# 구조적 모델링을 위한 객체지향적 모델베이스 조직화

정 대 율[1)]

## Ⅰ. 연구동기

모델링의 영역은 매우 넓고, 다양한 패러다임이 존재하기 때문에 각 영역마다 다양한 형태의 모델이 개발되고 사용되어 왔으며, 이를 관리할 수 있는 매카니즘이나 컴퓨터지원 도구의 개발이 중요시 되고 있다. 이 중 의사결정지원을 위한 구조적 모델링(Structured Modeling : SM)을 지원할 수 있는 모델관리시스템(Model Management Systems : MMS)의 연구가 활발히 진행중이다.

Geoffrion(1987)의 구조적 모델링은 다양한 형태의 의사결정모델을 표현하기 위한 강력한 이론적 프레임웍을 제공해 준다. 구조적 모델링의 공식적 프레임웍은 이산수학에 근거를 두고 있으며, 모델을 표현하기 위해서 계층적으로 조직화되고 분할된 유속성 비순환 그래프(attributed acyclic graph)를 사용한다.

이러한 구조적 모델링을 지원하기 위한 많은 모델관리시스템들이 구현되었다. 구조적 모델링의 아이디어를 최초로 구현한 것은 LEXICON(Clemence, Jr. 1984)일 것이다. 이것은 메인프레임에서 작동할 수 있도록 FORTRAN으로 구현하였으며, 대규모의 LP문제의 최적화를 지원해주는 데 주된 목적을 두었다. 모델스키마와 데이타는 벳취파일 형태로 입력된다. Farn(1985)은 혼합형 정보/분석적 모델링시스템의 가능성을 타진하기 위해서 Micro Data Base System에서 개발한 KnowledgeMan을 이용하여 IIS(Integrated Information System)를 개발했다. 이것은 또한 LP문제를 위해서 LINDO와 인터페이스 할 수 있다. LEXICON과 IIS는 구조적 모델링 개념이 발전하는 과정에서 개발되었으며, 극히 한정된 목적만을 가지며 구조적 모델링의 일부분만을 지원하고 있다.

구조적 모델링 개념을 전적으로 지원할 수 있는 시스템으로는 Geoffrion(1991)이 개발한 FW/SM과 이를 발전시킨 Neustadter 등(1992)의 연구일 것이다. 이것은 통합팻키지인 Framework Ⅲ로 구현한 프로토타입 시스템으로 20 여 가지의 기능을 제공해주고 있다. 이외에도 구조적 모델링의 특수한 일부분을 구현한 시스템들도 많이 있다. 이중에서 특히 주목할 것은 GBMS(Graph-Based Modeling System)를 결합한 Jones(1992)의 연구, 논리 시스템과의 결합을 시도한 Chari & krishnan(1993)의 연구, 그리고 관계형 DB와의 결합을 연구한 Lenard(1986)의 연구와 객체지향적 데이타베이스관리시스템(ODBMS)와의 결합을 시도한 허순영(1994)의 연구 등을 들 수 있다. 또한 Dolk(1988)는 IRDS (Information Resource Dictionary System)의 모델표현법으로 구조적 모델링을 채택하였다. 이외에도 많은 연구들이 구조적 모델링 개념을 채택하거나 언급하고 있다.

---

1) 동명전문대학 경영정보과

이들 연구들은 모두 구조적 모델링의 개념중 일부분 또는 대부분을 실현할 수 있는 모델관리시스템의 기능구현에 중점을 두었으며, 구조적 모델링지원을 위한 모델관리시스템의 모델베이스 조직화와 설계를 위한 방법에 대한 연구는 없었다.

대규모의 모델베이스를 효과적으로 조직화하기 위해서는 메타모델링 개념의 도입이 필요하다. 모델을 조직내의 자원으로 본다면, 이들 자원을 전체적으로 조직화하여 모델베이스 내에 저장하므로써 비중복성, 일치성, 무결성 등을 향상시킬 수 있다. 메타모델링개념은 조직 전체적인 입장에서의 개념적 수준에서의 모델베이스 조직화를 용이하게 하므로써 조직의 모델링 자원을 효과적으로 활용할 수 있게 한다.

이러한 문제를 해결하기 위해서 객체지향개념을 구조적 모델링을 위한 모델베이스 조직화에 도입할 수 있다. 본 연구에서는 Rumbaugh 등(1991)의 객체모델링기법(OMT: Object Modeling Techenique)을 모델베이스 조직화를 위한 기본적인 방법으로 사용한다. OMT를 구조적 모델링을 위한 모델베이스 조직화에 도입할 경우 구조적 모델링을 구성하는 여러 구성요소들 간의 관계를 쉽게 파악할 수 있다. 즉, OMT의 객체다이어그램을 이용하여 구조적 모델을 구성하는 여러 가지 객체타입들이 어떻게 모델베이스에 조직화될 수 있는가를 보일 수 있다.

또한 본 연구에서는 구조적 모델링을 위한 모델베이스를 구성하는 객체들을 설계하고자 하며, 구조적 모델링과 모델관리에 관한 여러 가지의 메타지식들(무결성 보증을 위한 규칙들 포함)을 도출하고자 한다.

# II. 구조적 모델링의 개요

구조적 모델링(SM)은 모델의 기본적인 요소와, 이들 요소들 간의 관계를 식별하며, 모델이 "구조적"이라고 할 수 있는 조건을 명기한다. SM은 모델스키마(모델 클래스)를 표현하기 위한 표기법을 제공하며, 세부적인 것을 포착하기 위해서 데이타 테이블을 기술하므로써 모델스키마를 특정 모델 인스턴스로 전환하도록 한다.

## 2.1 구조적 모델의 구조

구조적 모델링은 세 가지 수준의 구조, 즉 요소적 구조(elemental structure), 일반적 구조(generic structure), 모듈러 구조(modular structure)를 갖는다. 이들 세 가지 수준의 구조는 모델의 개념적 수준을 표현하는 개념 스키마로 볼 수 있다.

구조적 모델링은 모델을 이산 요소들(discrete elements)로 구성된 것으로 본다. 요소적 구조는 특정 모델인스턴스의 정의적 세부 사항을 표현하는 것으로 다음의 다섯 종류의 요소로 구성된다.

● 원시 실체(primitive entity : pe): 수학적으로 정의되지 않는 요소로써 관련된 값을 갖지 않으며, 모델에서 고유하게 식별될 수 있는 실체(사람, 장소, 사물, 행동, 사건, 질, 상태 등)이다. 즉, 모델 수립자가 축소 불가능하거나 분석할 수 없는 사물을 표현하는 데

있어 가정 없이 자의적으로 도입되는 실체이다.

● 복합 실체(compound entity : ce): 이미 정의된 원시 실체 요소나 또는 다른 복합 실체 요소들을 참조하여 정의된 실체(사물이나 개념)로써 관련된 값을 지니지 않는다. 복합 실체 요소는 이산 수학에서는 집합이나 릴레이션의 한 멤버로 표현되어질 수 있다.

● 속성(attribute : a): 실체(사물이나 개념)의 특성을 나타내는 것으로 일정한 값을 갖는다. 보통 모델의 "계수(파라메터)"나 의사결정"변수"들이 구조적 모델링에서는 속성에 해당한다. 모델에서 상수나 계수와 같이 그 값이 미리 정해진 속성을 고정 속성(fixed attribute : fa)이라 하며, 의사결정변수와 같이 그 값이 미리 정해져 있지 않는 속성을 변수 속성(variable attirbute : va)이라 한다. 그런데 구조적 모델링에서는 "계수"와 "변수"를 굳이 구분하지 않는다. 왜냐하면 이러한 구분은 대부분의 모델 수명 주기 단계(통계적 추정, 민감도분석, "What-if" 분석 등을 생각해 볼 때)에서 고정된 것이 아니기 때문이다.

● 함수(function : f): 호출된 요소의 값에 대한 명기된 규칙에 따라 종속된 값을 지니는 것으로, 보통 계산 가능한 특성과 모델의 보다 더 복잡한 측면을 나타낸다.

● 테스트(test : t): 함수와 같으나 참 또는 거짓의 두 값만을 가진다.

요소적 구조는 이들 다섯 가지 요소(nodes)와 이들 간의 정의적 참조를 나타내는 가지(arc)로 구성된 방향 그래프(요소 그래프)로 볼 수 있다. 모델요소들간의 정의적 상호의존관계(definitional interdependencies)가 구조적 모델링의 초점이며, 이들 의존관계를 나타내는 데 호출순서가 이용된다. 만일 요소 A가 요소 B의 호출순서 상에 있다면 요소 B(호출자)는 요소 A(피호출자)를 호출한다고 한다.

일반적 구조는 요소적 구조에서 유사한 요소들을 묶어 "Genus graph"로 표현한 것으로 문제 클래스를 나타낸다. 이것은 모델의 모든 요소들을 유사한 타입을 지닌 것끼리 묶어 "genera"로 만든다. 따라서 각 "genus"는 동일한 타입의 요소들로 구성된 세그멘트이며, 앞에서 제시한 5가지 타입 중의 하나가 된다. 일반적 구조는 모든 genus에(원시 실체 genera는 제외) 대해서 다음이 만족될 경우 일반적 유사성(generic similarity)이 만족된다 : genus내의 모든 요소들이 동일한 수의 호출순서 세그멘트를 가지고, 그 세그멘트 내의 모든 호출자들이 동일한 genus내의 요소들이고, 그리고 모든 요소들이 호출순서 세그멘트에 대응하여 동일한 genera를 호출한다.

모듈러 구조는 일반적 구조 내의 관련 구조체를 상위 객체(모듈)로 그룹화 하여 일반적 구조를 계층적으로 조직화한 모듈러 트리(modular tree)로 나타낸다. 모듈러 구조는 상위 수준의 추상화를 통하여 모델의 복잡성을 관리하도록 한다. 모듈러 구조는 나무 형태(rooted tree)로써 그 시발점(root)은 전체 모델을 나타내며, 끝 마디는 genera와 1:1로 대응된다. 그리고 모듈러 구조는 단조 계층(monotone ordering)을 지닌다. Geoffrion(1989)은 단조 계층을 다음과 같이 정의하고 있다.

"유사성을 만족시키는 일반 구조에 대해 정의된 모듈러 구조의 단조 계층은 각 근방 집합(sibling set)에 대한 계층(order)으로 명세되어진다. 이러한 계층들은 루트(root)를 제외한 모든 마디에 걸쳐서 엄격한 부분 계층(strict partial order)을 얻기 위해서 일반적인 방법으로 확장되어진다. 여기서 두 마디가 다른 마디의 루트 경로(rootpath)상에 놓여있지 않다면 이들은 비교되어질 수 있다. 이러한 엄격한 부분 계층은 다음과 같은 점에서

단조(monotone)이다 : 만약 genus B가 genus A를 호출한다면, 엄격한 부분 계층에서 A가 B 앞에 온다."

이상에서 제시한 세 가지 모델 구조에 근거하여 Geoffrion(1989)은 구조적 모델(structured model)을 다음과 같이 정의하고 있다. "일반적 유사성을 만족시키는 일반적 구조와 단조 계층을 지닌 모듈러 구조를 함께 가진 요소 구조이다."

## 2.2 모델클래스와 인스턴스

모델을 이론적으로나 실제적으로 적용하는 데 있어 특정한 데이타를 지닌 하나의 모델 인스턴스 뿐만 아니라, 특성이 유사한 모델 인스턴스들을 묶은 모델 클래스가 유용한 경우가 많다. 즉, 대부분의 모델 사용에 있어 하나의 특정 모델 인스턴스를 기술하는 데 필요한 데이타보다는 모델의 일반적인 형태에 더 많은 초점을 두고 있다.

구조적 모델링에서 모델 스키마(model schema)는 다음과 같은 면에서 동형성(isomorphism)을 만족시키는 구조적 모델들의 클래스이다 : 클래스 내에 두 모델이 주어진 경우 그 모듈과 genera가 다음과 같이 1:1 대응될 수 있어야 한다 : ① 모듈러 구조 트리에서 마디 근방(node adjacency)이 지켜지며, ② 상응하는 genera는 동일한 수의 호출순서 세그멘트를 가지며 각 세그멘트로부터 상응하는 genera를 호출하며, ③ 상응하는 genera가 동일한 요소 타입을 지녀야 한다.

보통 구조적 모델은 항상 완전히 명세되지는 않는다. Geoffrion(1989)은 완전하게 명세된 구조적 모델(completely specified structured model)과 "A-partially specified structured model"을 다음과 같이 정의하고 있다. "완전하게 명세된 구조적 모델은 모든 호출순서, 속성 값, 함수와 테스트 요소의 규칙을 포함한 모든 요소들을 상세히 명세하며, 일반적 구조를 만족키는 명세와, 단조계층으로된 모듈러 구조를 명세해야 한다. 그렇지 않으면 구조적 모델은 불완전하게 명세 되었다고 한다. 완전하게 명세된 요소적 구조는 명백한 정의를 지닌다." 임의적인 값을 갖는 속성 요소들을 변수 속성(variable attributes)으로 지정될 수 있다. 이 변수 속성들은 변동적이거나 해산출기의 통제를 받는다. "A-partially specified structured model은 변수 속성 요소의 값을 제외한 모든 것들이 완전하게 명세된 모델을 말한다." 변수 속성 요소 값은 여러 전통적 모델에서 사용되는 "변수"와 동일한 역할을 한다.

실제로 모델이 처음 수립될 때는 불완전하게 명세 되어졌다가 세부적인 사항들을 정해지고 자료가 수집되므로써 모델의 정도는 점차 향상되어 의도한 목적에 맞는 최종 수준(complete or A-partially specified structured model)에 이르게 된다. 따라서 최종 수준의 명세가 얻어지기까지는 여러 번의 모델 예(case)들이 만들어지게 되는데 이것들이 모두 동일한 모델의 상이한 명세들이 된다.

평가(evalution)는 요소적 구조의 함수와 테스트 요소의 값을 결정짓는 작업이다. 요소적 구조는 비순환성(acyclicity)을 지니므로, 평가는 요소적 구조의 위상적 분류순서에 따라 한번에 이루어질 수 있다. 구조적 모델 그 자체만으로는 평가를 수행하는 방법을 제공하지 못하며, 모델 외부의 어떤 매카니즘을 필요로 한다. 이러한 평가 매카니즘이 구조적 모델링 시스템(모델관리시스템)에 통합되어 있는 것이 이상적이다.

## 2.3 모델 뷰와 모델그래프

모듈러 구조의 뷰(view)는 원래의 루트(root)를 유지한 채 원래의 siblings를 나누지 않은 서브 트리(subtree: a subgraph of the original rooted tree that is also a rooted tree)이다. 즉, 모듈러 구조를 나타내는 원래의 트리에서 두 개의 마디가 동일한 부모를 가졌다면, 이것들은 그 서브 트리 내에 있거나 또는 밖에 있다. 이 때 원래의 트리를 마스터 뷰(master view)라 한다. 따라서 "마스터 뷰"는 "마스터 개념적 구조(master conceptual structure)"가 되며, "뷰"는 "개념적 구조"가 된다. 그리고 "서브트리의 마디"는 "개념적 단위(conceptual unit)"가 된다. 여기서 "뷰"는 데이타베이스관리시스템에서 사용되는 "뷰"와는 약간 다른 의미를 지닌다.

유사성을 만족시키는 일반적 구조에 대해서 정의된 모듈러 구조가 주어진 경우, 각 분할된 셀(cell)의 부분집합이 호출 순서 {C1,C2, ...., Cn}(즉, C1 내의 어느 genus가 C2 내의 어떤 genus를 호출하고, ····· , Cn-1 내의 어떤 genus가 Cn 내의 한 genus를 호출함)에서 Cn = C1(n>1)이 되도록 하는 순환이 일어나지 않는다면 뷰는 비순환성을 유지한다(acyclicity- preserving)고 한다. 모든 가능한 뷰가 비순환성을 유지한다면, 모듈러 구조 그 자체도 비순환성을 유지한다.

구조적 모델링에서는 모델의 가시성과 추상성을 높이기 위해서 그래프를 사용한다. 앞에서 제시한 세 가지 모델구조에 대해서 각각 요소 그래프(element graph), Genus 그래프, 모듈 그래프(module graph)가 사용된다.

요소적 구조의 요소그래프는 모든 요소를 위한 마디와 요소 A에서 요소 B로의 가지(요소 A가 B를 호출할 경우)를 가진 유속성 방향그래프(attributed directed graph)이다. 모든 마디는 그것의 타입(원시 실체, 복합 실체, 속성, 함수, 테스트)을 나타내는 하나의 속성을 지닌다. 모든 비실체마디(non-entity node)는 그 값을 부여하는 다른 속성을 가지며, 모든 속성 마디는 그 범위를 부여하는 다른 속성을 가지며, 그리고 모든 함수와 테스트 마디는 그 규칙을 부여하는 속성을 가진다. 모든 가지는 호출 순서와 세그멘트 내의 위치를 식별하는 두 개의 속성을 가진다.

유사성을 만족시키는 일반적 구조의 Genus 그래프는 모든 genus를 위한 마디와 모든 genus의 세그멘트를 위한 가지(피호출 genus로부터 호출 genus로 향함(원시 실체 genera를 제외))를 가진 방향 그래프이다. 따라서 genus graph는 genera 간의 상호 참조를 나타내므로 요소 그래프보다는 단순 명료하여 다루기 쉽다.

모듈러구조의 뷰에 대응되는 모듈 그래프는 연관성 있는 genus 분할의 모든 셀(cell)을 위한 마디와 셀 A에서 셀 B로 향하는 가지(셀 B의 genus가 셀 A의 genus를 호출함)를 가진 방향 그래프이다. 모듈 그래프는 뷰의 가장 작은 개념적 단위들 간의 상호 참조를 나타낸다.

## 2.4 구조적 모델링언어

구조적 모델링의 이러한 그래프적인 구조는 사용자가 수리적 복잡성에서 벗어나 문제 요소와 이들 요소들 간의 관계에 주목하도록 해준다. 그래프는 단지 효과적인 의사소통

목적이지 곧바로 컴퓨터 실행가능한 표현법은 아니다. 실질적인 실행을 위해서 구조적 모델링은 텍스트 중심의 모델스키마(text-based model schema)를 제공해 준다. 이러한 논리적 스키마 표현을 위한 모델링 언어가 바로 SML(Structured Modeling Language)이다.

SML은 모델 명세를 위한 문법(syntax)를 제공해줄 뿐만 아니라 오류 발견(error-trapping), 자동적 문서화, 해산출기 인터페이스, 그리고 데이타 편집 등 여러 가지 기능을 갖고 있다.

SML 스키마는 여러 문장으로 구성되며, 각 문장은 모델명세를 나타내는 공식적 부분(formal part)과 이에 대응하는 모델문서화를 제공하는 비공식적 부분(informal part)으로 구성된다. 그리고 각 요소 타입은 약어를 사용하여 양쪽에 슬래스를 친다. 예를 들어 수송문제를 SML로 나타내면 (그림 2-1)과 같다.

```
&SDATA   SOURCE DATA
  PLANTi /pe/                    There is a list of PLANTS.
  SUP(PLANTi) /a/ {PLANT}:R+       Every PLANT has a non-negativity SUPPLY
CAPACITY
                                 measured in tons.
&CDATA   CUSTOMER DATA
  CUSTj /pe/                     There is a list of CUSTOMERS.
  DEM(CUSTj) /a/ {CUST}:R+           Every CUSTOMER has a non-negativity
DEMAND
                                 measured in tons.
&TDATA   TRANSPORTATION DATA
  LINK(PLANTi,CUSTj) /ce/            There are some transportation LINKS from
PLANTS
                                 to CUSTOMERS. There must be at least one LINK
                                 incident to each PLANT, and at least one LINK
                                 incident to each CUSTOMER.
  FLOW(LINKij) /va/ {LINK}:R+        There can be a nonnegative transportation
FLOW
                                 (in tons) over each LINK.
  COST(LINKij) /a/ {LINK}:R+       Every LINK has a TRANSPORTATION COST
RATE
                                 associated with all FLOWS.
TOTAL_COST(COST,FLOW) /f/; SUMij(COSTij * FLOWij)
                                 There is a TOTAL COST associated with all
FLOWS.
T:SUP(FLOWi, SUPi) /t/ {PLANT}; SUMj(FLOWij) <= SUPi
                                 Is the total FLOW leaving a PLANT less than or
                                 equal to its SUPPLY CAPACITY?
                                 This is called the SUPPLY TEST
T:DEM(FLOWj, DEMj) /t/ {CUST}; SUMi(FLOWij) >= DEMj
                                Is the total FLOW arriving at a CUSTOMER
exactly
                                equal to its DEMAND?
                                This is called the DEMAND TEST.
```

(그림 2-1)수송문제의 SML 스키마

모델의 인스턴스를 위해 사용되는 데이타는 관계형 데이타 테이블(elemental detail tables) 형태로 따로 저장되어 있어 데이타독립성이 보장된다.

이상에서 제시된 구조적 모델링의 구조적 모델과 SML의 특성을 이해하기 위해서는 앞에서 제시한 여러 가지의 개념적 프레임웍을 이해해야 하며, 그 특성을 요약하면 다음과 같다.

구조적 모델링은 ① 하나의 단일화된 엄격한 형식(formalism) 내에 다양한 형태의 모델과 모델링 패러다임을 다룰 수 있다. ② 어떠한 모델에 대해서도 손쉽게 의사소통할 수 있는 다이어그램을 제공한다. ③ 모델부분들 간의 상호관계를 중요시 하고 있다. 이것은 대부분의 현행 모델링 형식들이 간과하고 있다. ④ 해산출기를 모델변경에서 독립시키고 있다. ⑤ 복잡성을 관리하기 위한 접근법으로 계층적 조직화를 이용한다. ⑥ 모델을 사용자가 직접이해할 수 있으면서 곧바로 컴퓨터 상에 실행가능하도록 한다. ⑦ 데이타 관리를 위해서 관계형 DB 툴을 쉽게 이용할 수 있도록 한다.

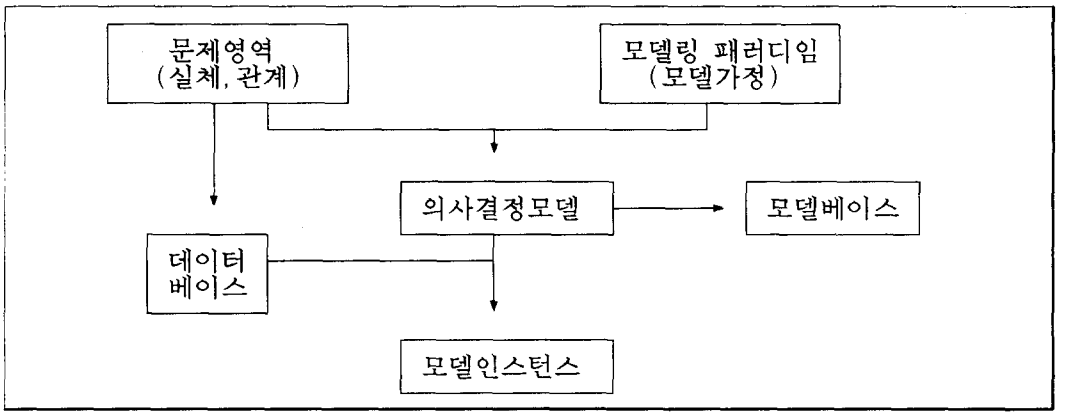
# III. 객체지향 모델베이스

## 3.1 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 3-1). 따라서 의사결정모델이란 현실 세계에 존재하는 시스템을 공식적 언어로 표현한 추상화된 객체이다. 모델화의 대상이 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, AMPL, LINDO, SML 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,

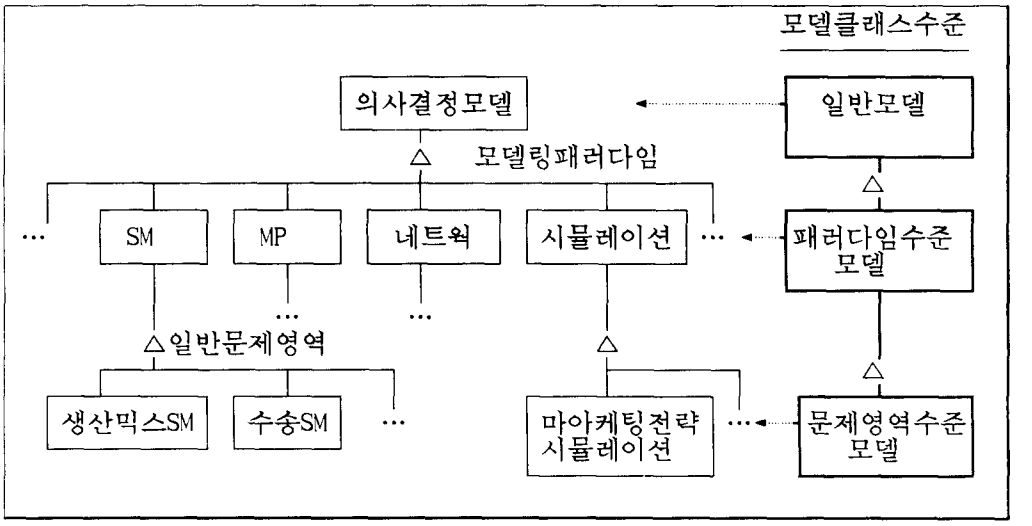



(그림 3-1) 모델링의 세계

이것 역시 타입화 할 필요가 있다. 예를 들면 선형계획 모델링 패러디임에서는 문제영역 내의 실체와 이들 간의 관계를 선형대수관계로 표현하므로 LP모델 클래스가 만들어진다.

### 3.2 추상화 수준에 따른 모델베이스

모델베이스를 구성하는 모델클래스는 그 추상화 수준에 따라 패러디임 수준과 문제영역 수준으로 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 3-2)는 그 추상화 수준에 따라 모델베이스 내에 존재하는 모델클래스들을 분류한 것이며 이들 간의 계승관계를 보여주고 있다.




(그림 3-2) 추상화 수준에서 본 모델클래스

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, 모델수립자는 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.

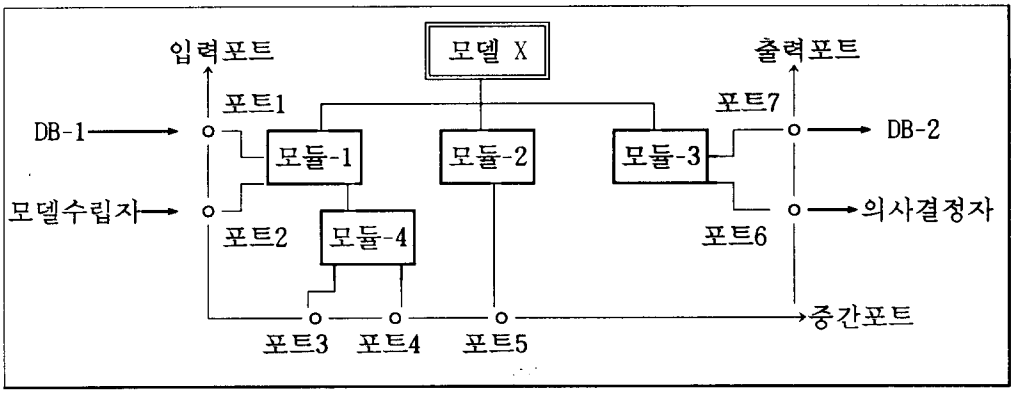
### 3.3 일반모델의 구성요소

일반모델은 현실 세계의 문제에 대한 추상적 표현으로써 모든 모델클래스에 대하여 특성을 계승시킬 수 있는 모델클래스이다. 일반모델에 대한 객체타입이 일반모델타입이다. 모델스키마는 그 관점에 따라 외부 명세와 내부 명세로 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 특징 지워진다. 따라서 일반모델타입은 포트(port)라는 연결점들로써 그 기능과 가정을 기술한다. 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 타입화 된다. 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 통하




(그림 3-3) 일반모델의 개념적 구조

여 그 추상화 수준(집합수준)을 높일 수 있다. 일반모델의 개념을 도식하면 (그림 3-3)과 같다.

### 3.4 개념적 스키마와 논리적 스키마

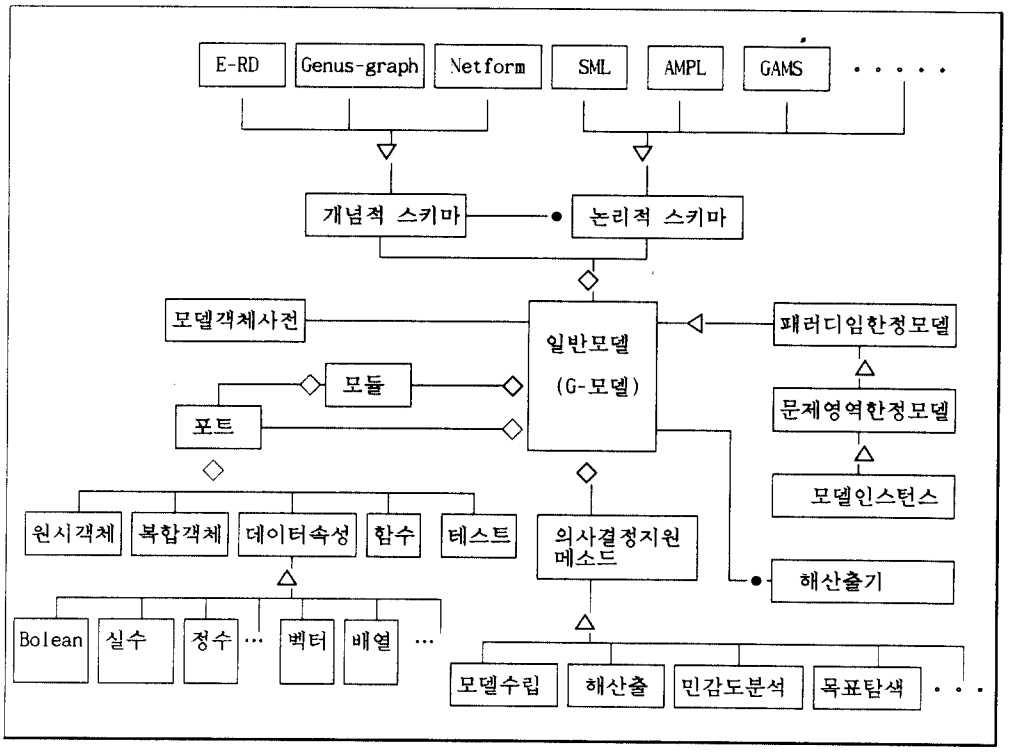
모델스키마는 특정 모델링시스템의 문법적 요구와는 독립적인 개념적 스키마(conceptual schema)와 특정 모델링시스템의 문법적 요구에 따라 모델을 표현한 논리적 스키마(logical schema)로 나눌 수 있다. 개념적 스키마는 모델수립자 또는 사용자의 관점을 나타내며, 논리적 스키마는 컴퓨터 시스템의 실행 관점을 나타낸다. 따라서 모델스키마 역시 타입화할 수 있다. 모델스키마타입은 개념스키마타입(conceptual_schema_type)과 논리스키마

타입(logical_schema_type)으로 세분화된다. 개념스키마는 지금까지 개발된 개념적 모델(여기서는 다이어그램이나 그래프 형태에 만 한정함)을 표현하는 객체타입이며, 논리스키마타입은 컴퓨터 상에서 실행 가능한 모델링언어를 표현하기 위한 객체타입이다.

### 3.5 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-modelbase)라 한다.

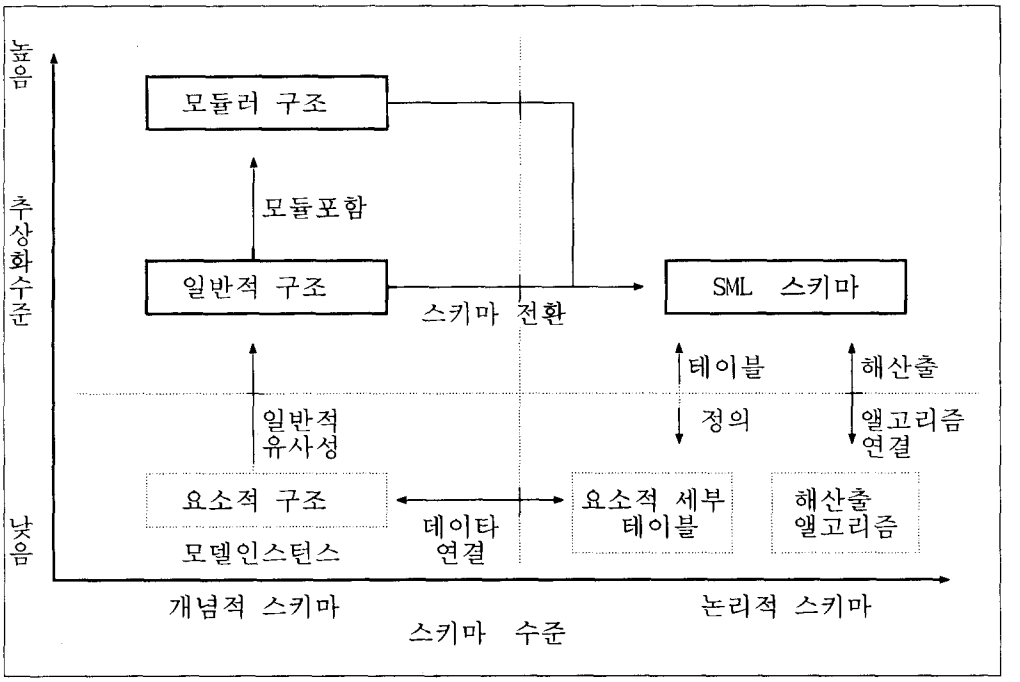
메타모델베이스는 앞에서 제시된 여러 유형의 모델타입과 이들 객체에 대한 정보를 담은 지식과 데이터들 간의 관계를 개념적으로 표현한다. 메타모델베이스 설계를 위해서 사용하는 도구는 앞에서 제시된 객체모델링기법의 객체다이어그램이다. 메타-모델베이스를




(그림 3-4) 메타-모델베이스

객체다이어그램으로 표시하면 (그림 3-4)와 같다.

모델베이스 내에는 특정 모델링 패러디임과는 독립적인 일반적 모델의 구조를 갖는 일




(그림 4-1) 구조적 모델의 구성

반모델클래스, 모델링 패러디임 수준의 모델을 나타내는 P-모델클래스(예, SM, MP모델), 특정 의사결정영역의 문제를 해결하는 데 사용되는 D-모델클래스 또는 모델템플릿(예, 정유회사의 제품믹스 SM모델) 간에는 일반화와 구체화 관계를 나타낸다.

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 모델 객체들에 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 변화에 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.

# Ⅳ. 구조적 모델링을 위한 객체지향 모델베이스의 설계

## 4.1 구조적 모델의 객체지향적 관점

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 Genus로 추상화한 일반적구조, 그리고 일반적구조를 한 단계 더 추상화하여 관련있는 Genus를 하나의 모듈로 묶어 나무형태(비순환 그래프)로 표현한 모듈러 구조를 지닌다. 그리고 이러한 개념적 스키마를 컴퓨터 상에서 곧바로 실행가능하도록 하기 위한 모델링 언어가 SML이다. SML로 표현된 모델스키마가 논리적 모델스키마이다. 그리고 모델-데이타 독립성을 보장하기 위해서 모델의 데이타는 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 도식화 하면 (그림 4-1)과 같다.

이러한 구조적 모델링을 통해 만들어진 구조적 모델은 모델구성요소에 대한 모듈러적, 계층적 특성들을 가장 잘 나타내 주므로 앞 장에서 설명한 여러 가지 모델타입의 개념과 가장 잘 대응된다. 즉, 구조적 모델의 모듈개념은 일반모델타입의 모듈개념과 일치하며, 각 Genera는 포트에 대응된다. 그리고 구조적 모델의 호출순서(calling sequence)는 각 Genera(즉, 포트) 간의 정의적 상호의존관계를 나타내므로 모델의 내부적 관점을 보여줄 수 있다. 예를 들어 (그림 2-1)에서 제시된 수송모델의 경우 각 모듈과 포트를 정의하면

(표 4-1) 수송모델의 모듈 및 포트 종류

| 타 입 | 수송모델의 모듈 또는 포트 종류 |
|---|---|
| 모 듈 | &S_DATA, &C_DATA, &T_DATA<br>TOTAL_COST, T:DEM, T:SUP |
| 입력포트 | PLANT, SUPPLY, CUST, DEMAND, COST |
| 출력포트 | TOTAL_COST, FLOW, T:SUP, T:DEM |
| 중간포트 | LINK |

(표 4-1)과 같다.

따라서 객체지향적 관점에서 구조적 모델을 볼 때, 모듈러 구조는 모듈의 계층적 집합이므로 일반모델타입의 하위타입으로 볼 수 있으며, 일반적 구조는 모델의 개념적 구조를 보여주므로 개념적 스키마의 하위타입으로 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. 이와 같이 구조적 모델을 객체지향적으로 표현하고 조직화 할 수 있다면, 객체지향

적 모델관리시스템에서 구조적 모델을 쉽게 다룰 수 있을 것이다.

### 4.2 구조적 모델링을 위한 모델베이스의 객체 클래스 조직화

구조적 모델을 릴레이션으로 보는 관점은 모델을 데이타로 간주하는 데이타베이스 지향적인 견해이다. 구조적 모델링을 지원하기 위한 관계형 데이타베이스에 사용되는 릴레이

(표 4-2) 구조적 모델링 릴레이션

| 구 분 | 릴레이션 정의 |
|---|---|
| 요소적 릴레이션 (elemental relation) | E(ElementName, Index, Ename)<br>CE(ElementName, Index, CalledElementName, Called Index)<br>AREAL(ElementName, Index, Value)<br>AINT(ElementName, Index, Value)<br>ACHAR(ElementName, Index, Value) |
| 일반적 릴레이션 (Generic Relation) | GENUS(GenusName,Type,Interp)<br>CALLS(GenusName, SeqNo, CalledGenusName)<br>RULES(GenusName,Rule)<br>DATATYPE(GenusName,Dtype) |
| 모듈러 릴레이션 (Modular Relation) | MODULE(ModName, Interp)<br>CONTENT(ModName, Contains) |

션을 정의하면 (표 4-2)와 같다(Lenard 1986). 객체지향적 관점에서 본다면 릴레이션은 하나의 객체로 간주할 수 있으며, 이 릴레이션 객체에 여러 가지의 메소드를 추가할 수 있다.

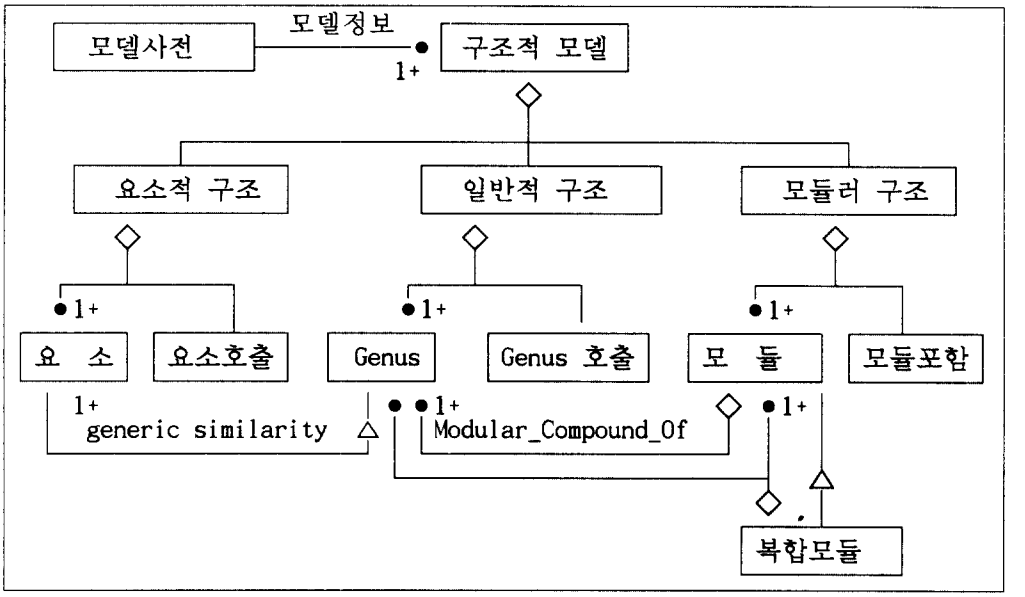
요소적 릴레이션에서 실체와 속성(실수값, 정수값, 문자값을 가지므로) 클래스의 인스턴스는 각각 E와 A(AREAL, AINT, ACHAR) 릴레이션으로 나타내고 있다. 데이타베이스를 완성하기 위해서는 모든 변수 속성과 함수를 열거하는 릴레이션도 있어야 한다. Lenard(1986)는 함수를 데이타베이스(모델베이스)에 표현하기 위해서 요인분해가능함수(factorable function) 개념을 도입하였다. 모델베이스에 요인분해가능함수를 표현하기 위해서 두 종류의 함수 규칙(function rule)인 요소 함수(elementary function)와 복합 함수(compound function)를 사용하였다. 요소 함수는 한 변수의 함수이며, 복합 함수는 둘 또는 그 이상의 함수들의 합이나 곱으로 이루어진다.

Genus 클래스의 모든 인스턴스들은 GENUS 릴레이션에 열거되어지며, 모듈 클래스의 모든 인스턴스들은 MODULE 릴레이션에 열거되어진다. 예를 들어 GENUS (PLANT, pe, A list of plants)라는 인스턴스가 있을 때, GENUS 클래스의 인스턴스 명은 PLANT(GenusName="PLANT")이며, 그 타입은 원시실체(Type="pe")이고, Interp="A list of plant"는 해석을 나타낸다.

또한 CE, CALLS, CONTENT 릴레이션은 사전적 의미를 담은 릴레이션으로, 각각 요소적 호출순서(Elemental Calling Sequence), 일반적 호출순서(Generic Calling Sequence), 모듈의 구성요소(Modular Contents)를 나타낸다. 그리고 DATATYPE 릴레이션은 속성

(a)과 함수(f) genera에 대한 인스턴스 값에 대한 데이타 타입을 나타내며, RULES 릴레이션은 함수(f) genera에 대한 함수규칙은 나타내는데 사용한다.

객체지향적 관점에서 구조적 모델링의 구조적 모델을 구성하는 객체 클래스와 이들 간




(그림 4-2) 구조적 모델의 객체다이어그램

의 관계를 객체다이어그램으로 나타내면 (그림 4-2)와 같다. 구조적 모델은 요소적 구조와 일반적 구조, 모듈러 구조를 갖는 모델이다. 요소적 구조는 하나 이상의 요소(객체다이어그램에서 ●는 연관관계의 다중성을 나타내며, 1+은 연관관계가 하나 이상임을 표시함)와 이들 간의 의존관계를 나타내는 요소호출로 구성된다.

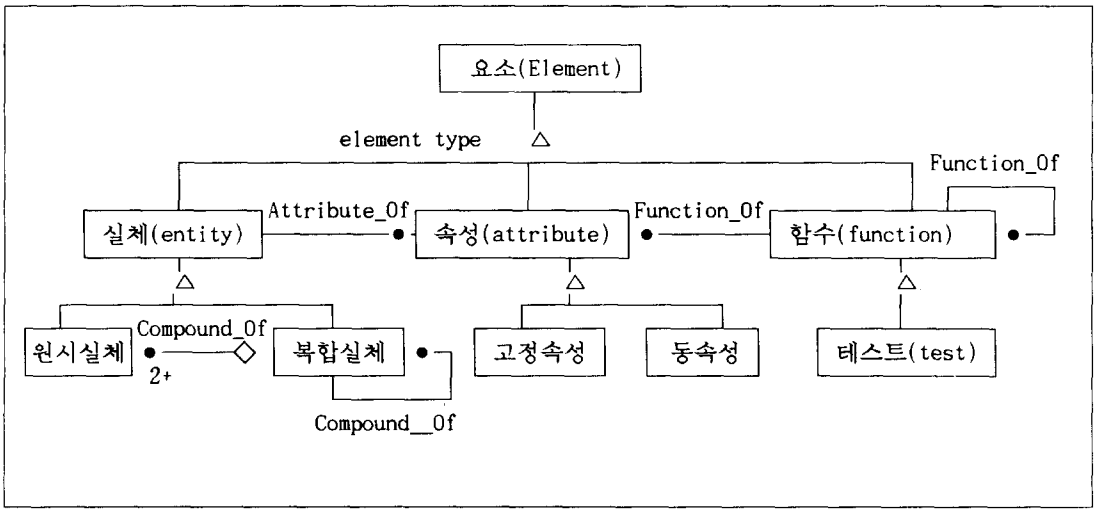
일반적 구조는 하나 이상의 Genus와 이들 간의 Genus 호출로 구성되며, 모듈러 구조는 하나 이상의 모듈 또는 복합모듈과 이들 간의 모듈포함관계로 구성된다. 또한 Genus는 동일한 특성을 갖는 요소들로 구성되며, 모듈은 하나 이상의 Genus로 구성된다. 복합모듈은 하나 이상의 모듈들로 구성되거나 또는 모듈과 Genus로 구성되며, 모듈의 속성을 이어받는다.

(그림 4-2)의 모델사전객체는 모델에 대한 기본적인 정보를 담은 사전으로 모델상담시스템에서 사용되는 객체이다. 또한 요소호출, Genus 호출, 모듈 포함 객체는 모델의 내부구조에 대한 정보를 담은 객체이다.

한편 그래프 중심의 모델링시스템의 입장에서 본다면, 요소 클래스, Genus 클래스, 모듈 클래스의 인스턴스는 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 클래스의 인스턴스는 가지(arc)로 표시되므로 가지 클래스(arc class)로 일반화될 수 있다.

구조적 모델링에서 구조적 모델을 구성하는 기본적인 모델이 요소적 구조이며, 요소적

구조의 기본적인 객체는 요소(element)이다. 요소객체는 그 타입에 따라 실체, 속성, 함수로 세분화할 수 있으며, 실체는 다시 원시실체와 복합실체로 세분화된다. 그리고 속성은 그 값의 변동성여부에 따라 고정속성과 변동속성으로 세분화될 수 있다. 또한 함수의 특수한 형태로서 테스트가 있다. 테스트는 함수의 하위클래스로 값이 항상 참(True) 또는 거짓(False) 둘 중의 하나를 갖는다는 특성을 제외하고는 함수의 일반적인 특성을 계승받는다. 따라서 이들 클래스들 간에는 일반화(generalization)관계가 성립하여 하위클래스가 상위클래스의 특성을 계승받는다. 이들 요소객체를 구성하는 객체들 간의 관계를 객체다




(그림 4-3) 요소 클래스의 객체다이어그램

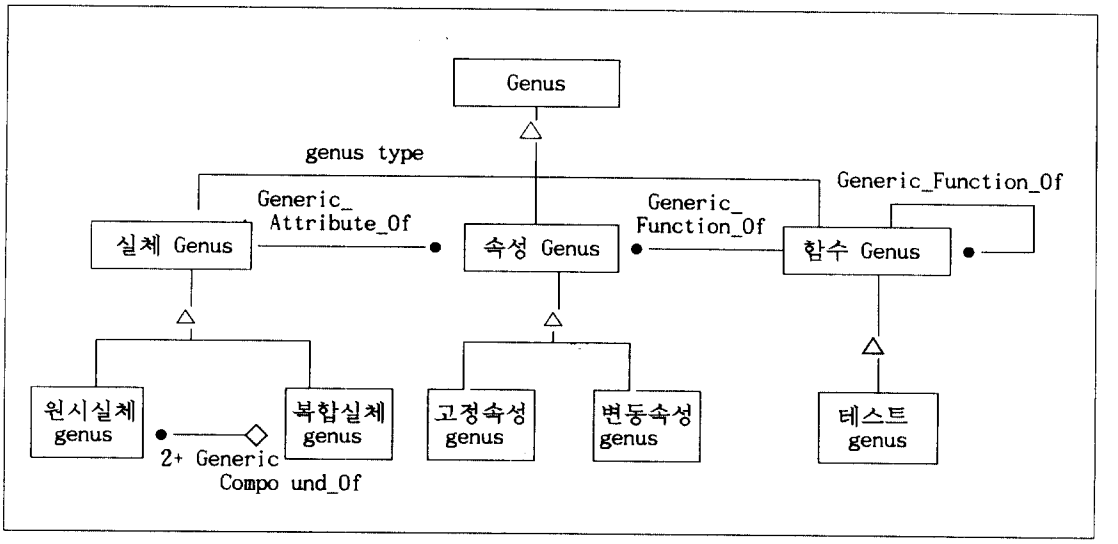
이어그램으로 나타내면 (그림 4-3)과 같다.

그리고 이들 요소를 구성하는 객체들 간의 연관관계는 다음과 같다. 복합실체는 둘 이상의 원시실체들로서 구성되거나 또는 또 다른 복합실체들로써 구성되므로 이들 객체사이의 관계는 "CompoudOf" 관계(이것의 역관계는 "ComponentOf" 임)로 규정할 수 있다. 그리고 실체객체는 여러 속성을 가지므로 이들 간에는 "AttributeOf"(역관계는 "IndexOf")가 있다. 또한 함수는 여러 개의 인수(속성)를 가지므로 "FunctionOf"(역관계는 "ArgumentOf")관계로 나타낼 수 있다.

이러한 요소들 간의 의존관계를 나타내는 것이 (그림 4-2)의 요소호출객체이며, 이는 구조적 모델의 요소적 구조를 구성하는 각 요소들 간의 호출순서를 나타낸다. 객체지향적 관점에서 본다면, 요소호출순서는 모델에서 사전(dictionary)이나 또는 요소들 간의 상호의 의존관계를 나타내는 클래스이다. 예를 들어 LINK-AB가 수송모델에서 PLANT-A에서 CUST-B로의 수송 LINK를 나타내는 복합실체인 경우 요소적 호출순서는 다음과 같은 두 개의 연관관계, 즉 (LINK-AB, 1: PLANT-A)와 (LINK-AB, 2: CUST-B)(여기서 : 는 Key와 연관관계의 값을 분리함)를 나타낸다.

요소적 구조를 한 단계 더 추상화 구조적 모델이 일반적 구조이다. 요소적 구조에서 동

일한 타입의 요소들을 묶어서 "일반적 유사성(generic similarity)"을 만족시킬 경우 Genus로 그룹화될 수 있다. 즉, GENUS_A 내의 한 요소가 GENUS_B 내의 어떤 요소에 의존(호출)할 경우, GENUS_A에 있는 모든 요소들은 GENUS_B의 요소들을 호출하는 특성을 가질 경우 "일반적 유사성"을 갖는다고 할 수 있다. 이러한 Genus 객체도 역시 (그




(그림 4-4) Genus 클래스의 객체다이어그램

림 4-4)와 같이 그 타입에 따라 실체 Genus, 속성 Genus, 함수 Genus로 세분화할 수 있으며, 실체 Genus는 원시실체 Genus와 복합실체 Genus로 세분화된다. 그리고 속성 Genus는 그 값의 변동성여부에 따라 고정속성 Genus과 변동속성 Genus로 세분화될 수 있다. 또한 함수 Genus의 특수한 형태로서 참 또는 거짓의 값만을 가질 수 있는 테스트 Genus가 있다. 따라서 Genus 객체를 구성하는 하위객체들 간에도 (그림 4-3)과 같이 요소객체와 유사한 연관관계가 형성된다. 즉, 두 객체 GENUS_A와 GENUS_B 간의 의존관계인 Generic_Attribute_Of, Generic_Function_Of, Generic_Compound_Of 등의 관계가 형성되며, 이러한 연관관계의 사전(dictionary) 격인 일반적 호출순서(Generic_Calling_Sequence)가 있다. 예를 들어 Genus LINK가 PLANT와 CUST Genera를 호출할 경우, (LINK, 1: PLANT)와 (LINK, 2:CUST)의 두 연관관계가 만들어진다.

구조적 모델에서 최상위 수준의 추상화는 하나 또는 그 이상의 관련된 Genera의 배열(array)로 묶은 모듈러 구조이다. 모듈러 구조는 그 요소들이 최소한 부분적으로 계층화되어야만 한다. 즉, 모듈내에서 i 번째 genus가 j 번째 genus 보다 낮은 계층에 있을 때만이 호출할 수 있다. 복합모듈은 다른 모듈의 배열이다. 따라서 모듈러 구조에는 모듈간의 의존관계(Modular Dependency)인 "Modular_ Compound_Of"와 이들 관계의 사전(dictionary)격인 모듈포함(Modular_Content)관계가 있다(그림 4-2 참조).

## 4.3 구조적 모델의 객체 설계

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 명세하는데 사용될 수 있다. 그리고 이들 개념적 수준의 모델을 실제 컴퓨터 상에서 실행가능하도록 한 논리적 스키마 수준의 모델이 SML이다. 이들은 모두 상이한 추상화 수준에서 본 하나의 모델클래스이다.

```
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 4-5) 모듈러 구조의 객체설계

구조적 모델의 모듈러 구조와 일반적 구조를 설계하면 (그림 4-5) 및 (그림 4-6)과같다. 그리고 SML 스키마 클래스를 설계하면 (그림 4-7)과 같다. (그림 4-5) 및 (그림 4-6)에서 HEURISTICS에 있는 내용들은 각각 모듈러 구조와 일반적 구조의 무결성(integrity)을 보증하기 위한 지식이다. 이들 지식을 정리하면 다음과 같다.

① 각 genus는 여섯 개의 genus타입(pe, ce, fa, va, f, t) 중의 하나에 속해야 한다.
② 원시실체(pe)는 다른 어떤 genus도 호출하지 않는다.
③ 원시실체(pe)를 제외한 모든 genus타입은 적어도 하나 이상의 genus를 호출해야한다.
④ 속성(a)과 복합실체는 실체 genus 만을 호출할 수 있다.
⑤ 함수(f)는 속성(a)과 함수 genera 만을 호출할 수 있다.
⑥ 각 테스트(t)는 두 개의 다른 genera를 호출해야 한다. 호출되는 genera는 함수(f)이거나 속성(a) 또는 변수속성(va) 중의 하나이다.
⑦ 각 함수(f)와 테스트(t)는 하나의 generic rule을 가져야 한다.

```
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 4-6) 일반적 구조의 객체설계

⑧ 각 복합실체(ce)는 적어도 하나 이상의 원시실체(pe)에 대해서 인덱스를 붙여야 한다. 그리고 각 속성(a), 함수(f), 테스트(t)는 단지 하나의 실체 genus에 대해서 만 인덱스를 붙여야 한다.

⑨ 일반적 구조(generic structure)는 비순환적(acyclic)이어야 한다.

⑩ 각 genus는 모듈이나 모델 내에 포함되어 있어야 한다.

⑪ genus는 다른 genus나 모듈 또는 모델을 포함할 수는 없다.

⑫ 각 genus는 모듈러 구조(modular structure)에서 그 순서를 정의하는 고유한 값과 관련되어야 만 한다.

⑬ 각 모듈은 다른 모듈(상위 모듈)이나 또는 모델에 포함되어야 한다.

⑭ 모듈러 구조는 rooted tree이어야 한다.

⑮ 모듈러 구조는 단조계층을 이루어야 한다.

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 (그림 2-1)과 같은 텍스트 형태의 SML 스키마로 전환하는 메소드이다.

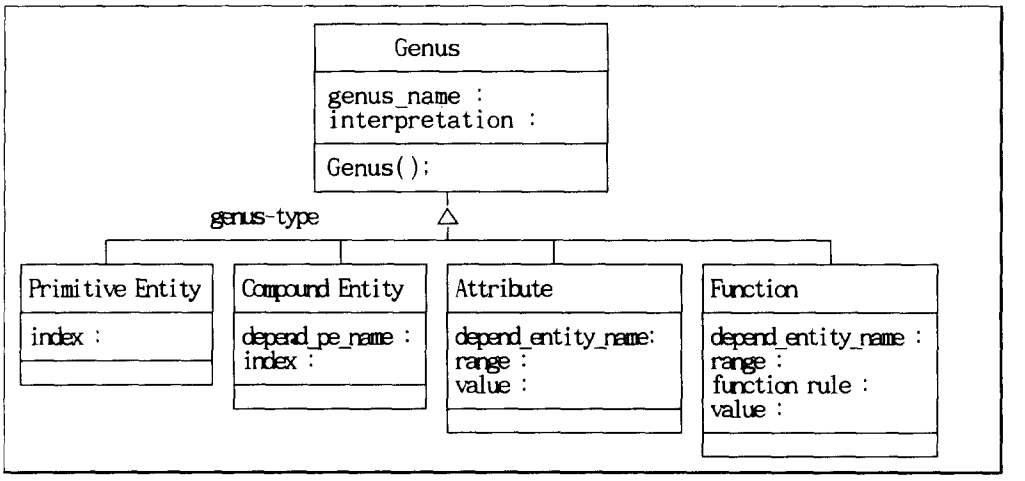
(그림 4-7)의 SML 스키마 클래스는 구조적 모델의 논리적 스키마를 나타내는 클래스로 모델의 텍스트 편집을 가능하게 하는 Edit_SML_Schema 메소드와 해산출기와의 인터페이스를 하는 Solver_Interface() 메소드를 가지고 있다.

또한 구조적 모델을 구성하는 Genus 객체를 그 타입에 따라 계승관계로 표현하면 (그

```
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 4-7) SML 스키마의 객체설계

림 4-8)과 같다. 그리고 구조적 모델을 구성하는 객체들 간의 관계(요소호출, Genus 호출, 모듈포함관계)에 대한 정보와 모델의 구체적인 데이터를 담은 요소세부테이블을 설계하면 (그림 4-9)와 같다. 이것들은 관계형 테이블로 설계되어 질 수 있다.



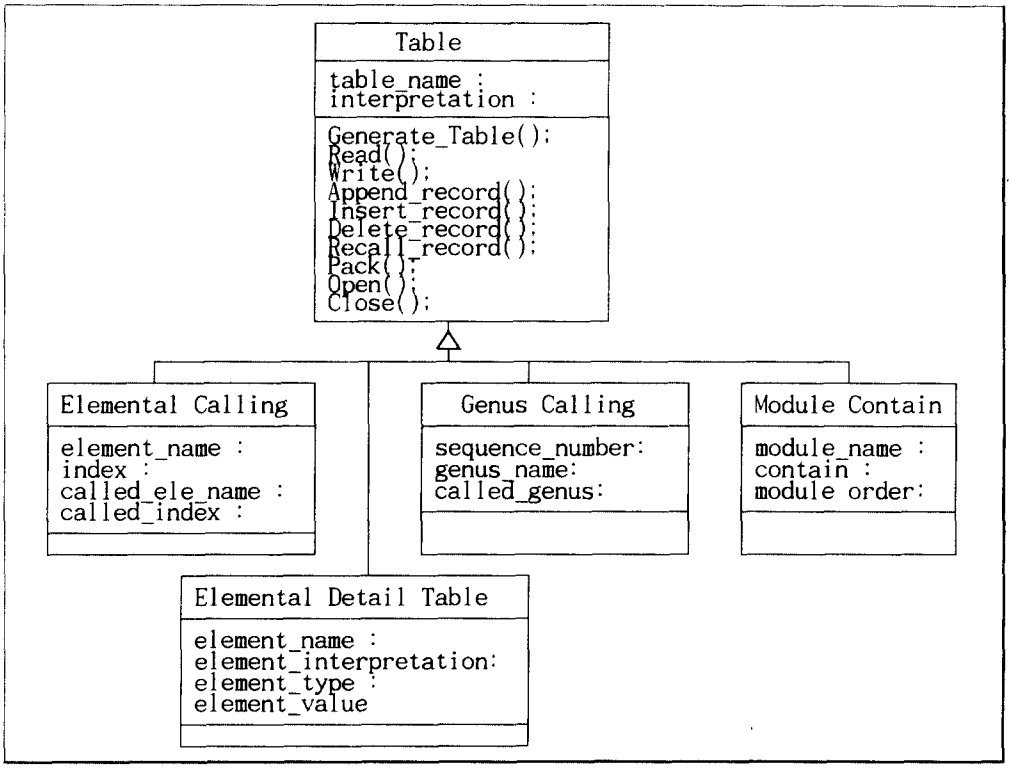

(그림 4-8) Genus 클래스 객체설계

# V. 결 론

객체지향 모델베이스의 개발은 조직내의 여러 부서에서 만들어지는 여러 유형의 모델들을 체계적으로 저장·관리할 수 있게 한다. 특히 메타-모델베이스 개념의 도입은 다양한 모델링 패러다임에서 개발된 모델들을 효과적으로 조직화할 수 있게 한다. 본 연구에서는 구조적 모델링 패러다임을 지원할 수 있는 객체지향 모델베이스 개발을 목표로 한다. 따라서 본 연구는 객체지향 모델베이스 개념이 구조적 모델링 지원을 위한 모델베이스 개발에도 적용될 수 있음을 보이는 데 그 목적이 있다 할 수 있다.

구조적 모델링은 다양한 형태의 모델을 표현하는 데 있어 공식적인 프레임웍을 제공해 주며, 모델이나 모델클래스를 표현하기 위해서 계층적으로 조직화되고 분할된 유속성 비순환 그래프를 사용한다. 구조적 모델링에서 구조적 모델은 그 추상화 수준에따라 세 가지 수준의 구조, 즉 요소적 구조, 일반적 구조, 모듈러 구조를 갖는다. 이들 세 가지 수준의 구조는 모델의 개념적 수준을 표현하는 개념적 스키마로 볼 수 있으며, 또한 컴퓨터 상에서 곧바로 실행가능한 언어(EML)인 SML로 표현된 구조적 모델은 논리적 스키마로 볼 수 있다.

객체지향적 모델관리의 관점에서 볼 때, 일반적 구조 및 모듈러 구조와 SML 스키마는 의사결정지원을 위한 하나의 모델클래스이며, 요소적 구조는 그 모델클래스의 구체적인 인스턴스가 된다. 그리고 구조적 모델을 구성하는 객체로는 요소, Genus, 모듈이 있고, 각 요소와 Genus는 그 타입에따라 실체(원시실체, 복합실체), 속성(고정속성, 변동속성), 함수(테스트)객체로 나눌 수 있다.




(그림 4-9) 구조적 모델을 위한 데이타베이스 테이블 객체의 설계

구조적 모델링을 위한 객체지향 모델베이스 개발에 있어 객체모델링기법(OMT)을 적용하므로써 다음과 같은 잇점을 얻을 수 있다.

첫째, 구조적 모델링을 위한 모델베이스를 구성하는 여러 객체들(구조적 모델, 요소적

구조, 일반적 구조, 모듈러 구조, 요소, 요소호출, Genus, Genus 호출, 모듈, 모듈포함) 간의 관계를 객체다이어그램으로 표현하므로써 개념적 모델베이스 스키마 설계를 용이하게 한다.

둘째, 구조적 모델을 구성하는 요소들 간의 계층적 관계를 표현하는 데 있어 집합개념을 도입하므로써, 모델베이스 내에서 이들 간의 계층적 조직화를 가능하게 하며, 이들 간의 참조적 무결성을 가지도록 한다.

셋째, 모델베이스 내에서 구조적 모델을 구성하는 객체들의 타입화와 이들 간의 계승관계를 모델링하므로써 객체의 재사용성을 강화할 수 있다. 계승개념은 코드의 재사용을 가능하게 하므로 써, 기존의 모델베이스 내에 존재하는 모델객체들 간의 계승관계를 이용하여 신속한 모델수립활동을 돕는다.

본 연구에서는 구조적 모델링을 위한 모델베이스 내의 개별 모델객체설계에 있어 각 객체의 속성과 메소드 종류 만을 열거하고 있으며, 이들 속성과 메소드의 세부적인 설계는 제시하지 않았다. 이것은 본 연구의 범위를 벗어나므로 생략하였으며, 추후 구조적 모델링 지원을 위한 모델관리시스템(OOMMS/SM : Object-Oriented Model Management System for Structured Modeling)의 설계와 구현에서 제시하고자 한다.

# 참 고 문 헌


정대율, "객체지향적 모델관리시스템을 위한 기능요구분석," 정보시스템연구, 제4권, 제1호 1995년, pp. 129-153.

정대율, "DSS의 모델베이스 개발을 위한 객체모델링 프레임웍," 박사학위논문, 부산대학교 대학원, 1996. 2..

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

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.

Brodie,M.L., J. Mylopoulos, J. W. Schmidt, *On Conceptual Modeling*, Apring-Verlag, New York, 1984.

Bruegge, B., J. Blythe, J. Jackson, and J. Shufelt, "Object-Oriented System Modeling with OMT," *OOPSLA'92,* 1992, pp. 359-376.

Chang, A., C. W. Holsapple, and A.B. Whinston, "Model Management Issues and Directions," *Decision Support Systems,* Vol. 9, 1993, pp. 19-37.

Chari,S.,and R. Krishnan,"Toward a Logical Reconstruction of Structured Modeling," Decision Support Systems, Vol. 10, 1993, pp. 301-317.

Clemence, Jr., R. D.,"LEXICON: A Structured Modeling System for Optimization," Maser's Thesis, Naval Postgraduate School, Monterey, CA, June 1984.

Dolk, D. R., "Model Management and Structured Modeling: The Role of an Information Resource Dictionary System," Communication of ACM, Vol. 31, 1988, pp. 704-718.

Farn, C. K., "An Integrated Information System Architecture Based on Structured Modeling," *Ph.D. Dissertation,* Graduate School of Management, UCLA, 1985.

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 Formal Aspects of Structured Modeling," *Operations Research,* Vol. 37, 1989, pp. 30-51.

Geoffrion, A. M., "FW/SM: A Prototype Structured Modeling Environment," *Management Science*, Vol. 37, No. 12, 1991, pp. 1513-1538.

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

Geoffrion, A. M., "Structured Modeling: A Survey and Future Research Directions," *ORSA CSTS Newsletter,* Vol. 15, No. 1, Spring 1994b, pp. 10-20.

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.

Jones, C., "Attributed Graphs, Graph-grammars, Structured Modeling," in: B. Shetty (ed.), *Annals of Operations Research, Model Management Systems,* Vol. 38, 1992, pp. 281-324.

Lenard, M. L.,"Representing Models as Data," *Journal of Management Information Systems,* Vol. 2, 1986, pp. 36-48.

Lenard, M. L., "Fundamentals of Structured Modeling," in G. Mitra(ed.), *Mathematical Models for Decision Support,* Springer-Verlag, Berlin, 1988, pp. 695-713.

Lenard, M. L., "Structured Model Management," in G. Mitra(ed.), *Mathematical Models for Decision Support,* Springer-Verlag, Berlin, 1988, pp. 375-391.

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

Muhanna, W. A., "On the Organization of Large Shared Model Bases," in: B. Shetty(ed.), *Annanls of Operations Research, Model Management,* 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., and R.A. Pick, "Meta-modeling Concepts and Tools for Model Management: A Systems Approach," *Management Science,* forth coming.

Neustadter, L., A. Geoffrion, S. Maturana, Y. Tsai, and F. Vicuna, "The Design and Implementation of a Prototype Structured Modeling Environment," in: B. Shetty (ed.), *Annals of Operations Research, Model Management Systems,* Vol. 38, 1992, pp. 453-484.

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.

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

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

# On the Organization of Object-Oriented Model Bases for Structured Modeling


## <Abstract>

This paper focus on the development of object-oriented model bases for Structured Modeling. For the model base organization, object modeling techniques and model typing concept which is similar to data typing concept are used.

Structured modeling formalizes the notion of a definitional system as a way of describing models. From the object-oriented concept, a structured model can be represented as follows.

Each group of similar elements(genus) is represented by a composite class. Other type of genera can be represented in a similar manner. This hierarchical class composition gives rise to an acyclic class-composition graph which corresponds with the genus graph of structured model. Nodes in this graph are instantiated to represent the elemental graph for a specific model. Taking this class composition process one step further, we aggregate the classes into higher-level composite classes which would correspond to the structured modeling notion of a module.

Finally, the model itself is then represented by a composite class having attributes each of whose domain is a composite class representing one of the modules. The resulting class-composition graph represent the modular tree of the structured.