경영정보학연구
제10권 제2호
2000년 6월

부서간 협동적 작업을 지원하는 모형관리 체계의 개발

허 순 영*, 김 형 민*

A Model Management Framework for Supporting Departmental Collaborative Work

Huh, Soon-Young, Kim, Hyung-Min

Recently, as business problems become more complicated and require more precise quantitative results, large-scale model management systems are increasingly in demand for supporting the decision-making activities. In addition, as distributed computing over networks gains popularity, departmental computing systems are gradually adopted in an organization to facilitate collaboration of geographically dispersed multiple departments. In departmental collaborative model management systems, multiple departments share common models but approach them with different user-views depending on their departmental needs. Moreover, the shared models become evolved as their structures and the corresponding data sets change due to the dynamic nature of the operating environment and the inherent uncertainty associated with the problems. In such capacity, providing the multiple departmental users with synchronized and consistent views of the models is important to improve the overall productivity. In this paper, we propose a collaborative model management framework for coordinating model change and automatic user-view update in a departmental computing environment. To do so, we describes changes in the model and their effects occurred in departmental model management environments and identifies the constructs and processes for maintaining the consistency between a shared model and its departmental user-views. Especially, in this framework, generic model concept was adopted for accommodating diverse mathematical models in a uniform way in a modelbase and object-oriented database management systems (ODBMS) for combining the model management constructs and automatic user-view update mechanisms in a single formalism. A prototype object-oriented modeling environment was developed using an ODBMS called ObjectStore and C++ programming language on Windows NT.

* 한국과학기술원 테크노경영대학원

Ⅰ. 서 론

최근 들어, 회사의 업무환경이 점차 복잡하여지고 그 변화속도가 빨라짐에 따라 조직 내의 의사결정에 도움을 주기 위하여 보다 정량적이고 현재상황을 정확하게 반영하는 분석이 필요하게 되었으며, 이를 지원해 주기 위한 분석 도구로서 모형관리시스템 (model management system)의 중요성이 점차 커지고 있다. 모형관리시스템은 회사 내에서 여러 가지 분석을 위해 사용되는 수리적 모형들 (mathematical models) - 예를 들어, 최적화 모형, 예측 모형, 모의실험 모형 등 - 을 여러 사용자가 공유할 수 있는 정보자원 (information resource)으로 인식하여, 모형들을 모형베이스 (modelbase) 내에 저장 관리하고, 저장되어 있는 모형들을 여러 사용자가 접근하여 모형을 실행시키고, 그 결과를 분석하여 업무에 이용할 수 있도록 지원해 준다 [Blanning, 1993; Dolk, 1988; Dolk, 1993; Mannino 1990; Muhanna, 1994]. 한편, 회사 내의 네트워크 환경이 발달하여 감에 따라 회사 내에서 관리되는 각종 정보자원이 회사 전체의 관점에서 이용될 수 있는 방향으로 나아가고 있으며, 의사결정에 도움을 주기 위해 분석업무를 지원하는 모형관리시스템도 하나의 단위부서의 업무를 지원하기 위하여 이용되기 보다는 관련된 여러 부서간에 업무를 협동적으로 처리할 수 있도록 전사적인 관점에서 운용될 수 있어야 한다.

여러 부서에서 모형베이스에 저장되어 있는 하나의 모형을 공유하고 이를 이용하는 경우, 각 부서는 맡고 있는 업무가 서로 다르기 때문에 공유되는 모형으로부터 얻고자 하는 정보의 내용과 관심사항이 서로 달라지게 된다. 예를 들어, 생산비용 및 생산능력, 창고수용용량 그리고 예측된 시장수요 등을 입력 받아 미래의 생산 및 판매 계획을 수립해 주는 수리계획모형 (mathematical programming model)이 모형베이스에 저장되어 있는 경우, 생산부서는 이 모형에 생산비용 및 생산능력 등을 입력하여 미래의 제품 생산 계획을 얻고자 할 것이며, 판매부서는 예측된 미래의 시장수요를 입력하여 각 제품에 대한 판매 계획을 알기를 원할 것이다. 한편, 물류를 담당하는 부서는 창고수용용량 및 물류비용 등을 모형에 입력하고 미래의 수송계획을 얻기를 원할 것이다. 이와 같이, 회사업무 전반에 관련되는 수리적 모형들은 여러 부서의 업무에 연관이 되며, 이를 지원해 주기 위한 협동적 모형관리시스템 (collaborative model management systems)은 각 부서의 모형 이용에 대한 요구사항을 만족시켜 줄 수 있도록 각 부서에 동일한 모형에 대한 서로 다른 사용자 뷰 (모형을 이용하기 위한 시스템 화면)를 제공해 주어야 한다.

한편, 모형관리시스템 내에 구축되어진 모형들은 한번 개발된 이후에 그대로 고정되는 것이 아니라 모형이 표현하는 실제 세계의 변화와 모형이 가지는 내재적 불확실성으로 인해 그 구조가 계속적으로 개선되어지고 변화되게 된다 [Muhanna, 1995; Sprague, 1980]. 모형의 구조에 대한 개선 및 변화는 회사 내에서 모형을 이용한 업무 분석을 지원하는 모형 개발 관련 부서에서 주로 이루어 지게 된다. 또한, 개발 되어진 모형에 필요한 데이터를 입력하고 그 실행결과를 이용하게 되는 현업 부서 (operational department)에서는 자신의 업무환경의 변화에 따라 모형에 입력하는 데이터, 즉, 모형의 인스턴스를 변화 시켜야 할 필요가 있다. 이러한 모형의 구조 (model structure)의 변화와 모형의 인스턴스 (model instance)의 변화는 해당 모형의 실행결과를 변화 시키게 되며, 따라서 이를 이용하는 여러 관련된 부서는 모형의 변동사항을 즉각적으로 알아야 할 필요가 있다. 본 논문에서는 이와 같은 부서간 협동적 모형관리시스템에서 모형에 발생하는 변화를 회사 내의 여러 부서에 존재하는 사용자 뷰들로 자동적으로 반영하기 위한 기법을 제안하고자 한다.

모형관리시스템에 대한 기존의 연구들은 (1)

수리적 모형들을 시스템적으로 표현하고 관리하기 위한 여러 가지 모델링 언어 (AMPL [Fourer, 1990], GAMS [Bisschop, 1982], SML [Geoffrion, 1987] 등)에 대한 연구를 비롯하여, (2) 데이터베이스를 이용하여 모형베이스를 구축하기 위한 연구 (관계형 데이터베이스 기반 모형베이스 [Blanning, 1985; Lenard, 1986], 객체지향 데이터베이스 기반 모형베이스 [Huh, 1993; Muhanna, 1995]), (3) 그룹의사결정지원을 위한 모형관리에 관한 연구 [Liang, 1988; Martin, 1996; Nunamaket, 1988], (4) 저장되어진 모형을 이용하기 위한 사용자 뷰에 관한 연구 [Greenberg, 1995; Liang, 1986; Ma, 1989; Murphy, 1992] 등 모형을 회사 내에서 공유되는 정보자원으로 이용하기 위한 연구들이 폭 넓게 이루어져 왔다. 하지만 모형관리시스템의 운용과정에서 나타나는 모형의 변화와 이에 따른 여러 사용자 뷰에 미치는 영향 및 사용자 뷰로의 자동적인 갱신에 관한 연구는 아직까지 이루어 지고 있지 않다. 특히, 여러 부서간에 협동적으로 이용되는 모형 관리 환경에서 특정 부서에 의한 모형의 변화와 이에 대한 다른 부서로의 통보 및 반영은 기존의 모형관리시스템의 운용범위를 부서 단위에서 전사적인 범위로 확장하기 위한 매우 중요한 기능으로 생각되어 진다.

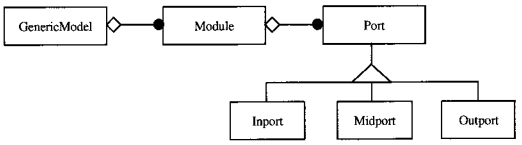
이 논문에서 제안하고자 하는 부서간 협동적 모형관리시스템은 모형들을 관리하기 위한 모형베이스의 이론적 토대로서 객체지향 데이터베이스를 기반으로 하는 일반적 모형 개념 (generic model concept) [Huh, 1993]을 이용한다. 일반적 모형 개념은 다양한 형식의 수리적 모형들을 하나의 일관된 형태로 표현 할 수 있는 개념적 틀을 제공해 주며, 이를 이용하여 실제적으로 다양한 모델들을 데이터베이스 내에 객체지향 구성체들을 이용하여 저장관리 할 수 있다. 한편, 이러한 모형베이스를 기반으로 하여 저장되어진 모형에 변화가 발생한 경우 각 부서에 생성되어져 있는 사용자 뷰들을 자동적으로 갱신하기 위하여 이 논문에서는 첫째, 공유되는 모형과 각 부서의 사용자 뷰들 간의 종속관계를 관리하기 위한 기법과 둘째, 모형의 변화를 자동적으로 각 부서로 통보하고 이를 사용자 뷰에 반영하기 위한 기법을 제안한다. 특히, 저장되어진 모형에 발생하는 변화를 사용자 뷰에 미치는 영향에 따라 모형의 구조적 단계에서의 변화와 모형의 인스턴스 단계에서의 변화로 나누어 관리한다. 이러한 모든 기능들은 실제로 객체지향 데이터베이스 시스템인 ObjectStore [Lamb, 1991]를 기반으로 C^{++} 프로그래밍 언어를 이용하여 구현되었다. 이 논문의 구성은 다음과 같다. 먼저, 2절에서는 객체지향 데이터베이스를 기반으로 하여 모형을 저장관리하기 위한 일반적 모형 개념에 대하여 알아보고, 3절에서는 저장되어 있는 모형에 발생되는 변화와 이러한 변화가 각 부서로 통보되어야 하는 필요성에 대하여 논의한다. 그리고, 4절에서는 변화통보와 사용자 뷰의 자동갱신을 위한 기본적 구성체에 대하여 설명하며, 5절에서는 4절에서 설명한 기본적 구성체들을 부서간 협동적 모형관리시스템과 같은 분산환경에 적용하는 문제와 사용자 뷰의 자동갱신을 위한 구체적인 과정에 대하여 논의하고자 한다. 마지막으로 6절에서는 이 논문을 요약하고 향후 연구방향을 소개한다.

Ⅱ. 객체지향 데이터베이스를 이용한 수리적 모형의 저장

모형베이스는 회사에서 공유되는 수리적 모형들을 저장 관리하며, 모형관리시스템은 모형베이스에 저장되어 있는 모형들에 여러 사용자가 접근하여 모형의 개발, 실행, 결과분석, 수정 등의 작업을 할 수 있는 기능들을 제공해 준다. 일반적으로 모형베이스 내에서 관리되는 수리적 모형들은 그 구성 요소들을 시스템 내에서 체계적으로 관리하기 위하여 계층적인 구조로 나타

낼 수 있다 [Bisschop, 1982; Fourer, 1990; Geoffrion, 1987; Geoffrion, 1992]. 예를 들어, 최적화 모형의 경우 하나의 모형은 목적식 (objective function), 제약식 (constraint), 결정변수 (decision variable), 인수 (parameter), 인덱스 (index) 등으로 이루어지며, 각각의 구성 요소들은 문제에 따라 여러 가지의 제약식, 여러 가지의 결정변수 및 인수 등을 가지게 된다. 따라서 하나의 모형은 나무 (tree) 형태의 계층적인 구조로 나타낼 수 있으며, 각각의 노드 (node)들은 객체지향 구성체들로 표현이 가능하다 [Huh, 1993; Le Claire, 1990; Muhanna, 1995]. 이 논문에서는 모형베이스를 구축하기 위하여, 객체지향 데이터베이스를 기반으로 하는 일반적 모형 개념 (generic model concept) [Huh, 1995]을 이용한다. 일반적 모형 개념은 모형을 모형베이스 내에 계층적인 구조로 표현하고 이를 저장하기 위한 객체지향 구성체들을 제공해 주기 때문에 계승 (inheritance), 다형성 (polymorphism), 캡슐화 (encapsulation) 등과 같은 객체지향 방법론이 가지는 일반적인 장점들을 모두 이용할 수 있으며, 다양한 형태의 모형들을 모형베이스 내에 일관성 있는 방법으로 저장 관리 할 수 있도록 해준다. 일반적 모형 개념에 대한 기본적인 구성체들과 이러한 구성체들을 이용하여 수리적 모형이 객체지향 데이터베이스를 이용한 모형베이스 내에 저장되는 구조를 간략히 설명하면 다음과 같다.

일반적 모형 개념에서는 하나의 모형을 객체지향 데이터베이스 내에 표현하기 위하여 기본적으로 세가지 타입의 클래스를 제공하는데, 이는 가장 상위 단계인 일반적 모형 타입 (generic model type), 중간 단계인 모듈 타입 (module type), 그리고 가장 하위 단계인 포트 타입 (port type)이며, 포트타입은 다시 입력포트 (inport), 출력포트 (outport), 중간포트 (midport) 세가지로 구분한다. 입력포트는 모형을 실행시키기 위하여 필요한 입력 데이터들을 받아들이게 되는 부분으로 사용자나 회사 내의 데이터베이스로부터 데이터를 받아들이게 된다. 출력포트는 모형의 실행 결과를 외부로 내보내는 부분으로 모형을 이용하는 각 부서는 이러한 출력포트를 통해서 원하는 분석 결과를 얻게 된다. 한편, 중간포트는 모형을 구성하는 부분 중 모형의 입출력 부분을 제외한 나머지 부분으로 입력포트의 값으로부터 출력포트의 값을 생성해 내기 위하여 필요한 요소들을 나타낸다. <그림 1>은 이러한 기본적인 클래스들의 관계를 Rumbaugh의 OMT (Object Modeling Technique) 방법 [Rumbaugh, 1991]으로 표현한 것이다. OMT 방법의 표현법에서 사각형과 실선은 각각 클래스와 클래스들 간의 관계를 나타내며, 클래스 간의 관계 중 계승관계 (inheritance relationship)는 삼각형으로, 집합관계 (aggregation relationship)는 마름모로 나타내며, 실선 끝의 검은 점은 복수개가 연결될 수 있음을 나타낸다.



<그림 1> 일반적 모형 개념에서 모형을 표현하기 위한 기본적인 구성체

<그림 1>에서 보는 바와 같이 하나의 일반적 모형 (generic model)은 여러 개의 모듈 (module)들로 구성되며, 각각의 모듈들은 다시 여러 개의 포트들로 구성된다. 한편, 입력포트 (inport), 출력포트 (outport), 중간포트 (midport)는 포트의 세가지 종류를 나타내며 포트 (port)의 성질을 계승 받게 된다. 실제적으로 모형을 구성하는 모든 요소들은 세가지 포트 타입 중에 한가지로 표현이 되며 (모형의 입력부분은 입력포트, 출력부분은 출력포트, 나머지는 중간포트), 포트들을 그 성격에 따라 그룹화 해 주기 위하여 모듈 타입을 이용한다. 일반적 모형 개념을 이용하여 수리적 모형을 모형베이스 내에 저장하는 방법을 알아보기 위하여 <그림 2>와 같은 미래의 생산

입력변수

$T > 0$ 미래의 생산 및 판매 기간의 개수

$ProdCost \geq 0$ 1단위의 제품을 생산하는데 필요한 제조비용

$StorCost \geq 0$ 1단위의 제품을 1기간동안 보관하는데 필요한 보관비용

$ProdCapa_{t1} \geq 0$ $t1 \in \{1,\cdot,T\}$: 기간 t1의 생산능력

$Demand_{t2} \geq 0$ $t2 \in \{1,\cdot,T\}$: 기간 t2의 시장수요

결정변수

$Quant_{t1\ t2} \geq 0$ $t1 \in \{1,\cdot,T\}$, $t2 \in \{1,\cdot,T\}$, $t2 \geq t1$: 기간 t1에 생산하여 기간 t2에 판매하는 제품의 수량

목적식

$$\text{Minimize} \sum_{t1 \in \{1,\cdot T\},\ t2 \in \{1,\cdot,T\}} (ProdCost + (t2 \quad t1) StorCost) Quant_{t1\ t2}$$

$t2 \in t1$: 미래의 전체 기간에 대한 생산비용과 보관비용의 합을 최소화

제약식

$$\sum_{t2 \in \{1,\cdot T\}} Quant_{t1\ t2} <= ProdCapa_{t1}$$

$t1 \in \{1,\cdot,T\}$, $t2 \geq t1$: 기간 t1에 생산되는 제품의 수량은 해당 기간의 생산능력을 넘을 수 없음

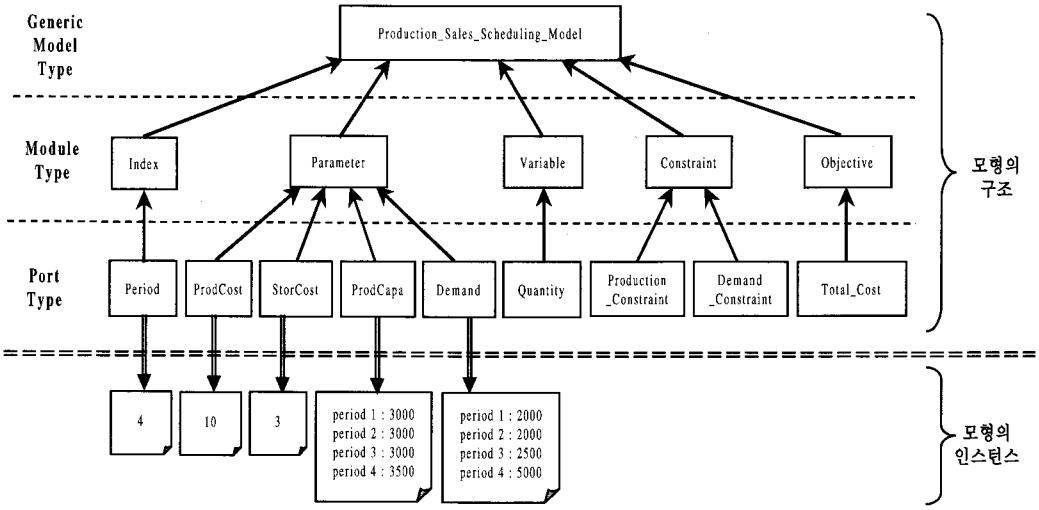
$$\sum_{t1 \in \{1,\cdot T\}} Quant_{t1\ t2} = Demand_{t2}$$

$t2 \in \{1,...,T\}$, $t2 \geq t1$: 기간 t2에 판매되는 제품의 수량은 해당 기간의 시장수요를 만족시킴

<그림 2> 생산 및 판매 계획 수립을 위한 선형계획모형

및 판매 계획을 수립하기 위한 선형계획모형 (linear programming model)을 생각해 보자. 예로 사용하는 생산 및 판매 계획 모형 (production and sales scheduling model)은 미래를 T개의 기간으로 나누고 각 기간별 생산능력을 넘지 않는 범위 내에서 각 기간의 시장수요를 만족시키기 위하여 생산비용과 저장비용의 합을 최소화하는 기간별 생산과 판매 계획을 세우는 문제를 다루고 있다.

<그림 2>에 나타나 있는 수식들은 생산 및 판매 계획 모형의 구조를 나타내고 있으며, 구체적인 문제에 적용되어 실제로 실행되기 위해서는 입력변수에 제조비용, 보관비용 등과 같은 실제 상황에 대한 데이터를 입력변수의 인스턴스로 입력시켜 주어야 한다. 한편, 모형을 실행시키게 되면 결정변수에 값이 결정되게 되고, 이는 결정변수의 인스턴스가 된다. 따라서, 모형의 인스턴스는 해당 모형의 입력 데이터 뿐 아니라 출력 데이터까지를 포함한다. 한편, 위와 같은 모형을 일반적 모형 개념을 이용하여 모형베이스 내에 저장하기 위해서는 앞에서 언급한 바와 같이 모형을 구성하는 요소들을 일반적 모형 개념의 가장 하위 단계인 포트들로 표현하여야 한다. 먼저, 생산 및 판매 계획 모형에서 입력변수에 해당하는 기간의 개수 (T), 제조비용 (ProdCost), 보관비용 (StorCost), 생산능력 (ProdCapa), 시장수요 (Demand)는 외부로부터 입력데이터로 받아들여야 하는 부분으로 입력포트에 해당한다. 다음으로 결정변수인 각 기간별 생산 및 판매 수량 (Quant)은 모형의 실행 후 결과 값을 저장하는 부분으로 출력포트에 해당한다. 마지막으로 입출력 부분을 제외한 목적식과 제약식을 나타내는 총 비용을 계산하는 함수와 생산능력 제약식 그리고 시장수요 제약식 등은 중간포트가 된다. 일반적 모형 개념에서는 이와 같은 포트들을 모듈을 이용하여 모형을 구성하는 요소들의 성격에 따라 인덱스 (index), 인수 (parameter), 변수 (variable) 등과 같은 몇 개의 그룹으로 나누어 관리할 수 있으며, 최종적으로 이러한 모듈들이 모여서 하나의 모형을 구성하게 된다. <그림 3>은 일반적 모형 개념을 이용하여 <그림 2>의 생산 및 판매 계획 모형이 객체지향 데이터베이스



<그림 3> 객체지향 데이터베이스 구성체를 이용한 모형의 저장 예

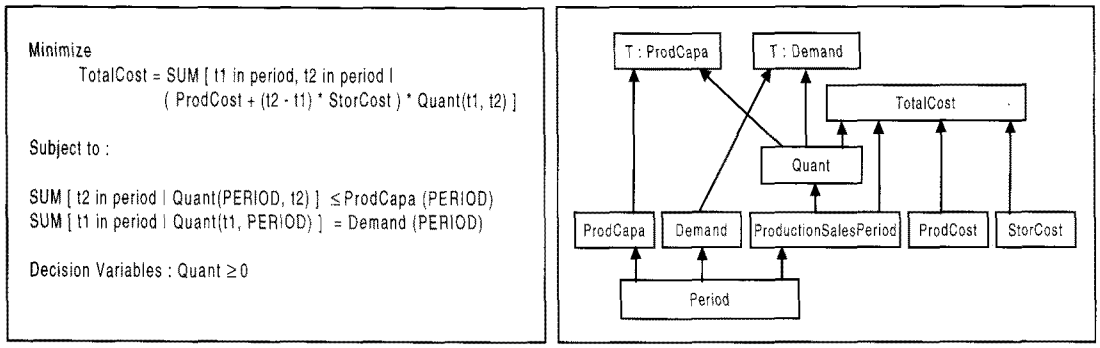
내에 저장되는 방법을 나타낸다.

위의 그림은 생산 및 판매 계획 모형의 구조를 일반적 모형 타입, 모듈 타입, 포트 타입과 같은 객체지향 구성체들을 이용하여 표현하고, 이중 데이터의 입력부분에 해당하는 입력포트에 모형의 인스턴스가 입력되어 있는 상황을 나타낸다. 모형의 인스턴스는 모형을 이용하여 분석하고자 하는 구체적인 문제를 표현하며, 그림에서는 미래 기간의 개수로 4, 생산비용으로 10, 저장비용으로 3, 그리고 각 기간에 대한 구체적인 생산능력과 시장 수요들을 인스턴스로 가지고 있다. 이와 같이, 모형의 구조와 모형의 인스턴스는 명시적으로 구분이 될 수 있으며, 모형의 구조에 전혀 영향을 미치지 않고도 모형의 인스턴스는 변화할 수 있다. 예를 들어, 생산비용을 10에서 12로 변경하는 경우 이는 모형의 구조에 전혀 영향을 미치지 않는다. 하지만, 모형의 인스턴스는 모형의 구조가 어떻게 구성되어 있는가에 따라 영향을 받게 되므로, 모형의 구조가 변하는 경우에는 보통 모형의 인스턴스도 따라서 변하게 된다. 이러한 모형의 변화와 그에 따른 영향에 대해서는 다음 절에서 보다 자세히 다루도록 한다.

Ⅲ. 저장된 모형의 변화와 이에 대한 통보의 필요성

앞에서 설명한 바와 같이 회사 내의 여러 부서에서 공유하게 되는 수리적 모형들은 일반적 모형 개념을 이용하여 객체지향 데이터베이스를 기반으로 한 모형베이스 내에 저장관리 될 수 있다. 그리고 저장되어진 모형들은 여러 부서에 존재하는 클라이언트 (client)들에 생성되어지는 모형에 대한 사용자 뷰를 통해서 접근되어지고 이용되어진다. 모형관리시스템에서 제공해 주는 사용자 뷰는 모형을 이용하는 사용자 각자마다 개인적인 취향이나 맡고 있는 업무 그리고 사용자의 모형에 대한 지식 등이 서로 다르기 때문에 하나의 모형을 여러 관점에서 볼 수 있도록 다양한 형태로 제공되어져야 하며, 이러한 사용자 뷰들은 크게 모형의 인스턴스를 제외한 모형의 구조만을 보여주는 구조적 뷰 (structure view)와 모형에 적용되는 구체적인 문제에 해당하는 모형의 인스턴스를 보여주는 인스턴스 뷰 (instance view)로 나눌 수 있다.

먼저, 구조적 뷰는 모형의 구조를 표현하는 사용자 뷰로서 주로 모형 개발 부서에서 수리적



(a) 대수적 뷰 (b) 지너스 그래프 뷰

<그림 4> 생산 및 판매 계획 모형에 대한 구조적 뷰의 예

모형에 대한 전문적인 지식을 가지고 있는 사용자들이 모형을 개발하고, 업무 환경의 변화에 따라 개발되어진 모형의 구조를 수정하기 위하여 이용하게 된다. <그림 4>는 <그림 2>에서 예로 든 생산 및 판매 계획 모형에 대한 구조적 뷰의 예를 나타내는데, (a)와 (b)는 각각 이에 대한 대수적 뷰 (algebraic view)와 Geoffrion의 지너스 그래프 뷰 (genus graph view) [Geoffrion, 1987; Geoffrion, 1992]를 나타낸다. 대수적 뷰는 수리적 모형의 구조를 대수적인 수식을 이용하여 표현한 사용자 뷰이며, 지너스 그래프 뷰는 모형을 여러 개의 구성 요소들로 나누어 표현하고 모형의 구성 요소들 간에 미치는 영향을 사이클이 없고 방향성이 있는 그래프 (acyclic directed graph)로 나타낸 사용자 뷰이다. 이 밖에도 모형의 구조를 표현하기 위하여 Block-Schematic Representation, Activity-Constraint Graph, Netform Graph [Greenberg, 1995; Murphy, 1992] 등을 이용할 수 있다. 이와 같이, 하나의 모형에 대하여 여러 가지 다른 구조적 뷰가 존재할 수 있으며, 모형의 구조를 보고자 하는 사용자들은 자신이 선호하는 사용자 뷰를 이용할 수 있다. 예를 들어, 모형에 대한 전문적인 지식을 가지고 있지 않은 의사결정자들은 보통 <그림 4> (b)에 나타나 있는 지너스 그래프 뷰와 같은 개념적인 사용자 뷰를 선호하며, 모형 개발자들은 그림 4의 (a)와 같이 모형을 수식으로 표현한 사용자 뷰를 통해서 수리적 모형의 구조를 보기를 원할 것이다.

앞에서 언급한 바와 같이, 모형은 한번 개발되어진 이후로 그대로 고정되는 것이 아니라 모형이 표현하는 업무환경이 변하게 되면, 이를 모형에 반영하기 위하여 모형의 구조 또한 변화하여야 한다. 물론, 모형을 개발하는 단계에서도 모형 개발자에 의해 모형에 대한 계속적인 개선 작업이 이루어 지게 되며, 이 때에도 모형베이스 내에 저장되어진 모형의 구조는 변화하게 된다. 모형베이스 내에 저장되어 있는 모형의 구조를 변화 시킨 경우 각 부서에 생성되어 있는 해당 모형에 대한 구조적 뷰들은 이러한 변화사실을 모르게 되며, 따라서 모형베이스의 실제 모형의 구조와 다른 내용을 가지게 된다. 이러한 모형과 사용자 뷰간의 불일치성 (inconsistency)은 모형을 공유하는 사용자들이 서로 다른 모형의 내용을 보게 됨으로써 업무의 효율성을 떨어뜨리게 된다. 예를 들어, <그림 2>에 나타난 생산 및 판매 계획 모형에서 상품에 대한 운송비용을 새롭게 고려하고자 하는 경우, <그림 2>의 선형계획 모형에 <그림 5>의 (a)와 같이 새로운 변수 (Ttans Cost)를 추가하여야 하며, 목적식이 운송비용을 포함하는 형태로 수정되어야 한다. 그리고 이에 따라 <그림 4>의 구조적 뷰는 <그림 5>의 (b)와 같이 바뀌어져야 한다.

입력변수 ....

TrnasCost 0 1단위의 제품을 운반하는데 필요한 운송비용

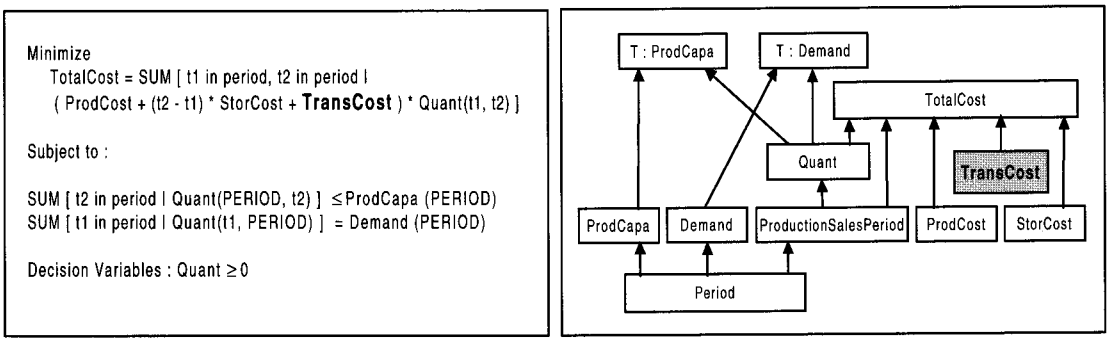
....

목적식

Minimize $\sum_{t1\in\{1,\cdot T\},\ t2\in\{1,\cdot,T\}}$ (ProdCost+(t2 - t1) StorCost+**TransCost**) Quant $_{t1\ t2}$

t2≥t1 : 생산비용, 보관비용, 운송비용의 합을 최소화

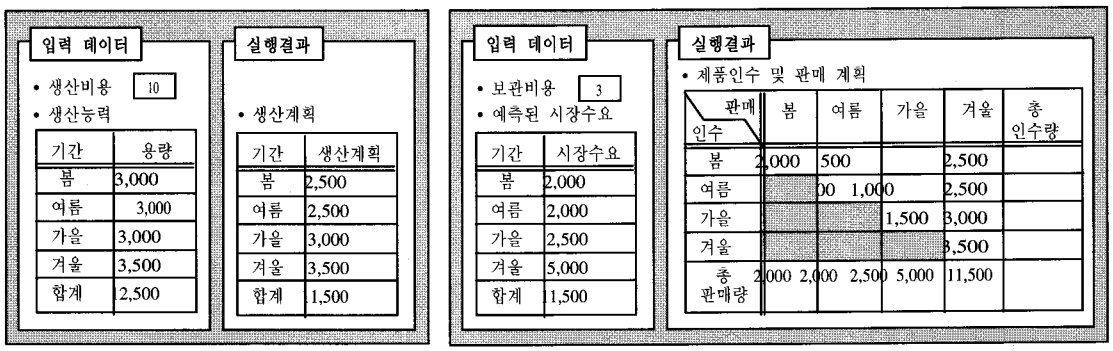
(a) 모형의 구조적 변화의 예



(b) 구조적 뷰로의 모형의 구조적 변화의 반영

<그림 5> 모형의 구조적 변화와 사용자 뷰로의 반영

한편, 각 부서의 사용자들은 개발되어진 모형에 구체적인 문제에 대한 데이터인 모형의 인스턴스를 적용하여 모형을 실행시키고 그 결과를 보게 되는데, 이를 위해서 사용자들은 모형의 인스턴스 뷰를 이용하게 된다. 인스턴스 뷰는 모형의 구조에 적용되어질 실제적인 문제에 대한 데이터를 입력하고 모형의 실행결과를 출력해 주는 사용자 뷰로서, 주로 생산, 보관 및 판매 부서와 같은 현업부서에서 모형을 이용하여 자신의 업무와 관련 있는 분석작업을 수행하기 위하여 이용하게 된다. 각 부서에서 사용하는 인스턴스 뷰들은 해당 부서의 업무의 내용에 따라 공유되고 있는 모형에 대한 관심사항이 서로 다르며, 입력 가능한 데이터와 원하는 출력결과도 서로 다르기 때문에 표현하는 내용이 서로 달라지게 된다. 예를 들어, <그림 2>에서 정의한 생산



(a) 생산부서의 인스턴스 뷰 (b) 판매부서의 인스턴스 뷰

<그림 6> 생산 및 판매 계획 모형에 대한 인스턴스 뷰의 예.

및 판매 계획 모형의 입력 데이터 중 생산비용과 생산능력 등은 생산부서에서 입력할 수 있는 데이터들이며, 제품에 대한 시장수요 데이터는 판매부서에서 미래의 시장수요를 예측하여 입력할 수 있는 데이터이다. 한편, 생산 부서에서는 <그림 2>와 같은 모형을 통해 미래의 생산계획을 알기를 원하는 반면에 판매부서에서는 예측된 시장수요에 따른 제품의 공장으로부터의 인도 계획과 판매 계획을 알기를 원할 것이다. 따라서 하나의 공유되는 모형에 대한 인스턴스 뷰들은 부서의 업무에 따라 서로 다른 형태를 가져야 하며 부서간 협동적 업무를 지원하는 모형관리시스템은 이러한 요구를 만족시켜 줄 수 있어야 한다. <그림 6>은 그림 2의 생산 및 판매 계획 모형에 대한 인스턴스 뷰의 예로서 (a)는 생산 부서에서 사용하는 인스턴스 뷰를 나타내며, (b)는 판매 부서에서 사용하는 인스턴스 뷰를 나타낸다.

위의 그림에서 보는 바와 같이 생산부서에서는 제품에 대한 생산비용과 생산능력을 모형의 인스턴스로 입력하게 되며, 모형의 실행결과로 제품의 생산계획을 얻게 된다 <그림 6의 (a)>. 한편, 판매부서에서는 제품별 시장수요의 예측치와 보관비용을 모형의 인스턴스로 입력하게 되며, 모형의 실행 결과로 제품의 인수 및 판매 계획을 얻게 된다 <그림 6의 (b)>. 이 때, 두 부서는 조직 내의 모형베이스에 저장되어 있는 동일한 모형을 사용하게 되며, 한 부서의 인스턴스의 변화는 다른 부서의 결과값에 영향를 주게 된다. 예를 들어, 생산부서에서 제품에 대한 생산비용을 변화시키게 되면, 제품의 생산계획이 변하게 되어 <그림 6> (a)의 생산부서의 인스턴스 뷰의 결과 값이 변하게 된다. 한편, 이러한 변화는 제품의 인수 및 판매 계획에도 영향을 주게 되어, <그림 6> (b)의 판매부서의 인스턴스 뷰의 결과 값 또한 변하여야 한다.

모형을 여러 부서의 사용자가 공유하고 있는 상황에서 이와 같이 모형의 구조 또는 인스턴스가 변화하는 경우, 각각의 사용자가 모형베이스에 저장되어 있는 모형에 대한 내용과 불일치하는 사용자 뷰를 가질 수 있는 가능성이 있으며, 이는 업무 생산성을 저하시키는 중요한 요인이 될 수 있다. 따라서 모형의 구조의 변화와 인스턴스의 변화는 관련된 사용자에게 즉각적으로 통보될 수 있어야 한다. 특히, 모형의 인스턴스의 변화는 각각의 부서가 가지고 있는 해당 모형의 인스턴스 뷰에만 영향을 미치게 되지만 모형의 구조의 변화는 모형의 구조적 뷰 뿐만 아니라 해당 모형을 사용하고 있는 인스턴스 뷰에도 영향을 미치게 된다. <표 1>은 이러한 모형의 변화와 이에 따른 사용자 뷰에의 영향을 정리한 것이다.

<표 1> 모형의 변화가 미치는 영향

영향 / 모형의 변화	사용자 뷰	
	인스턴스 뷰	구조적 뷰
인스턴스의 변화	영향 있음	영향 없음
구조의 변화	영향 있음	영향 있음

따라서, 모형의 인스턴스에 변화가 발생하는 경우에는 변화된 새로운 인스턴스를 적용하여 모형을 다시 실행하고, 그 결과를 여러 부서에 산재하는 인스턴스 뷰로 통보하여야 한다. 한편, 모형의 구조에 변화가 발생한 경우에는 새로운 모형의 구조를 각각의 구조적 뷰에 통보하여야 하며, 새로운 모형의 구조에 기존의 인스턴스와 필요한 경우 새롭게 입력 받은 인스턴스를 적용하여 모형을 다시 실행하고 그 결과를 해당 인스턴스 뷰로 통보하여야 한다. 이러한 문제를 효과적으로 해결하기 위하여 다음 절에서 이 논문에서 제안하고자 하는 모형의 변화에 따른 사용자 뷰 자동갱신 기법에 대하여 설명한다.

Ⅳ. 변화 통보를 위한 기본적 구성체

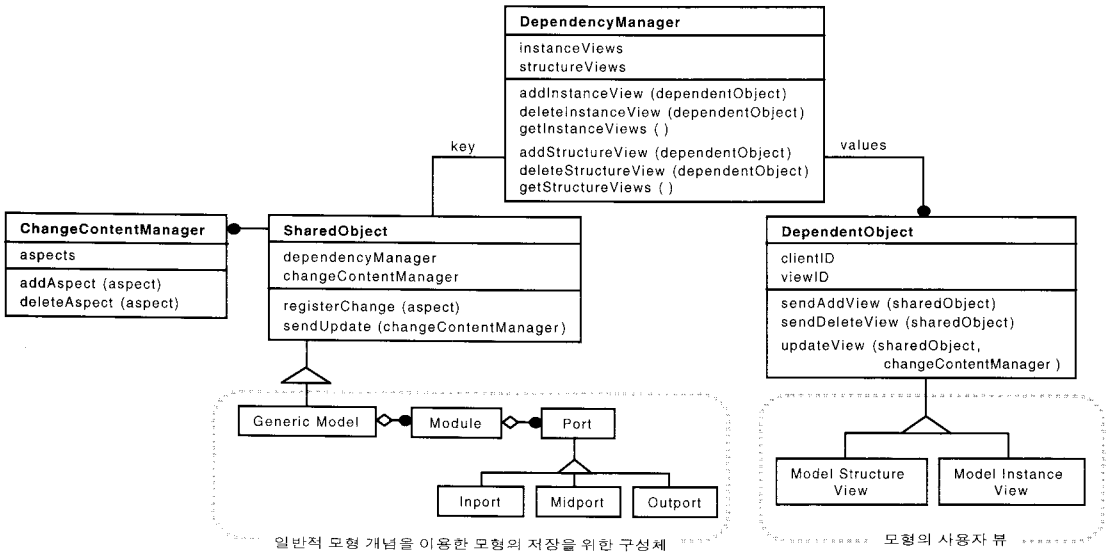
본 논문에서 제안하고자 하는 사용자 뷰의 자동 갱신을 위한 변화통보 기법은 첫째, 객체지향

구성체들을 이용하여 공유되어지는 객체 (모형베이스에 저장되어 있는 모형들)와 이들 공유객체의 변화에 영향을 받는 객체들 (여러 부서에서 생성되어지는 사용자 뷰들) 간의 종속관계를 관리하고, 둘째, 공유객체에 변화가 발생한 경우 변화내역을 종속객체에게로 즉각적으로 통보하여 이를 반영하기 위한 프로세스를 지원한다. 먼저, 모형의 변화 발생시 영향을 받는 사용자 뷰로의 변화통보를 위한 기본적인 구성체들은 다음과 같다.

- 공유객체 (SharedObject)는 여러 사용자들에 의해서 공유되어 사용되는 객체로서 데이터베이스와 같은 영속적 메모리 (persistent memory)에 저장되는 객체이다. <그림 3>과 같은 모형베이스 안에 저장되어 있는 수리적 모형들은 공유객체의 예가 된다.
- 종속객체 (DependentObject)는 영속적 메모리에 저장되어 있는 공유객체를 사용자의 관점이나 필요에 맞게 표현하는 객체로서 일시적 메모리 (transient memory)에 생성되는 객체이고 공유객체의 변화에 영향을 받는다. <그림 4>나 <그림 6>과 같은 여러 부서의 클라이언트에 생성되는 모형의 구조적 뷰나 인스턴스 뷰들은 종속객체의 예가 된다.
- 종속성관리자 (DependencyManager)는 공유객체와 종속객체 간의 종속관계를 관리하기 위한 관리도구 객체로서 한 공유객체에 대해 어떠한 종속객체들이 현재 존재하는가를 관리하며, 종속객체의 생성과 소멸에 따라 이에 대한 사항을 동적으로 등록 및 삭제하게 된다.
- 변화내역관리자 (ChangeContentManager)는 공유객체에 변화가 발생한 경우 어떠한 변화가 발생하였는지 그 내역을 저장해 두었다가 공유객체에 대한 변화가 정상적으로 완료된 경우에 이를 종속객체에게 통보하기 위해 사용하는 객체이다. 공유객체는 데이터베이스 상에 존재하는 객체로서 여러 사용자에 의해서 동시적으로 접근 되어지기 때문에 여러 작업간에 충돌이 발생할 수 있고, 따라서 공유객체를 변화 시키기 위한 연산이 성공적으로 끝나지 못할 가능성이 있다. 그러므로 종속객체에게로의 변화통보는 공유객체에 대한 변화작업이 성공적으로 완료된 경우에만 이루어져야 하며, 이를 위해서 공유객체에 발생하는 변화는 일단 변화내역관리자에 저장되어진다. 최종적으로 데이터베이스에 존재하는 공유객체를 변화시킨 트랜잭션 (transaction)이 성공적으로 완료 (commit)된 경우에만 변화내역관리자의 내용이 종속객체에게로 통보되며, 공유객체를 변화시키기 위한 트랜잭션이 실패로 끝날 경우 (abort)에는 변화내역관리자의 내용은 그대로 삭제되게 된다.

<그림 7>은 변화통보를 위한 기본적인 구성체들과 이들이 가지는 관계 및 연산구조를 OMT 방법 [Rumbaugh, 1991]을 이용하여 나타낸 것이다. OMT 방법에서 객체 모델링을 위한 표시법에 의하면 클래스를 나타내는 사각형은 세 부분으로 나누어지게 되는데, 가장 위의 부분은 클래스의 이름을 나타내고, 가운데 부분은 클래스의 속성 (attribute)을, 그리고 가장 아래 부분은 그 클래스가 가지는 연산 (operation)들을 나타낸다.

<그림 7>에서 보는 바와 같이 공유객체를 나타내는 SharedObject 클래스는 자신의 변화에 영향을 받는 종속객체들이 어떤 것들이 있는가를 관리하기 위하여 dependencyManager 라는 속성을 가지며, 공유객체가 변화한 경우 그 변화내역을 관리하기 위하여 changeContentManager 라는 속성을 갖는다. 그리고 이들은 각각 종속성관리자인 DependencyManager 클래스와 변화내역관리자인 ChangeContentManager 클래스와 연결이 된다. 한편, DependencyManager 클래스



<그림 7> 변화통보를 위한 기본적인 구성체에 대한 객체 모델링

는 구체적인 종속객체들, 즉, 모형에 대한 사용자 뷰들을 관리하는데, 3절에서 설명한 바와 같이 모형의 사용자 뷰는 인스턴스 뷰와 구조적 뷰로 나눌 수 있으며, 모형의 변화가 이 두 가지 사용자 뷰에 미치는 영향이 각각 다르므로, 이를 구분하여 관리하여야 할 필요성이 있다. 따라서, DependencyManager 클래스는 종속되어 있는 인스턴스 뷰가 어떠한 것이 있는가를 나타내는 instanceViews 속성과 종속되어 있는 구조적 뷰가 어떤 것이 있는가를 나타내는 structureViews 속성을 가지며, 이들은 구체적인 종속객체들과 연결이 된다. 종속객체를 표현하는 Dependent Object 클래스는 구체적인 사용자 뷰를 나타내며, 모형에 대한 사용자 뷰는 실제로 각 부서에서 사용하는 클라이언트에 생성된다. 따라서 DependentObject 클래스는 사용자 뷰가 어느 클라이언트에 생성되었는가를 나타내는 clientID 속성을 가지며, 해당 클라이언트 내에서도 사용자 뷰가 여러 개 있을 수 있으므로 세부적으로 어느 사용자 뷰인지를 나타내는 viewID 속성을 갖는다.

종속객체인 사용자 뷰는 사용자의 필요에 따라 생성과 소멸이 이루어 지며, 이는 각각 DependentObject 클래스의 sendAddView() 연산과 sendDeleteView() 연산을 통해 공유객체로 알려지게 된다. 공유객체와 종속객체 간의 종속관계를 관리하는 DependencyManager 클래스는 이러한 종속객체의 생성과 소멸을 동적으로 관리하기 위하여 기본적으로 세가지 타입의 연산을 제공하는데, (1) 종속객체가 생성된 경우 이에 대한 등록을 위한 연산 (addInstanceView(), addStructureView()), (2) 종속객체가 소멸된 경우 이에 대한 삭제를 위한 연산 (deleteInstanceView(), deleteStructureView()) (3) 현재 생성되어 있는 종속객체에 대한 검색을 위한 연산 (getInstanceViews(), getStructureViews()) 등이 있다.

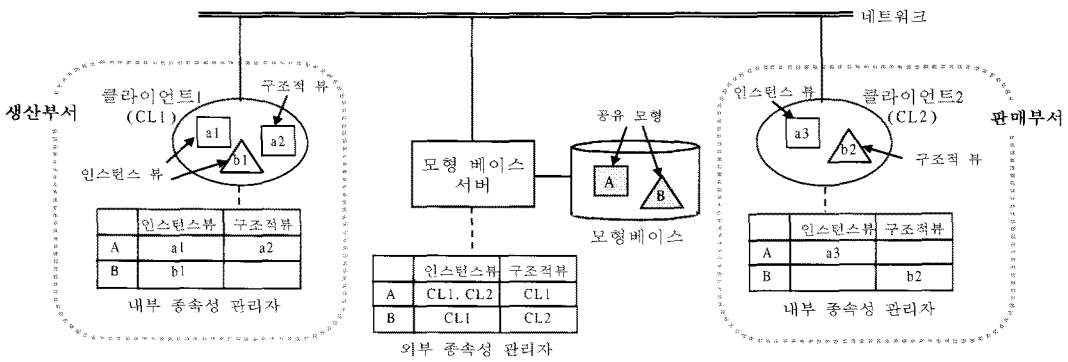
한편, 공유객체에 변화가 발생한 경우에는 SharedObject 클래스의 registerChange() 연산을 통해 ChangeContentManger 클래스 내의 aspects 속성에 그 내역이 등록되며, 공유객체를 수정한 연산이 정상적으로 종료되면, SharedObject 클래스의 sendUpdate() 연산이 내부적으로 실행되게 된다. sendUpdate() 연산은 현재 등록되어 있는 종속객체들을 검색하여 종속객체를 가지고 있는

클라이언트 들에게 공유객체의 변화사실을 통보해 주게 되며, 이를 통보 받은 종속객체들은 DependentObject 클래스 내의 updateView() 연산을 통하여 공유객체의 변화내역을 사용자 뷰에 반영하게 된다. 공유객체의 변화에 따른 사용자 뷰로의 변화통보 과정은 다음 절에서 보다 자세히 논의하기로 한다.

지금까지 설명한 사용자 뷰의 자동갱신을 위한 객체지향 구성체들은 공유객체와 종속객체의 종속관계를 관리하고 공유객체에 변화가 발생한 경우 이에 대한 종속객체로의 통보를 위한 일반적이고 기본적인 기능들을 가지고 있다. 이러한 변화통보를 위한 기본적인 기능들을 모형관리시스템에 적용하기 위하여 객체지향 데이터베이스가 가지는 중요한 성질 중의 하나인 계승 (inheritance) 메커니즘을 이용한다. <그림 7>에서 보는 바와 같이, 수리적 모형들을 모형베이스 내에 저장하기 위하여 사용하는 일반적 모형 개념을 이용한 구성체들은 SharedObject 클래스의 하위 클래스 (sub-class)로 정의되며, 각 부서의 사용자들이 모형을 이용하기 위하여 사용하게 되는 사용자 뷰들은 DependentObject 클래스의 하위 클래스로 정의된다. 따라서 상위 클래스 (super-class)인 SharedObject 클래스와 DependentObject 클래스가 가지는 속성과 연산들을 구체적인 하위 클래스들에서 모두 이용할 수 있으며, 각각의 구체적인 모형들과 사용자 뷰들은 변화통보와 이에 대한 반영을 위한 기능들을 수행할 수 있다. 이와 같이 모형관리 기능을 수행하는 구성체들과 변화통보 및 반영 기능을 수행하는 구성체들을 구조적으로 분리시켜 주고, 계승 메커니즘을 통해 연결시켜 줌으로써 첫째, 사용자 뷰의 자동갱신을 위한 복잡한 기능들을 모형관리시스템 내에 구현함에 있어서 단지, 변화통보 및 반영 기능을 가지고 있는 구성체들을 계승 받음으로써 원하는 기능을 시스템 내에 구현 할 수 있으며, 둘째, 변화통보를 위한 일반적인 구성체들을 본 논문에서 다루고 있는 모형관리시스템 이외에 사용자 뷰의 자동갱신 기능을 필요로 하는 다른 응용 프로그램들에도 손쉽게 적용할 수 있는 장점을 가지게 된다.

V. 부서간 협동적 모형관리를 위한 각 부서로의 변화통보 기법

앞 절에서 설명한 사용자 뷰 자동갱신을 위한 기본적인 메커니즘을 모형을 이용하는 부서들이 회사 내에 지역적으로 분산되어 있는 분산 컴퓨팅 환경에 적용하기 위해서는 몇 가지 기능적 확장을 필요로 한다. 먼저, 공유객체와 종속객체 간의 종속관계를 관리하기 위한 종속성관리자의 경우, 분산 컴퓨팅 환경에서 공유객체의 변화시 종속객체로의 통보를 효율적으로 수행하기 위하여 이를 외부종속성관리자 (external dependency manager)와 내부종속성관리자 (internal dependency manager)로 나누어 2단계로 종속성을 관리하는 기법을 사용한다. 첫번째 단계에서 외부종속성관리자는 각각의 공유객체에 대하여 프로세스 외부에 존재하는 종속객체들을 관리하는 다른 프로세스들이 무엇인지를 관리하며, 두 번째 단계에서 내부종속성관리자는 각각의 공유객체에 대하여 프로세스 내부에 존재하는 종속객체들이 무엇인지를 관리한다. 다시 말해서, 외부종속성관리자는 모형베이스 서버에서 자신이 관리하는 모형들과 그 모형의 사용자 뷰들을 가지고 있는 종속된 클라이언트들이 어떤 것이 있는가를 관리하며, 내부종속성관리자는 하나의 클라이언트 내에서 공유객체인 모형과 그 클라이언트 내에 생성되어 있는 해당 모형에 대한 사용자 뷰들과의 종속관계를 관리한다. 따라서 변화통보를 위한 종속성을 관리하는데 있어서 서버와 클라이언트는 각각 자신이 필요로 하는 정보만을 둘 간에 분리하여 관리할 수 있으며, 모형의 변화발

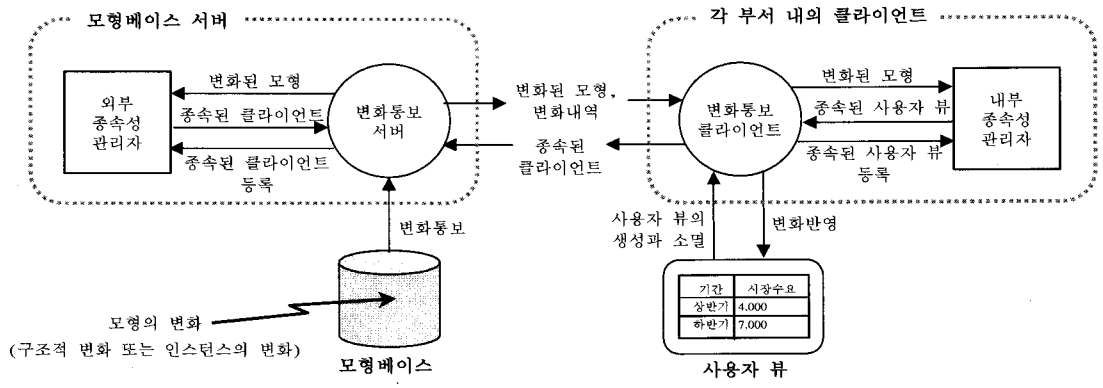


<그림 8> 내부종속성관리자와 외부종속성관리자

생시 하나의 클라이언트에 두개 이상의 종속된 사용자 뷰가 생성되어 있는 경우에도 해당 클라이언트에 한번만 변화내역을 통보하면 되는 장점을 갖는다. 특히, 이는 모형을 이용하는 부서의 클라이언트의 수가 늘어나고, 한 클라이언트당 사용자 뷰의 생성이 많아짐에 따라 더욱 중요해 진다.

<그림 8>은 분산 컴퓨팅 환경에서 여러 부서에 의해 공유되고 있는 모형과 이러한 모형을 이용하기 위해 각 부서에 생성된 사용자 뷰간의 종속관계를 2가지 종속성관리자를 이용하여 관리하는 방법의 예를 보여주고 있다. 모형베이스 내의 회색으로 표시된 삼각형과 사각형은 회사에서 공유되고 있는 수리적 모형들을 나타내며, 타원으로 표시된 클라이언트 내의 흰색의 삼각형과 사각형은 같은 모양으로 표시된 모형에 대한 사용자 뷰를 나타낸다. 그림의 예에서는 공유객체인 모형 A (사각형)와 모형 B (삼각형)가 모형베이스 내에 저장되어 있으며, 이에 대한 사용자 뷰가 생산부서의 클라이언트 1 (CL1)과 판매부서의 클라이언트 2 (CL2)에 생성되어 있다. 첫 번째 단계에서 모형베이스 서버는 공유되는 모형과 종속객체인 사용자 뷰를 가지고 있는 클라이언트 간의 종속관계를 관리하기 위하여 외부종속성관리자를 가지고 있으며, 두 번째 단계에서 각 클라이언트는 내부적으로 모형과 사용자 뷰 간의 종속관계를 관리하기 위하여 내부종속성관리자를 가지고 있다. 예를 들어, 그림에서 모형 A의 경우, 이에 대한 인스턴스 뷰가 클라이언트 1과 클라이언트 2에 생성되어 있으며, 구조적 뷰가 클라이언트 1에 생성되어 있다. 그러므로 모형베이스 서버가 관리하고 참조하게 되는 외부종속성관리자에는 인스턴스 뷰가 CL1과 CL2에 있으며, 구조적 뷰가 CL1에 있다는 사실을 관리한다. 한편, 클라이언트 1과 클라이언트 2는 각각 자신의 내부종속성관리자 내에 모형 A에 대하여 클라이언트 1은 a1이라는 인스턴스 뷰와 a2라는 구조적 뷰를 가지고 있으며, 클라이언트 2는 a3라는 인스턴스 뷰를 가지고 있다는 사실을 관리한다.

분산 컴퓨팅 환경에서 모형베이스에서 발생하는 모형의 변화를 지역적으로 분산되어 있는 여러 클라이언트에 통보해 주기 위해서는 서버와 클라이언트 내에 각각 이러한 기능을 전담하는 프로세스가 필요하며, 변화통보서버 (change notification server) 프로세스와 변화통보클라이언트 (change notification client) 프로세스가 이러한 역할을 담당한다. 변화통보서버는 모형에 변화가 발생한 경우 서버 쪽에서 변화통보 과정을 처리하기 위한 프로세스이며, 변화통보클라이언트는 클라이언트 쪽에서 변화통보 과정을 처리하기 위한 프로세스이다. 그리고 이들 두 가지 변화통보를 위한 프로세스들은 서버와 클라

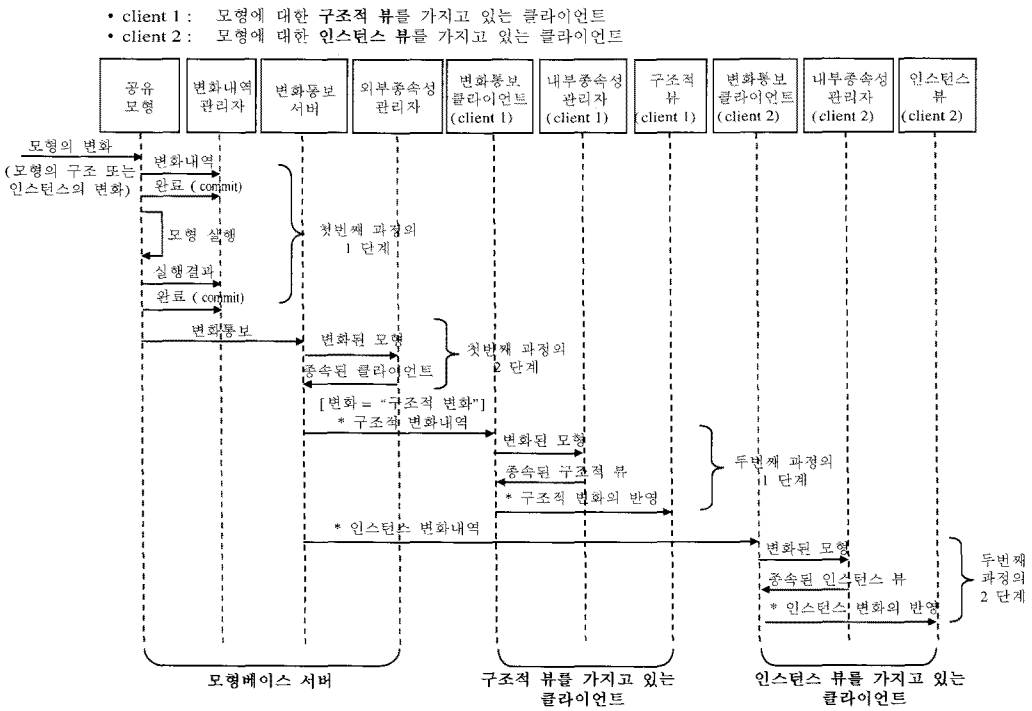


<그림 9> 변화통보서버와 변화통보클라이언트

이언트 간의 통신을 위한 여러 가지 연산들을 지원한다. <그림 9>는 2가지 변화통보 프로세스들과 이들이 관리하는 2가지 종속성관리자들 간의 관계를 나타낸 것으로 화살표는 변화통보 과정에서 일어나게 되는 사건 (event)들을 나타낸 것이다. 그림에서 보는 바와 같이, 모형베이스 서버 쪽의 변화통보 업무를 담당하게 되는 변화통보서버는 외부에 존재하는 종속된 클라이언트들이 무엇인지를 알아내기 위하여 외부종속성관리자를 참조하게 되며, 클라이언트 쪽의 변화통보 업무를 담당하게 되는 변화통보클라이언트는 각 클라이언트 내부에 존재하는 사용자 뷰들이 구체적으로 무엇인지를 알아내기 위하여 내부종속성관리자를 참조하게 된다.

공유되고 있는 모형에 변화가 발생하게 되면, 클라이언트와 서버 간에 미리 정해진 규칙에 따라 이러한 변화를 사용자 뷰에 반영하기 위한 과정이 진행되어야 하며, 이러한 작업은 실질적으로 변화통보서버와 변화통보클라이언트 간의 통신을 통해서 이루어지게 된다. 부서간 협동적 모형관리시스템에서 사용자 뷰의 자동갱신을 위한 전체 프로세스는 크게 2개의 과정으로 나누어 지는데, 첫번째 과정은 모형의 구조 또는 인스턴스에 변화가 발생한 경우 이를 각 클라이언트로 통보하기 위한 서버쪽 처리과정을 나타내며, 두 번째 과정은 통보 받은 변화내역을 사용자 뷰에 반영하기 위한 클라이언트쪽 처리과정을 나타낸다. <그림 10>은 UML (Unified Modeling Language)의 순서 다이어그램 (sequence diagram) [Booch, 1999]을 이용하여 그 내부적인 처리순서를 표현한 것이다. 순서 다이어그램은 객체간의 상호작용을 표현하기 위하여 서로 주고 받는 메시지들을 순서적으로 표현하는데 적합한 구조를 가지고 있다. 순서 다이어그램에서 각 객체들은 사각형으로 표시되고 객체들 간에 주고 받는 메시지는 화살표로 표현되며, 시간의 흐름은 위에서 아래방향 순서이다. 또한 순서 다이어그램에서 메시지가 특정한 조건을 만족시킬 때에만 전달되는 경우는 그 조건을 [ ] 안에 표현하며, 메시지를 받게 되는 객체가 두개 이상인 경우에는 메시지에 별표 (*)를 하여 둘 이상의 메시지 수신자 집합에 모두 메시지가 전달됨을 표시한다.

변화통보 과정 중 서버쪽 처리 과정을 나타내는 첫 번째 과정의 처리단계는 다음과 같다. 먼저, 1 단계에서 모형베이스에 저장되어 있는 공유 모형의 구조가 변하거나 인스턴스가 변하게 되면, 변화된 내용은 변화내역관리자 내에 쌓이게 되며, 최종적으로 데이터베이스에 저장되어 있는 모형을 변화 시킨 트랜잭션이 정상적으로 완료 (commit)된 경우에 다음 단계로 넘어가게 된다. 만일 이 트랜잭션이 중지 (abort)되면 그 내용은 사용자 뷰로 통보되지 않는다. 모형에 대



<그림 10> 변화통보 과정에 대한 순서 다이어그램 (Sequence Diagram)

한 변화가 끝나게 되면, 변화된 모형을 실행하게 된다. 변화된 모형의 실행은 2절에서 설명한 일반적 모형 (generic model)의 출력포트의 인스턴스에 변화를 가져오게 되며, 이 또한 변화내역관리자 내에 쌓이게 된다. 변화된 모형의 실행결과 내용에 대한 수정작업을 수행한 트랜잭션 역시 완료 상태로 끝나게 되면, 2 단계로 넘어가게 된다. 2 단계에서는 변화통보서버가 공유 모형으로부터 변화를 통보 받게 되면, 변화된 모형에 종속된 사용자 뷰들을 가지고 있는 클라이언트들이 어떤 것이 있는가를 알아내기 위하여 자신이 관리하는 외부종속성관리자를 참조하여 구조적 뷰를 가지는 클라이언트들과 인스턴스 뷰들을 가지는 클라이언트들을 찾아낸다.

두 번째 과정은 찾아진 클라이언트들로 변화내역을 통보하고 이를 사용자 뷰에 반영하는 과정으로 처음 공유 모형의 변화가 모형의 구조에 대한 변화인지, 인스턴스에 대한 변화인지에 따라 달라지게 된다. 다시 말해서, 인스턴스에 대한 변화인 경우에는 각 부서에 존재하는 인스턴스 뷰로만 변화된 인스턴스의 내용 (모형의 실행 결과 포함)을 통보하면 되지만, 모형의 구조에 대한 변화인 경우에는 먼저 각 부서에 생성되어 있는 구조적 뷰로 변화를 통보하고, 해당 모형의 인스턴스 뷰가 생성되어 있는 경우에는 변화된 모형을 실행한 결과를 포함한 인스턴스의 변화를 해당 인스턴스 뷰로 통보해 주어야 한다. 이 때, 구조적 뷰들이나 인스턴스 뷰들은 앞에서 설명한 바와 같이 여러 가지 종류가 존재할 수 있으며, 통보된 모형의 변화 내용은 각 사용자 뷰들의 형식에 따라 적절히 반영되어야 한다.

두 번째 과정의 1 단계는 모형의 구조에 대한 변화가 발생한 경우에만 거치게 되는 단계로서 구조적 뷰를 가지는 클라이언트의 변화통보클라이언트로 구조적 변화의 내용을 통보하게 된다.

모형의 구조적 변화를 통보 받은 변화통보클라이언트는 구체적인 구조적 뷰들을 찾기 위해서 해당 클라이언트에서 관리하는 내부종속성관리자를 참조하게 되며, 찾아진 구조적 뷰들에 모형의 구조적 변화를 적절하게 반영하게 된다. 한편, 2 단계는 인스턴스 뷰를 가지는 클라이언트로의 변화 통보 과정이며, 이는 모형의 구조에 대한 변화와 인스턴스에 대한 변화 두 가지 경우에 모두 거치게 되는 단계이다. 먼저, 변화통보서버로부터 인스턴스 변화 내역을 통보 받은 인스턴스 뷰를 가지는 클라이언트의 변화통보클라이언트는 자신이 관리하는 내부종속성관리자를 참조하여 구체적인 인스턴스 뷰들을 찾게 되고, 그 결과에 따라 구체적인 각각의 인스턴스 뷰들을 수정하게 된다.

이상과 같은 변화 통보과정을 통해 여러 부서에서 동일한 모형을 이용하여 업무를 수행하는 부서간 협동적 모형관리시스템에서 업무환경의 변화에 따라 발생하게 되는 모형의 구조 또는 인스턴스에 변화와 그 변화에 따른 모형의 실행결과는 관련된 모든 부서의 사용자에게로 즉각적이고 자동적으로 통보될 수 있다.

Ⅵ. 결 론

부서간 협동적 모형관리시스템은 회사 내에서 사용되는 수리적 모형들을 여러 부서가 공유할 수 있도록 지원하며, 각 부서는 업무의 필요에 따라 동일한 모형에 대한 각기 다른 사용자 뷰들을 이용하여 모형들을 사용하게 된다. 한편, 모형베이스 내에 저장되어 있는 모형들은 업무환경의 변화와 모형 자체의 개선 요구로 인하여 모형의 구조 또는 모형의 인스턴스가 계속적으로 변화하게 된다. 이러한 환경 하에서 각 부서의 사용자들은 자신이 사용하는 모형의 사용자 뷰가 항상 모형베이스에 저장되어 있는 모형의 내용과 일치하기를 원하게 된다. 따라서 모형이 변화한 경우에 그 변화내역을 각 부서에 즉각적이고 자동적으로 반영할 수 있는 기법이 필요하다. 본 논문에서는 부서간 협동적 모형관리시스템 환경에서 공유되는 모형의 구조나 인스턴스에 변화가 발생한 경우 이를 각 부서의 사용자 뷰로 즉각적으로 통보하여 항상 모형베이스의 내용과 일치하는 사용자 뷰를 유지할 수 있는 기법을 제안하였다.

본 논문에서 제안하고자 하는 부서간 협동적 모형관리시스템에서 사용자 뷰의 자동갱신을 위한 기법이 가지는 특징들은 다음과 같다. 첫째, 모형베이스에 저장되어 있는 모형들을 조직 내의 여러 부서에 있는 사용자들이 자신의 사용자 뷰를 이용하여 동시에 접근하여 이용할 수 있다. 이때, 각 부서들은 자신의 업무와 관련하여, 자신의 관점에서 모형을 이용하게 되며, 따라서 각 부서가 가지는 모형에 대한 사용자 뷰들은 각 부서의 업무를 지원해 줄 수 있도록 서로 다른 형태를 가지게 된다. 이러한 사용자 뷰들은 크게 모형에 대한 입력 데이터와 실행결과를 보여주는 인스턴스 뷰와 모형에 대한 구조를 보여주는 구조적 뷰로 구성되며, 각각의 부서들은 이러한 사용자 뷰들을 이용하여 자신의 부서와 관련된 입력 데이터를 제공하고 실행결과를 볼 수 있으며, 필요한 경우 모형의 구조를 보거나 수정할 수 있다.

둘째, 한 부서가 발생시키는 모형에 대한 변화는 관련된 다른 부서에게로 즉각적이고 자동적으로 통보된다. 각 부서에서 모형에 대한 사용자 뷰를 생성시키는 시점에 공유된 모형과 각 부서에 흩어져 있는 사용자 뷰들 간의 종속관계가 등록 관리되며, 모형에 변화가 발생된 경우에는 변화된 내용과 변화된 내용에 따라 달라지게 되는 모형의 실행결과가 각 부서로 통보되게 된다. 이때, 모형에 대한 변화는 사용자 뷰에 미치는 영향에 따라 모형의 인스턴스에 대한 변화와 모형의 구조에 대한 변화로 나누어 관리되며, 전자의 경우에는 인스턴스 뷰에게만 영향을 미치는 반면, 후자의 경우에는 인스턴스 뷰와 구조적

뷰 모두에게 영향을 미치게 된다.

셋째, 통보된 변화 내용은 각 부서에 생성되어 있는 사용자 뷰의 표현 내용과 형식에 따라 다르게 반영되어 진다. 공유되어진 모형에 대한 사용자 뷰들은 각 부서의 업무 내용과 사용자 개인의 이용 목적에 따라 여러 가지 표현 형태가 존재하며, 통보 받은 모형에 대한 변화 내역은 생성되어진 사용자 뷰의 종류에 따라 적절히 반영된다. 이는 변화통보와 반영을 위한 일반적인 기능들을 가지고 있는 객체지향 구성체들과 모형의 내용에 대한 표현을 담당하는 개별 사용자 뷰의 구현을 위한 구성체들을 서로 분리하여 구현하고, 이들 둘을 계승 메커니즘을 이용하여 연결함으로써 가능하다. 상위 클래스들은 각 모형의 사용자 뷰들의 공통되는 사항인 모형과의 종속관계 등록과 해제 그리고 변화 발생시 변화통보와 접수의 역할을 수행하며, 구체적인 사용자 뷰로의 반영은 이러한 클래스의 속성을 계승받는 하위 클래스에서 구현된다. 따라서 각 사용자 뷰들은 공유되는 모형과의 일관성 유지를 위한 변화 통보의 일반적인 기능과 변화 발생시 이에 대한 반영을 위한 자신의 표현 형태에 따른 특화된 반영 기능을 모두 가질 수 있게 된다.

넷째, 모형베이스 내에 다양한 형태의 수리적 모형들을 일관성 있는 방법으로 관리하기 위하여 일반적 모형 개념을 이용하며, 모형관리 기능과 사용자 뷰의 자동갱신을 위한 기능을 하나의 형식론 (formalism) 안에서 통합하기 위하여 객체지향 데이터베이스 시스템 (ODBMS)을 사용한다. 특히, 객체지향 데이터베이스를 기반 플랫폼으로 사용함으로써 제안하고자 하는 부서간 협동적 모형관리시스템은 다중 사용자가 동시에 모형베이스에 접근하여 공유자원인 모형들을 이용하기 위한 트랜잭션 관리기능을 지원해 줄 수 있으며, 모형의 변화 발생시 이에 대한 사용자 뷰로의 통보기능을 안정적으로 수행할 수 있는 환경을 제공해 준다.

본 논문에서 제안하는 모형에 대한 다중 사용자 뷰 지원 기능과 모형의 변화 통보 및 반영 기능을 가지는 협동적 모형관리시스템에 대한 원형 (prototype) 시스템은 Windows NT를 기반으로 객체지향 데이터베이스 시스템인 ObjectStore [Lamb, 1991] 위에 C++ 프로그래밍 언어를 이용하여 구현되었으며, 향후 연구 과제로는 첫째, 변화통보시 업무 분야에 따른 복잡한 통보 규칙을 수용하기 위한 규칙베이스 기능에 관한 연구와 둘째, 변화통보기법을 다중 기업간의 생산, 물류, 판매 기능을 연결하는 가상 기업 (virtual organization)과 같은 분야에 적용하는 문제를 연구하고 있다.

〈참 고 문 헌〉

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

[2] Blanning, R., A Relational Framework for Join Implementation in Model Management, *Decision Support Systems*, Vol. 1, 1985, pp. 69-82.

[3] Blanning, R., Model Management Systems: An Overview, *Decision Support Systems*, Vol. 9, 1993, pp. 9-18.

[4] Booch, G., Rumbaugh, J., and Jacobson, I., *The Unified Modeling Language*, User Guide, Addison Wesley, 1999.

[5] Dolk, D., Model Management and Structured Modeling: The Role of an Information Resource Dictionary System, *Communications*

on ACM, Vol. 31, 1988, pp. 704-718.

[6] Dolk, D., An introduction to model integration and integrated modeling environments, *Decision Support Systems*, Vol. 10, 1993, pp. 249-254.

[7] Fourer, R., Gay, D., and Kernighan, B., A Mathematical Programming Language, *Management Science*, Vol. 36, No. 5, 1990, pp. 519-554.

[8] Geoffrion, A., An Introduction to Structured Modeling, *Management Science*, Vol. 33, 1987, pp. 547-588.

[9] Geoffrion, A., The SML Language for Structured Modeling: Level 1 and 2, *Operations Research*, Vol. 40, 1992, pp. 38-57.

[10] Greenberg, H. J. and Murphy, F. H., Views of Mathematical Programming Models and Their Instances, *Decision Support Systems*, Vol. 13, 1995, pp. 3-34.

[11] Huh, S.-Y., Modelbase Construction with Object-Oriented Constructs, *Decision Sciences*, Vol. 24, 1993, pp. 409-434.

[12] Huh, S.-Y. and Chung, Q. B., A Model Management Framework for Heterogeneous Algebraic Models: Object-oriented Database Management Systems Approach, *Omega*, Vol. 23, 1995, pp. 235-256.

[13] Huh, S.-Y. and Rosenberg, D. A., Dependency Maintenance for Collaborative Computing Environment, *Journal of Systems and Software*, Vol. 34, No. 3, 1996, pp. 231-246.

[14] Lamb, C., Landis, G., Orenstein, J., and Weinreb, D., The ObjectStore Database System, *Communications of the ACM*, Vol. 34, No. 10, 1991, pp. 50-63.

[15] Le Claire, B., and Sharda, R., An Object-Oriented Architecture for Decision Support Systems, *Proceedings of the 1990 International Society for Decision Support Systems Conference*, 1990, pp. 567-586.

[16] Lenard, M., Representing Models as Data, *Journal of Management Information Systems*, Vol. 11, 1986, pp. 36-48.

[17] Liang, T., A Graph-Based Approach to Model Management, *Proceedings of 7th International Conference on Information Systems*, 1986, pp. 136-151.

[18] Liang, T., Model Management for Group Decision Support, *MIS Quarterly*, 1988, pp. 667-680.

[19] Ma, P. C., Murphy, F. H., and Stohr, E. A., A Graphics Interface for Linear Programming, *Communications of the ACM*, Vol. 32, 1989, pp. 996-1012.

[20] Mannino, M., Greenberg, B., and Hong, S., Model Libraries: Knowledge Representation and Reasoning, *ORSA Journal of Computer*, Vol. 2, 1990, pp. 287-301.

[21] Martin, J., *Cybercorp: The New Business Revolution*, American Management Association, 1996.

[22] 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.

[23] Muhanna, W., An Object-Oriented Framework for Model Management and DSS Development. *Decision Support Systems*, 1995.

[24] Murphy, F. H., Stohr E. A., and Asthana, A., Representation Schemes for Mathematical Programming Models, *Management Science*, Vol. 38, 1992, pp. 964-991.

[25] Nunamaker, J., Applegate, L., and Konsynski, B., Computer-Aided Deliberation: Model Management and Group Decision Support, *Operations Research*, Vol. 36, 1988,

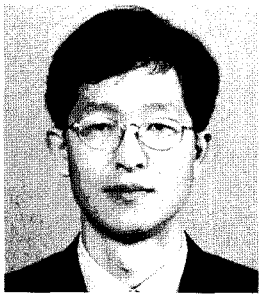
pp. 826-848.

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

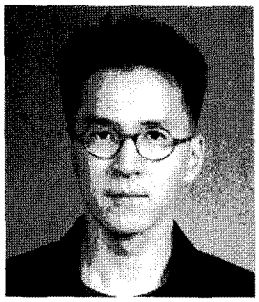
[27] Sprague, R. H., A Framework for the Development of DSS, *MIS Quarterly*, 1980.

◈ 이 논문은 1999년 5월 7일 접수하여 1차 수정을 거쳐 2000년 1월 3일 게재확정 되었습니다.

◆ 저자소개 ◆



허순영 (Huh, Soon-Young)
현재 한국과학기술원 테크노경영대학원 조교수로 재직중이다. 서울대학교 전자공학과에서 공학사(1981), 한국과학기술원 경영과학과에서 공학석사(1983), 미국 UCLA 대학교 경영대학원에서 경영정보시스템 분야로 경영학 박사학위(1992)를 취득하였다. 주요 관심분야는 객체지향 기술을 기반으로 한 의사결정지원시스템, 지능형 금융정보시스템, 실시간 협동작업을 지원하는 그룹웨어시스템 등이다.



김형민 (Kim, Hyung-Min)
현재 한국과학기술원 테크노 경영대학원 박사과정에 재학중이다. 한국과학기술원 경영과학과에서 이학사(1993) 및 공학석사(1995)를 취득하였다. 주요 관심분야는 모형관리 시스템, 지식관리 데이터베이스 시스템, 객체지향 데이터베이스 기반 그룹협동 시스템 등이다.