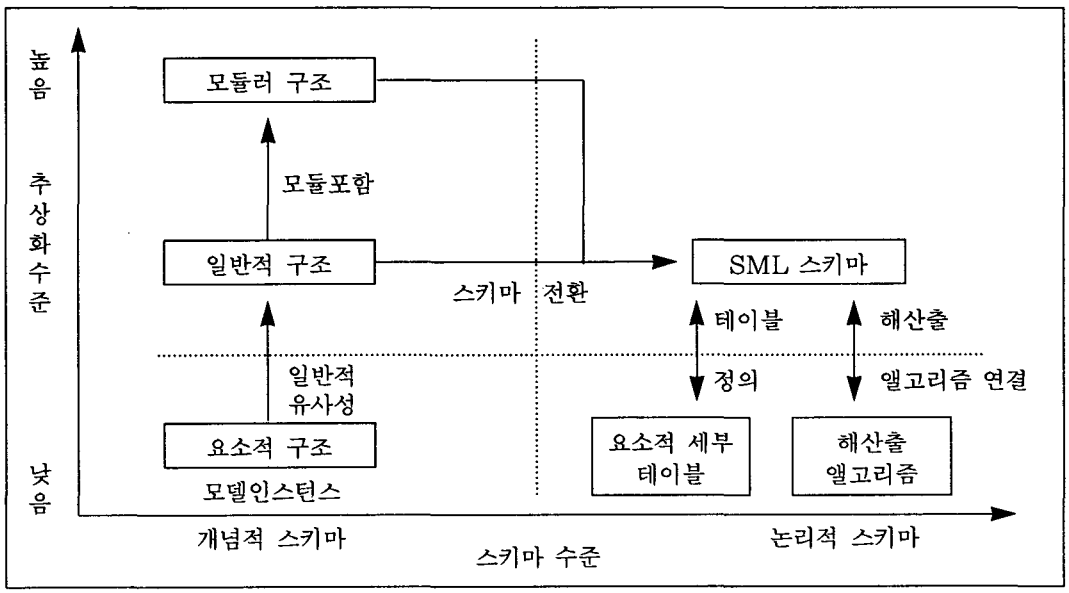
〈그림 5-6〉 모델객체사전

Ⅵ. 모델링 패러디임 수준에서의 적용

객체모델링기법을 적용한 모델베이스 설계가 모델링 패러디임수준에서 적용가능함을 보이기 위해서 본 연구에서는 Geoffrion(1987)의 구조적 모델링분야를 선택하여 객체지향 모델베이스의 설계 예를 보여 주고자 한다.

1. 객체지향적 관점에서 본 구조적 모델링

구조적 모델링에서 구조적 모델은 그 추상화 수준에 따라서 모델의 세부적인 모든 요소들을 모두 다 표현한 요소적 구조와 요소들 간의 일반적 유사성을 가진 요소들끼리 묶어 하나의 포트로 추상화한 일반적 구조, 그리고 일반적 구조를 한 단계 더 추상화하여 관련 있는 포트를 하나의 모듈로 묶어 나무형태(비순환 그래프)로 표현한 모듈러 구조를 지닌다. 그리고 이러한 개념적 스키마를 컴퓨터 상에서 곧바로 실행 가능하도록 하기 위한 모델링 언어가 SML(Structured Modeling Language)이다. SML로 표현된 모델 스키마가 논리적 모델스키마이다. 그리고 모델-데이터 독립성을 보장하기 위해서 모델의



〈그림 6-1〉 구조적 모델의 구성

데이터는 요소적 세부 테이블(elemental detail table)에 저장되어진다. 그리고 모델-해산출기 독립성을 보장하기 위해서 해산출 앨고리즘도 독립적으로 저장되어진다. 이들 구조적 모델링에 관련된 여러 가지 수준의 모델들을 추상화 수준(abstraction level)과 스키마 수준(schema level)의 양 차원에서 도식화하면 (그림 6-1)과 같다.

이러한 구조적 모델링을 통해 만들어진 구조적 모델은 모델구성요소에 대한 모듈러적, 계층적 특성들을 가장 잘 나타내주므로 앞 장에서 설명한 여러 가지 모델타입의 개념과 가장 잘 대응된다. 즉, 구조적 모델의 모듈개념은 일반모델타입의 모듈개념과 일치하며, Genus는 포트에 대응된다. 그리고 구조적 모델의 호출순서(calling sequence)는 각 Genus 간의 정의적 상호의존관계를 나타내므로 모델의 내부적 관점을 보여줄 수 있다.

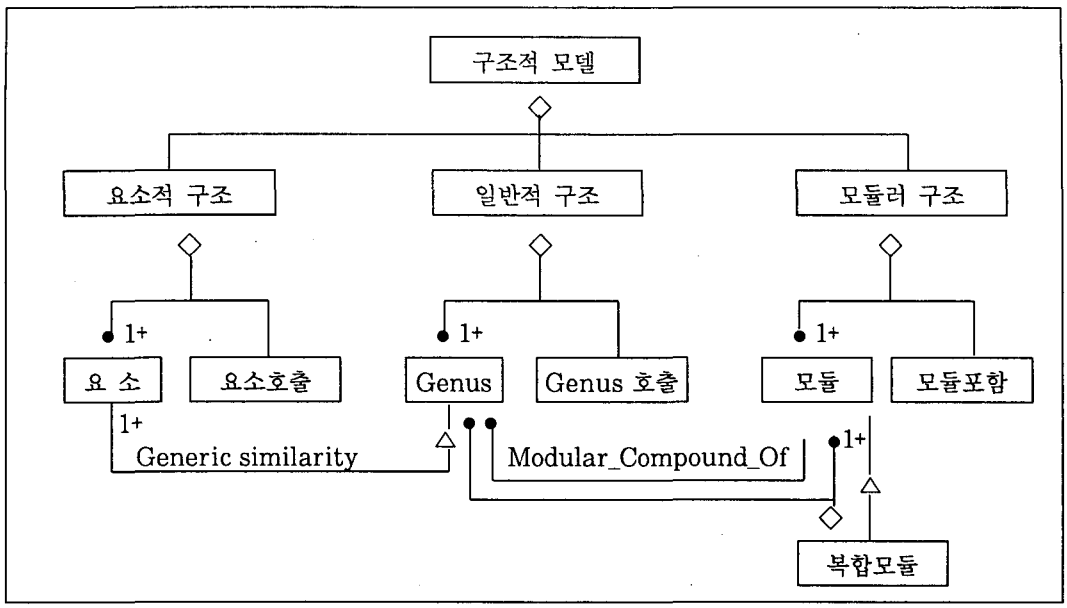
따라서 객체지향적 관점에서 구조적 모델을 볼 때, 모듈러 구조는 Genus와 모듈의 계층적 집합을 나타낸 그래프이며, 일반적 구조는 모델을 구성하는 구성요소(Genus)들 간의 호출관계를 보여주는 그래프이다. 따라서 이 두 그래프는 추상화 수준이 다른 개념적 모델이므로 개념적 스키마의 하위클래스로 볼 수 있다. 또한 SML 스키마는 직접 컴퓨터 상에서 실행가능한 언어로 작성되므로 모델의 논리적 스키마의 하위타입으로 볼 수 있다. 그리고 요소적 구조는 그 모델클래스의 구체적인 예를 나타내는 모델인스턴스가 된다.

2. 구조적 모델링을 위한 객체지향적 모델베이스 설계

구조적 모델을 객체지향적으로 표현하고 조직화 할 수 있다면, 객체지향적 모델관리시스템에서 구조적 모델을 쉽게 다룰 수 있을 것이다. 구조적 모델링의 구조적 모델을 구성하는 객체 클래스와 이들 간의 관계를 객체다이어그램으로 나타내면 (그림 6-2)와 같다. 여기서 구조적 모델은 요소적 구조와 일반적 구조, 모듈러 구조의 집합관계(◇)로 표시되어진다.

요소적 구조는 하나 이상의 요소(객체다이어그램에서 •는 연관관계의 다중성을 나타내며, 1+은 연관관계가 하나 이상임을 표시함)와 이들 간의 의존관계를 나타내는 요소호출로 구성된다.

일반적 구조는 하나 이상의 Genus와 이들 간의 Genus 호출로 구성되며, 모듈러 구조는 하나 이상의 모듈 또는 복합모듈과 이들 간의 모듈포함관계로 구성된다. 또한 Genus는 동일한 특성을 갖는 요소들로 구성되며, 모듈은 하나 이상의 Genus로 구성된



〈그림 6-2〉 구조적 모델의 객체다이어그램

다. 복합모듈은 하나 이상의 모듈들로 구성되거나 또는 모듈과 Genus로 구성되며, 모듈의 속성을 이어받는다.

한편 그래프 중심의 모델링시스템의 입장에서 본다면, 요소 클래스, Genus 클래스, 모듈 클래스는 각각 요소적 그래프, Genus 그래프, 모듈러 그래프 상에서는 마디(node)로 표시되므로 마디 클래스(node class)로 일반화될 수 있다. 또한 요소호출 클래스와 Genus호출 및 모듈포함 클래스는 가지(arc)로 표시되므로 가지 클래스(arc class)로 일반화될 수 있다.

구조적 모델링은 원소모델(atomic model)을 모델링하는 데 있어 훌륭한 방법론이 된다. 구조적 모델링을 통해 나온 구조적 모델은 모델의 개념적 스키마와 논리적 스키마를 제공할 수 있다. 즉, 구조적 모델의 기본구조인 요소적 구조와 이를 일반화한 일반적 구조, 그리고 일반적 구조를 한 단계 더 추상화한 모듈러 구조는 모델의 개념적 스키마를 명세하는 데 사용될 수 있다. 그리고 이들 개념적 수준의 모델을 논리적 모델스키마로 표현하는 데 사용하는 모델링 언어가 SML이다.

구조적 모델의 모듈러 구조와 일반적 구조를 설계하면 (그림 6-3) 및 (그림 6-4)와 같다. 그리고 SML 스키마 클래스를 설계하면 (그림 6-5)과 같다. (그림 6-3) 및 (그림 6-

4)에서 HEURISTICS에 있는 내용들은 각각 모듈러 구조와 일반적 구조의 무결성(integrity)을 보증하기 위한 지식이다.

그리고 메소드 Show_Master_View()와 Show_Partial_View는 각각 구조적 모델의 마스터 뷰(master view)와 부분 뷰(partial view)를 보여주는 메소드이며, Show_Genus_Structure()는 Genus 그래프를 보여주는 메소드이다. Convert_SML_Schema는 모듈러 그래프와 Genus 그래프에 근거하여 텍스트 형태의 SML 스키마로 전환하는 메소드이다.

(그림 6-5)의 SML 스키마 클래스는 구조적 모델의 논리적 스키마를 나타내는 클래스로 모델의 텍스트 편집을 가능하게 하는 Edit_SML_Schema 메소드와 해산출기와의 인터페이스를 하는 Solver_Interface() 메소드를 가지고 있다.

또한 구조적 모델을 구성하는 Genus 객체를 그 타입에 따라 계승관계로 표현하면 (그림 6-6)과 같다. 그리고 구조적 모델을 구성하는 객체들 간의 관계(요소호출, Genus 호출, 모듈포함관계)에 대한 정보와 모델의 구체적인 데이터를 담은 요소세부테이블을 설계하면 (그림 6-7)과 같다. 이것들은 관계형 테이블로 설계되어 질 수 있다.

```
OBJECT-TYPE Modular_Structure IS-A Conceptual_Schema;
 ATTRIBUTES:
     modular_stru_name : string;
     no_of_module: integer;
     module_name : array[] of Module;
     module_contain : Module_Contain_Table;
 MEMBERS:
     genus_stru_name : LIST OF Generic_Structure ;
     SML_schema_name : LIST OF SML_Schema;
 HEURISTICS:
     module_calling_rule;
     monotone_ordering;
     rooted tree;
     acyclicity;
 METHODS:
     Modular_Structure(); //constructer
     Show_Master_View();
     Show_Partial_View();
     Insert_module();
     Delete_module();
     Insert_Calling_Arc();
     Grouping_module();
     Decompose_module();
 END Modular_Structure;
```

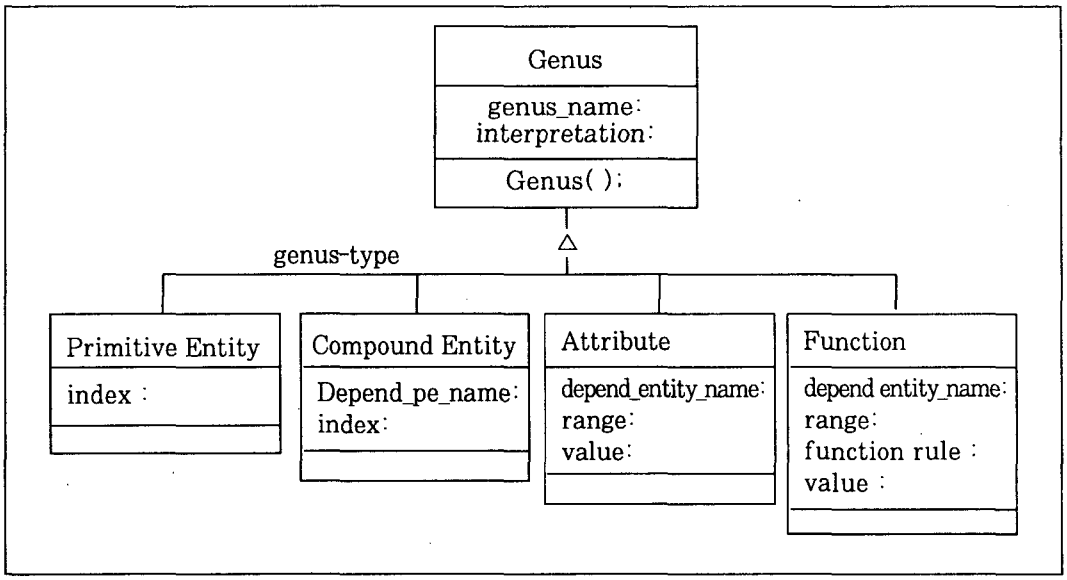
〈그림 6-3〉 모듈러 구조의 객체설계

```
OBJECT-TYPE Genus_Structure IS-A Conceptual_Schema;
  ATTRIBUTES:
      genus_stru_name : string;
      no_of_genus: integer; //number genus structured model.
      genus_name : array[] of Genus;
      no_of_depend : integer; //number of genus calling in structured model.
      genus_calling : Genus_Call_Table;
  MEMBERS:
      elem_stru_name : LIST OF Elemental_structure ;
      modular_stru_name : LIST OF Modular_structure ;
      SML_schema_name : LIST OF SML_Schema;
  HEURISTICS:
      generic_similarity;
      acyclicity;
      calling_constraint;
      genus_calling_rule;
  METHODS:
      Genus_Structure(); //constructer
      Show_Genus_Structure();
      Insert_genus();
      Delete_genus();
      Insert_Calling_Arc();
      Make_Genus_Calling_Seq_Table();
      Convert_SML_Schema();
      Convert_Modular_Structure();
  END Genus_Structure;
```

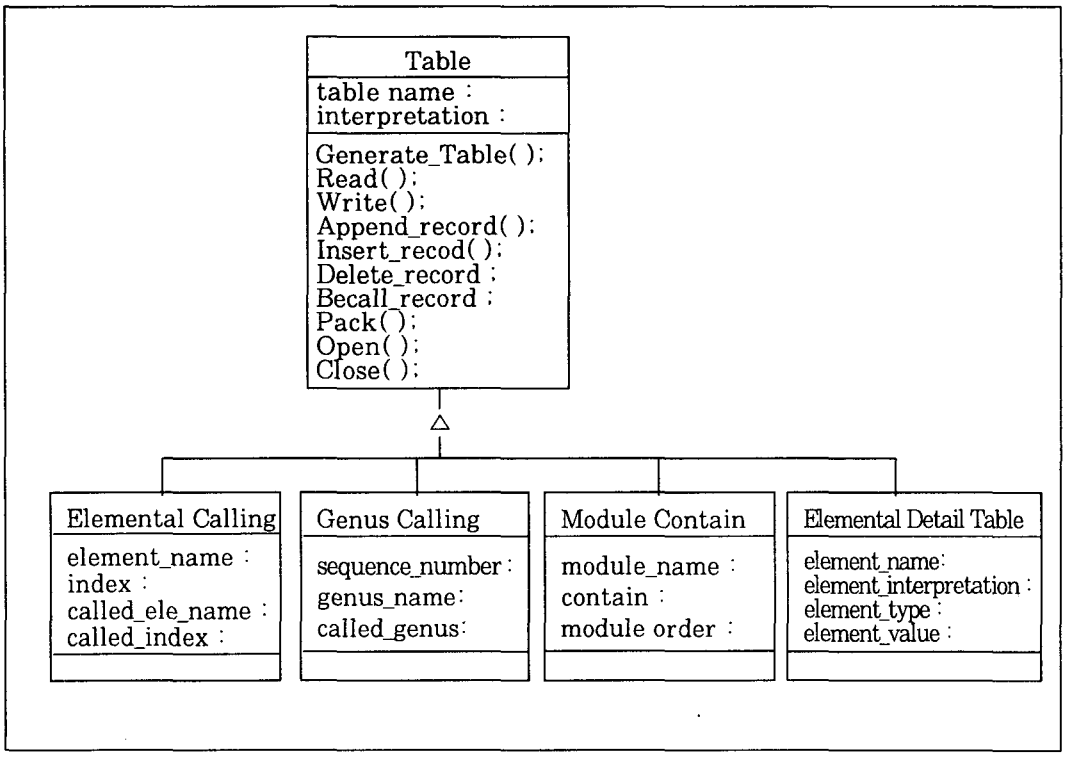
〈그림 6-4〉 일반적 구조의 객체설계

```
OBJECT-TYPE SML_Schema IS-A Logical_Schema;
  ATTRIBUTES:
      SML_schema_name : string;
  MEMBERS:
      module_name : LIST OF Module ;
      genus_name : LIST OF Genus;
  METHODS:
      SML_Schema(); //constructer
      Show_SML_Schema();
      Edit_SML_Schema();
      Solver_Interface();
 END SML_Schema;
```

〈그림 6-5〉 SML 스키마의 객체설계



〈그림 6-6〉 Genus 클래스 객체설계



〈그림 6-7〉 구조적 모델을 위한 데이타베이스 테이블 객체의 설계

제Ⅷ장. 結　　論

DSS의 모델베이스는 현실 세계에 존재하는 단순한 데이터의 집합체가 아니라 이러한 데이터를 변환하고 분석하는 절차적, 비절차적 지식을 내포한 지식의 집합체이다. 따라서 모델베이스는 서로 이질적인 구성요소(데이터, 모델, 지식, 해산출 앨고리즘)들을 저장하고, 관리해야 한다. 이러한 이질적인 요소들을 표현하고 관리하는 데 있어 객체지향의 여러 개념(일반화, 집합, 타입화)들이 유용하게 적용될 수 있다.

모델베이스설계는 메타수준, 개념적 수준, 논리적 수준으로 나눌 수 있다. 메타수준의 모델베이스(메타-모델베이스)는 모델베이스를 구성하는 구성요소들에 대한 전체적인 정보를 나타내는 스키마이다. 본 연구에서는 메타-모델베이스를 설계하기 위해서 OMT의 객체모델을 이용하여 모델베이스 내에 존재하는 여러 객체들 간의 개념적 관계를 표현하고 있다.

지금까지 모델베이스 개발과 관리를 위해서 여러 가지의 프레임웍들(수리적 모델링 언어, 데이터베이스 지향, 지식베이스 지향, 그래프 지향, 객체지향 프레임웍)이 제시되었으며, 그 학문적, 이론적 배경도 다양하다. 이중에서 모델베이스설계에 있어 객체모델링 기법을 도입할 경우 다음과 같은 장점을 가질 수 있다.

① 유사한 모델구성요소(포트)와 모델링 가정을 지닌 모델에 대하여 모델클래스 개념을 도입할 수 있으며, 이 모델클래스 역시 타입화 할 수 있다. 이를 모델타입이라 한다. 모델타입은 모델링 패러디임마다 만들어질 수 있다.

② 일반화-특수화관계, 즉 계승개념에 의하여 모델베이스 내에 이미 존재하는 모델을 재사용할 수 있게 하므로써 새로운 모델을 쉽게 만들 수 있게 한다. 계승개념은 코드의 재사용을 가능하게 하므로써, 기존의 모델베이스 내에 존재하는 모델객체들 간의 계승관계를 이용하여 신속한 모델수립활동을 돕는다.

③ 집합(aggregation)개념의 사용은 모델클래스를 구성하는 구성요소(포트)들의 계층적 조직화를 가능하게 할 뿐만 아니라 이들 간의 참조적 무결성을 가지도록 한다.

④ 정보은폐(information hiding) 개념은 모델클래스를 구성하는 다양한 타입의 객체와 이들 간의 구조적 관계뿐만 아니라 모델통합과 모델링 지식을 캡슐화하므로서 모델 추상화를 가능하게 한다. 즉, 모델베이스를 구성하는 이질적인 형태의 복잡한 요소들(모델데이터, 모델링 지식, 해산출 앨고리즘)을 하나의 틀 속에 캡슐화(encapsulation) 할 수 있게 한다. 즉, 모델베이스에 정보은폐 개념을 도입할 경우 DB의 데이터추상화(data

abstraction)와 유사한 모델추상화(model abstraction) (Dolk & Konsynski 1984)를 가능하게 한다.

여러 패러디임에서 개발된 다양한 유형의 모델을 단일의 모델베이스에 저장하여 관리하기 위해서는 그래픽(아이콘), 네트웍, 그리고 대수적 표현법과 같은 여러 가지의 외부적 모델표현을 수용할 수 있는 방법을 개발해야 한다. 이를 위해서 본 연구에서는 메타-모델베이스개념과 일반모델타입 개념을 도입하고 있다. 그런데 본 연구에서는 메타-모델베이스에 대한 개념적 수준의 설계만을 제시하고 있으며, 이의 구현을 위한 완전한 메소드를 설계하지 않았다.

또한 지식중심의 시스템 설계에 있어 OMT 기법이 얼마나 잘 적용될 수 있는가를 검토하여야 한다. 왜냐하면 모델 관리를 위해서는 다양한 형태의 지식을 체계적으로 저장하고 관리할 수 있는 매카니즘이 필요하다. 예를 들어, 어느 한 패러디임에서 개발된 모델을 다른 패러디임에서 표현하는 모델로 전환할 경우 많은 지식이 필요하며 이를 자동화하기 위해서는 지식중심의 모델관리시스템이 설계되어져야 하며, 이를 위한 객체지향 지식베이스의 개발이 필요하다. 전문가의 선호도로 인하여 실제로 의사결정자가 원하는 형태의 모델을 제시하지 못하는 이러한 문제는 모델 뷰(model view) 또는 Dolk & Konsynski(1984)의 추상화 공간(abstraction space) 개념의 도입과 뷰 전환과 추상화 전환을 위한 지식중심의 메소드 설계와 구현으로 가능할 것이며, 모든 전환메커니즘에 일반적으로 적용될 수 있는 일반환된 전환규칙과 앨고리즘을 개발해야 할 뿐만 아니라 각 모델타입마다 전환을 위한 세부 규칙과 앨고리즘을 개발해야 한다. 따라서 모델의 추상화 수준에 따라 적절한 규칙과 앨고리즘을 개발할 필요가 있다. 이 때 상위수준의 클래스에서 사용한 전환메소드는 하위수준의 모델전환에 재사용할 수 있는 것에 대한 연구가 진행되어야 한다.

끝으로 특정 의사결정문제영역에서 문제를 구조화하고 이를 모델링하는 데 있어 OMT의 객체모델이 이용될 수 있는지를 검토할 필요가 있다. 문제영역에서의 의사결정요구를 분석하여 문제를 구조화하는 개념적 모델링 도구로써 객체모델을 이용할 경우 객체모델이 지닌 풍부한 모델링 구성자를 충분히 활용할 수 있을 것으로 기대된다. 객체모델은 데이베이스의 개념적 모델링(의미모델링) 도구로써 그 유용성이 입증되고 있다. 그런데 의사결정문제 모델링과 같은 복잡한 문제영역에서 모델링 도구로써 이것이 얼마나 잘 적용될 수 있는가에 대한 연구가 필요하다.

參 考 文 獻

1. 국내문헌

김유일, 김진수, 정대율,"모델관리시스템의 제 접근법에 대한 비교연구," 정보시스템연구, 제3권, 1994년, pp. 101-132.

정대율, "객체지향적 모델관리시스템을 위한 기능요구분석," 정보시스템연구, 제4권, 1995년, pp. 129-153.

정대율, "DSS의 모델베이스 개발을 위한 객체모델링 프레임웍," 박사학위논문, 부산대학교 대학원, 1996.

정대율, "구조적 모델링을 위한 객체지향적 모델베이스 조직화," 정보시스템연구, 제5권, 1996년, pp. 149-173.

허순영, "최적화 모델링 언어를 위한 객체지향 모형관리체계의 개발," 經營科學, 제11권, 제2호, 1994년 6월, pp. 43-63.

2. 외국문헌

Applecate, L.M., B.R. Konsynski, and J.F. Nunamaker, "Model Management Systems : Design for Decision Support," *Decision Support Systems*, Vol. 2, No. 1, 1986, pp. 81-91.

Banerjee, S., A. Basu, "Model Type Selection in an Integrated DSS Environment," *Decision Support Systems*, Vol. 9, 1993, pp. 75-89.

Bharadwaj, A., J. Choobineh, A. Lo, and B. Shetty, "Model Management Systems: A Survey," in: B. Shetty (ed.), *Annals of Operations Research, Model Management Systems*, Vol. 38, 1992, pp. 17-67.

Bhargava, H.K., and S.O. Kimbrough,"Model Management, An Embedded Approach," *Decision Support Systems*, Vol. 10, 1993, pp. 277-299.

Bhargava, H.K., and R. Krishnan,"Computer-aided Model Construction," *Decision Support Systems*, Vol. 9, No. 1, 1993, pp. 91-111.

Binbasiolu, M. , and M. Jarke, "Domain Specific DSS Tools for Knowledge-based Model Building," *Decision Support Systems*, Vol. 2, No. 1, 1986,

pp. 213-223.

Bisschop, J., and A. Meeraus, "On the Development of a General Algebraic Modeling Systems in a Strategic Planning Environment," *Mathematical Programming Study*, Vol. 20, 1982, pp. 1-29.

Blanning, R.W., "A Relational Framework for Join Implementation in Model Management," *Decision Support Systems*, Vol. 1, No. 1, 1985, pp. 69-82.

Blanning, R.W., "An Entity-relationship Approach to Model Management," *Decision Support Systems*, Vol. 2, No. 1, 1986, pp. 65-72.

Blanning, R.W., "A Relational Theory of Model Management," in: (eds.) C.W.Holsapple, and A.B. Whinston, *Decision Support Systems: Theory and Application*, NATO ASI Series Vol. F31, Springer, Berlin, 1987, pp. 19-53.

Blanning, R.W., "Model Management Systems : An Overview," *Decision Support Systems*, Vol. 9, No. 1, 1993, pp. 9-18.

Bonczek, R.H., C.W. Holsapple, and A.B. Whinston, "A Generalized Decision Support System Using Predicate Calculus and Network Database Management," *Operations Research*, Vol. 29, N0. 2, 1981, pp. 263-281.

Chen, Y.S., "An Entity-relationship Approach to Decision Support and Expert Systems," *Decision Support Systems*, Vol. 4, No. 2, 1988, pp. 225-234.

Choobineh, J., "SQLMP: A Data Sublanguage for Representation and Formulation of Linear Mathematical Models," *ORSA Journal on Computing*, Vol. 3, 1991a, pp. 358-375.

Choobineh, J., "A Diagramming Technique for Representation of Linear Models," *Omega, International Journal of Management Science*, Vol. 19, 1991b, pp. 43-51.

Chung, Q.B., and R.M. O'Keefe, "A Formal Analysis of the Model Management Literature," in: B. Shetty (ed.), *Annals of Operations Research, Model Management Systems*, Vol. 38, 1992, pp. 137-176.

Collaud, G., and J. Pasquier-Boltuck, "gLPS: A Graphical Tool for the Definition and Manipulation of Linear Problems," *European Journal of Operational Research*, Vol. 72, 1994, pp. 277-286.

Dempster, M.A.H, and A.M. Ireland, "Object-oriented Model Integration in a

Financial Decision Support System," *Decision Support Systems*, Vol.7, No.3, 1991, pp. 329-340.

Dolk, D.R., "A Generalized Model Management System for Mathematical Programming, *ACM Transactions on Mathematical Software*, Vol. 12, No.2, 1986a, pp. 96-126.

Dolk, D.R., "Data as Models : An Approach to Implementing Model Management," *Decision Support Systems*, Vol. 2, No.1, 1986b, pp. 73-80.

Dolk, D.R., "Object-oriented Model Management," *Paper Presented at the ORSA/TIMS Joint National Meeting*, Phoenix, October 31 - November 3, 1993.

Dolk, D.R., and B.R. Konsynski, "Knowledge Representation for Model Management Systems," *IEEE Transactions on Software Engineering*, Vol. 10, No. 6, 1984, pp. 619-628.

Dutta, A., and A. Basu, "An Artificial Intelligence Approach to Model Management in Decision Support Systems," *IEEE Computer*, Vol. 17, No. 9, 1984, pp. 89-97.

Elam, J., J. Henderson, and, L. Miller, "Model Management Systems : An Approach to Decision Support in Complex Organization," *In Proceedings of the First Conference on Information Systems*, Chicago, IL:Society for Management Information Systems, December 1980, pp. 98-110.

Fedorowicz, J., and G.B. Williams, "Representing Modeling Knowledge in an Intelligent Decision Support System," *Decision Support Systems*, Vol. 2, No. 1, 1986, pp. 3-14.

Fourer, R., D.M. Gay, B.W. Kernighan, "A Modeling Language for Mathematical Programming," *Management Science*, Vol. 36, No. 5, 1990, pp. 519-554.

Geoffrion, A.M., "An Introduction to Structured Modeling," *Management Science*, Vol. 33, No. 5, 1987, pp. 547-588.

Geoffrion, A.M., "The SML Language for Structured Modeling," *Operations Research*, Vol. 40, No. 1, 1992, pp. 38-75.

Glover, F., D. Klingman, and N.V. Phillips, "Netform Modeling and Applications," *Interfaces*, Vol. 20, No. 4, 1990, pp. 7-28.

Haveryly Systems Inc., *OMNI Linear Programming System : User and Operating Manual*, Denville, NJ, 1976.

Hong,S.N., M.V. Mannino, and B.S. Greenberg, "Measurement Theoretic Representation of Large, Diverse Model Bases," *Decision Support Systems*, Vol. 10, 1993, pp. 319-340.

Huh, S.Y., "An Object-Oriented Model Management Framework for Decision Support Systems," *Ph.D. Dissertation*, Anderson Graduate School of Management, UCLA, 1992.

Huh, S.Y., "Modelbase Construction with Object-oriented Constructs," *Decision Science*, Vol. 24, No. 2, 1993, pp. 409-434.

Jones, C., "An Introduction to Graph-based Modeling Systems, Part I: Overview," *ORSA Journal on Computing*, Vol. 2, 1990, pp. 136-151.

Jones, C., "An Introduction to Graph-based Modeling Systems, Part II: Graph-grammars and the Implementation," *ORSA Journal on Computing*, Vol. 3, 1991, pp. 180-206.

Krishnan, R., "Automated Model Construction: A Logic Based Approach," *Annals of Operations Research*, Vol. 21, 1989, pp. 195-226.

Krishnan, R., "PDM: A Knowledge-based Tool for Model Construction," *Decision Support Systems*, Vol. 7, No. 2, 1991, pp. 301-314.

Lazimy, R., "Knowledge Representation and Modeling Support in Knowledge-based Systems," In: C. Batini(Ed.), *Entity-Relationship Approach*, Elsevior Science Pub., North-Holland, 1989, pp. 133-161.

Lenard, M.L., "An Object-oriented Approach to Model Management," *Decision Support Systems*, Vol. 9, No. 1, 1993, pp. 67-73.

Liang, T.P., "Development of a Knowledge-based Model Management System," *Operations Research*, Vol. 36. No. 6, 1988, pp. 849-863.

Liang, T.P., and C. Jones, "Meta-Design Considerations in Developing Model Management Systems," *Decision Sciences*, Vol. 19, No. 1, 1988, pp. 72-92.

Liu, J.I.C., D.Y.Y Yun, and G. Klein, "An Agent for Intelligent Model Management," *Journal of Management Information System*, Vol. 7, 1990, pp. 101-122.

Ma, P.C., F.H.. Murphy, and E.A. Stohr, "Representing Knowledge about Linear Programming Formulation," *Annals of Operations Research*, Vol. 21, 1989, pp. 149-172.

MathPro, Inc., *MathPro Usage Guide, Introduction and Reference*, Washington, DC, 1990

Muhanna, W.A., "On the Organization of Large Shared Model Bases," in: B. Shetty(ed.), *Annanls of Operations Research, Model Management Systems*, Vol. 38, 1992, pp. 359-396.

Muhanna, W.A., "An Object-oriented Framework for Model Management and DSS Development," *Decision Support Systems*, Vol. 9, No. 2., 1993, pp. 217-229.

Muhanna, W.A., "SYMMS: A Model Management System That Supports Model Reuse, Sharing, and Integration," *European Journal of Operational Research*, Vol. 72, 1994, pp. 214-243.

Murphy, F.H., and E.A. Stohr, "An Intelligent Systems for Formulating Linear Programming," *Decision Support Systems*, Vol. 2, No. 1, 1986, pp. 39-47.

Potter, W.D., T.A. Byrd, J.A. Miller, and K.J. Kochut, "Extending Decision Support Systems: The Integration of Data, Knowledge, and Model Management," in: B. Shetty (ed.), *Annals of Operations Research, Model Management Systems*, Vol. 38, 1992, pp. 501-527.

Pracht, W.E., "An Object-Oriented Approach for Business Problem Modeling," in: M.G. Singh, K.S. Hindi, D. Salassa (ed.), *Managerial Decision Support Systems*, Elsevier Science Publishers B.V., North-Holland, 1988, pp. 143-154.

Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, Object-O*riented Modeling and Design*, Prentice-Hall Inc., Englewood Cliffs, NJ, 1991.

Schrage, L., *Linear, Integer and Quadratic Programming with LINDO*, The Scientific Press, Redwood City, CA, 1987.

Shaw, M., "Abstraction Techniques in Modern Programming Languages," *IEEE Software*, Vol. 1, No. 4, 1984, p. 10.

Sivasanakaran, T., and M. Jarke, "Logic-based Formula Management Strategies in Actuarial Consulting System," *Decision Support Systems*, Vol. 1, 1985, pp. 251-262.

Will, H.J., "Model Management Systems," in: (eds.) E. Grochla, and N. Szyperski, *Information Systems and Organization Structure*, Walter De Gruyter, Berlin, 1975, pp. 468-482.

Abstract

An Object-Oriented Model Base Design Using an Object Modeling Techniques

Jeong, Dae-yul

Recently, object-oriented concepts and technology are on the leading edge of programming language and database systems research, and their usefulness in those contexts has been successfully demonstrated. The adoption of object-oriented concept to the design of model bases has several benefits. From the perspectives of object-oriented approach, models in a model base are viewed as object which encapsulate their states and behaviors.

This paper focuses on the design of an object-oriented model base that handles various resources of DSS(data, knowledge, models, solvers) in a unified fashion. For the design of a model base, we adopted Object Modeling Techniques(OMT). An object model of OMT can be used for the conceptual design of an overall model base schema.

The object model of OMT provides several advantages over the conventional approaches in model base design. The main advantage are model reuse, hierarchical model construction, model sharing, meta-modeling, and unified model object management.