BEBR

FACULTY WORKING PAPER NO. 1364

Development of a Knowledge-Based Model Management System

Ting-peng Liang

College of Commerce and Business Administration
Bureau of Economic and Business Research
University of Illinois, Urbana-Champaign

BEBR

FACULTY WORKING PAPER NO. 1364

College of Commerce and Business Administration

University of Illinois at Urbana-Champaign

May 1987

Development of a Knowledge-Based Model Management System

Ting-peng Liang, Assistant Professor
Department of Accountancy

ABSTRACT

Model management is an important but, as yet, poorly researched area in decision support systems. One difficult issue in developing a model management system is to build automatic modeling capabilities which can create ad hoc models by integrating existing models and provide advice regarding effective use of models. This paper presents a software architecture and a graph-based framework for developing such capabilities. The architecture consists of three major components: model utilization subsystem, modeling subsystem, and inference engine. The core of the system is the inference engine which applies the graph-based framework to drive the process of model integration and selection. The graph-based framework includes a graph-based model representation scheme and reasoning mechanisms for model integration and selection. The representation scheme represents a set of data as a node and a set of functions as an edge. Since a model can be decomposed into two sets of data (inputs and outputs) and a set of functions for converting data, it is represented as a combination of two nodes and one edge connecting the two nodes. Based on this scheme, mechanisms for model integration and selection are developed. These mechanisms enable a model management system to create ad hoc models automatically. A prototype implemented in PROLOG is also presented to demonstrate the graph-based framework.

INTRODUCTION

In the past decade, much effort has been spent developing decision support systems (DSSs) to support semi-structured or unstructured decisions. A DSS is usually composed of three major components: data management, model management, and user interface. Since good models can significantly improve the performance of human decision-making by facilitating understanding about the decision problem, examining more alternatives, or enhancing prediction (Little, 1970), a model management system (MMS) that supports the development of decision models and their subsequent use has been considered crucial to the success of DSSs (Alter, 1980, Bonczek, Holsapple, and Whinston, 1981b, Keen and Scott Morton, 1978, Sprague and Carlson, 1982).

Previous research in MMSs primarily focused on two issues: model base organization and model representation. A model base is a repository of decision models. On the one hand, because the model base and the data base are similar in many aspects, researchers have studied the application of data models, such as the relational model (Codd, 1970), to the development of MMSs (Blanning, 1982,1983,1984,1985, Donovan, 1976). On the other hand, some researchers concentrated on adopting knowledge representation techniques to represent models in the model base. The model representation schemes developed include SI-net (Elam, Henderson, and Miller, 1980), knowledge abstractions (Dolk, 1982, Dolk and Konsynski, 1984, Konsynski and Dolk, 1982), predicate calculus (Bonczek, Holsapple, and Whinston, 1980, 1981a), and frame-based systems (Watson, 1983). In developing an MMS, both a

model base organization for model storage and an appropriate technique for model representation are essential.

In addition to these two issues, however, it is very important for an MMS to have the following two capabilities:

1. Providing advice regarding effective use of existing models, and
2. Integrating existing models to support ad hoc decisions automatically.

The advising capability helps the user to figure out which model in the model base can be applied to solve a particular problem. The capability of model integration allows an MMS to create complicated models by integrating existing models in the model base. In this case, the models stored in the model base are not only stand-alone decision models but also building blocks for creating new models. It will substantially enhance the capability of an MMS to meet unanticipated requirements. Since decomposition and integration are widely used strategies for offsetting human limitations in the human modeling process, the power of an MMS will be very limited unless it has these capabilities (Simon, 1981, Liang and Jones, 1986).

The purpose of this paper is to introduce an expert systems approach to building such capabilities in MMSs, with emphasis on the design of reasoning mechanisms that drive the process of model integration and selection. Models are usually knowledge-intensive and composed of many complicated and inter-related functions for data transformation. Developing mechanisms for providing advice and integrating models in MMSs is, therefore, much more difficult than implementing data management functions in data base management systems. This paper takes advantage of

recent progress in artificial intelligence and graph-based models in operations research to develop an architecture for MMSs and mechanisms for model integration and selection.

The remainder of this paper is organized as follows. First, background of MMSs and an architecture for MMS design are briefly described. The architecture consists of three major components: model utilization subsystem, modeling subsystem, and inference engine. Then, a graph-based approach for designing the inference engine is presented. It includes a graph-based representation scheme and mechanisms for model integration and selection. In this approach, a set of data is represented as a node. A set of functions for converting a set of data to another set of data is represented as an edge. Each basic model, a model stored in the model base, is a combination of two nodes and one edge connecting the two nodes. A modeling process is defined as a process that searches a graph capturing all possible alternatives for producing the desired information and then find a path to generate the information. According to this representation scheme, mechanisms for model integration and selection are developed. These mechanisms enable an MMS to automatically combine basic models to provide ad hoc support. Finally, TIMOS (The Integrated MOdeling System), a prototype implemented in PROLOG, is described to illustrate the graph-based approach. Sample consulting sessions are also presented. Successful implementation of the architecture and mechanisms in PROLOG has not only demonstrated the feasibility of building advising and model integration capabilities in MMSs but also indicated a

promising integration of operations research, DSSs and expert systems research.

1. KNOWLEDGE-BASED MODEL MANAGEMENT SYSTEMS

A model is an abstraction of a specific problem or a class of problems. Because of human cognitive limitations, such as limited short-term memory and bounded rationality, people usually use models to help them understand, organize, study, and solve problems (Simon, 1981). Most models designed to support today's human decision-making are complicated, knowledge-intensive, and implemented on computers.

A model base is a collection of those computerized models. In general, a model base is both integrated and shared. By "integrated" we mean that the model base may be thought of as a unification of many otherwise distinct models, with redundancy among those models partially or wholly eliminated. By "shared" we mean that any individual model in the model base may be called by more than one integrated model and may be accessed by any authorized user.

A model management system is a software system that handles all access to the model base and provides information to users on demand. Because of the similarity of the data base and the model base, early research in MMS considered models as data or subroutines and proposed that an MMS must support the following functions (Sprague and Carlson, 1982, Sprague and Watson, 1975, Will, 1975):

1. Creation of new models: providing an environment to support the model builder so that models can be developed with minimum effort;

2. Storage of existing models: maintaining a model base in which decision models are stored;

3. Access and retrieval of existing models: facilitating the utilization of decision models in the model base;

4. Execution of existing models: executing an existing model and reporting outputs of the model; and

5. Maintenance of existing models: supporting the update and modification of the existing models in the model base.

Although these traditional functions may suffice for the need of model management in some systems, such as institutional DSSs which deal with decisions of a recurring nature, they are not sufficient for ad hoc DSSs which are concerned with problems that are not usually anticipated or recurring (Donovan and Madnick, 1977).

For example, a model base contains an economic order quantity (EOQ) model and three demand forecasting models. The EOQ model computes the optimum order quantity for a specific year from the demand, holding cost, and ordering cost for that year. Each of the three demand forecasting models employs the regression, demand function, and moving average approaches to forecast future demand respectively. Suppose one needs to know the economic order quantity for 1987, but one does not have information about the demand for 1987, a required input for executing the EOQ model. There are two ways for the user to produce the desired output in a traditional MMS: first, create a new model that has functions for both demand forecasting and EOQ computation; and second, manually go through the following process:

1. Search the model base and find those demand forecasting models and the EOQ model;

2. Select one among those available demand forecasting models and get the input data required for executing the model;

3. Execute the selected demand forecasting model and then feed the forecasted demand to the EOQ model; and

4. Execute the EOQ model to produce the desired information.

The former approach needs effort to create a redundant model, which is the integration of two existing models; while the latter approach needs effort to integrate models by the user manually. Neither has effectively used existing models. Although this example is straightforward and can be solved easily by the user or any model builder, it does indicate that a powerful MMS needs ad hoc modeling capabilities, which provide advice regarding effective use of existing models in the model base. Namely, it needs the following two capabilities:

1. Model integration

   A mechanism for integrating existing models so that each model in the model base is not only a stand-alone model but also a module for creating ad hoc models which are built in case of need. In this example, the MMS must be able to integrate the demand forecasting models and the EOQ model automatically.

2. Model selection

   A mechanism that figures out what models are available to produce the requested information and then automatically selects or allows the user to select a model for execution. In this case, the MMS must be able to inform the user of all available combinations of the EOQ model and demand forecasting models and allow the user to select one for execution or to execute more than one model and then compare the results.

Since modeling knowledge is essential to the operations of model integration and selection, a model management system with the capabilities of model integration and selection is called a knowledge-based model management system. The major characteristic

that differentiates a knowledge-based MMS from a traditional MMS is its capability of figuring out what, when, and how to integrate models in the model base.

2. ARCHITECTURE FOR A KNOWLEDGE-BASED MMS

Since the process of model integration and selection involves reasoning and judgment, an architecture different from traditional DSS architectures must be used to develop a knowledge-based MMS. In this section, a software architecture adopted from expert systems, as illustrated in Figure 1, will be presented.

INSERT FIGURE 1

In order to support both model creation and model utilization, a Knowledge-based MMS needs two major subsystems: one is the modeling subsystem, and the other is the model utilization subsystem. The modeling subsystem focuses on improving the productivity of model builders; whereas the model utilization subsystem concentrates on effective use of models. In addition, an inference engine is required to drive the processes of model integration and selection and to integrate three basic components: model base, data base, and knowledge base. Basic models are stored in the model base; data pertaining to a decision-making are stored in the data base; and the knowledge regarding effective use of the models in the model base

and the data in the data base is stored in the knowledge base.

2.1 Model Utilization Subsystem

To support effective use of models the model utilization subsystem of an MMS should be able to accept user queries, report results to the user, and provide helpful messages in the course of consultation. In other words, it should have three major functions: query processing, report generation, and help. The query processor is the interface between decision makers and the system. It translates a user's query into a set of commands understood by the system. The report generator provides the requested output in a format the user prefers. The help module provides helpful messages such as how the results were generated.

Since the model utilization subsystem focuses on supporting decision makers, who use the produced information to improve the performance of decision-making, its primary concern is how to generate useful information instead of where the information came from. It would be very useful if the system can provide a unified language that can both retrieve data in the data base and execute models in the model base. The advantages of this language are three-fold.

1. The data base and the model base are integrated and the process for generating the requested information becomes transparent to the user. The user does not need to remember all models stored in the model base in order to use them. Of course, if the user wants to know more about how the information was generated, the help module should be able to provide an explanation,

2. Once a model is developed and stored in the model base, all authorized users will be able to use it without the need of a lengthy process to inform them. This guarentees maximum use of existing models, and

3. Once a model is updated, all users will use the updated version of the model. This reduces possible information inconsistency.

Developing such a natural language processor is an interesting research area (Blanning, 1984). It is, however, not the main concern of this research because it has little effect on the core of the research, development of reasoning mechanisms for model integration and selection. At the current stage, because natural language processing is still more a research area in laboratories rather than a practically applicable technique, a good interactive language may suffice. Once natural language techniques become mature, a pre-processor can be added to the top of the query processor. In later sections, a SEQUEL-like language implemented in the prototype will be described.

2.2 Modeling Subsystem

The modeling subsystem is designed to support the model builder, who is responsible for developing useful models. It should have three major functions: knowledge acquisition, user-assisted modeling, and automatic modeling. The MMS acquires knowledge of models, such as integrity constraints, through the model acquisition module; whereas the model builder interacts with the user-assisted or automatic modeling modules to create new models or modify existing models.

In the user-assisted modeling mode, the model builder edits existing models by performing a set of model manipulation operators, such as decomposition and integration. In the automatic modeling mode, the model builder specifies the information the new model must produce, and the MMS will try to

create the model by editing existing models automatically. In order to support automatic modeling, the inference engine of the system must have a mechanism able to integrate basic models in the model base to create new, more complex models automatically.

2.3 Inference Engine

The inference engine is the heart of a knowledge-based MMS. It performs two major functions: inference and control. The model utilization or modeling subsystem translates a user's request into commands understood by the system, and then activates the inference engine to process the commands, control the access of the data base and the model base, retrieve knowledge from the knowledge base, or make inference if necessary. After obtaining the required information or proving that it is not available, the inference engine passes messages back to those two subsystems and then reports the result to the user.

The inference engine of a knowledge-based MMS needs three major inference mechanisms:

1. Integrating the data base and the model base,
2. Integrating models in the model base, and
3. Controlling the execution of the selected model.

The first mechanism controls the access of the data base and the model base. For any query, the MMS first searches the data base. If the information is available in the data base, it will be retrieved. Otherwise, the system will search the model base and see whether there is a set of models available for producing the information. If there is any model available, the mechanism will check the availability of its inputs and then retrieve the

inputs to execute the model (Liang, 1985).

If no basic model in the model base is available to produce the required information, the second mechanism will take over and try to develop executable integrated models. A detailed discussion of a graph-based mechanism will be presented in the next section.

After the model for producing the desired information is chosen, the mechanism for model execution, which schedules the integrated models and makes sure that they are executed in a proper sequence, will be activated.

3. A GRAPH-BASED INFERENCE MECHANISM

The discussion in the previous section has indicated that the mechanism for integrating models is at the heart of a knowledge-based MMS. Although research in MMSs has increased dramatically in the past decade, most of it was concentrated on either the application of existing data models, such as the relational model (Blanning 1982) and the network model (Stohr and Tanniru, 1980), or knowledge representation schemes. Few mechanisms that provide advice about model integration and selection have been developed.

Recently, Geoffrion proposed an approach called structured modeling, which focused on exploring functional relationships among the modules constituting a model during the modeling process (Geoffrion, 1985). Although this approach may have significant impact on the development of MMSs, it was not specifically developed for model management. Nor did the approach provide mechanisms for model integration or model

selection if more than one model is available to support a specific decision.

In this section, a graph-based framework for building the capabilities of model integration and selection is presented. The framework covers the following three essential issues:

1. Representation of models,
2. Algorithm for integrating models, and
3. Algorithm for selecting models.

3.1 Graphical Representation of Models

In order to build model integration and selection capabilities, the first issue to be considered is how to represent models in the model base. Unless a model has been properly represented, an MMS will not be able to access it. This section introduces a graph-based approach to representing models in the model base. The advantages of this approach are as follows:

1. It is easy to understand;
2. It is compatible with the model integration and selection mechanisms to be discussed in the next section; and
3. Given the graphical representation, many heuristics and algorithms developed in graph theory can be applied to model management.

As problem solving is often described as a search through a vast maze of possibilities (Simon, 1981), so can the process of human modeling be described as a search through a number of possible relationships in order to find a route which can convert the initial state (available information) of a problem to the desired final state (output information). By this concept, models in the model base can be represented by two basic

elements: nodes and edges. The modeling process can be formulated as a process that creates a directed graph and selects a path on the graph. The directed graph, called a model graph, represents all possible alternatives for solving the problem; and each path in the graph represents a model. They can be defined as follows.

Definition 1: Node

A node, N, represents a set of data attributes. It could be the inputs or the outputs of a set of models.

[Example] In Figure 2a, node A represents a set of data including the demand, holding cost, and ordering cost. Node B represents the computed economic order quantity.

INSERT FIGURE 2

Definition 2: Edge

An edge, e, represents a set of functions that convert a set of input data (the starting node of the edge) to their associated output (the ending node).

[Example] The edge e1 in Figure 2a represents the function which computes EOQ from the demand, holding cost, and ordering cost.

Definition 3: Connectivity

Two nodes are connected if there exists at least one edge that converts the data in one node to that in another.

[Example] Node A and B in Figure 2a are connected because edge e1 converts the demand, holding cost, and ordering cost in node A to the EOQ in node B.

In practical applications, both nodes and edges should be nonempty sets. A combination of two connected nodes and one edge connecting the two nodes constitutes a basic model, the smallest

unit in the model base.

Definition 4: Basic model

A basic model, Mb, is a combination of two nodes and an edge connecting the two nodes. The starting node of the edge represents the inputs of the basic model, and the ending node of the edge represents the outputs of the basic model. Hence, a basic model can be represented as a triple, <N1,e,N2>.

[Example] The combination of <A,e1,B> in Figure 2a is a basic model.

Each basic model in the model base is a stand-alone model, but it is also a basic element for automatic modeling. Since there is usually more than one way to convert a set of inputs to a set of outputs, the edge between two nodes may not be unique. That is, a model base may have more than one model for solving a particular problem. For example, if one wants to forecast demand for the next year based on the demand data in the last 15 years, one may use the moving average, exponential smoothing, regression, or the Box-Jenkins approach, as illustrated in Figure 2b. In other words, four basic demand forecasting models in the model base, <C,a,D>, <C,b,D>, <C,c,D>, and <C,d,D>, are available for forecasting the future demand.

In addition to the case where more than one model is available to produce a set of required outputs, it is possible that a set of basic models, in combination, produce the required outputs, but each individual model produces only a subset of the required outputs. In order to differentiate these two situations, we need to define two types of nodes: AND nodes and OR nodes.

Definition 5: AND node

An AND node, Na, is a node that is the ending node of more than one basic model. Each model produces a subset of the required output, but the combination of these models produces the whole set of the required outputs. An AND node is true only if all edges ending at the node are true.

[Example] Node D in Figure 2c is an AND node because the model <A,a,D> produces the demand information, the model <B,b,D> produces the holding cost, and the model <C,c,D> produces the ordering cost. Therefore, the three models, in combination, produce the information contained in node D, but each model produces only a subset of the information. In this paper, an AND node is represented as a circle.

Definition 6: OR node

An OR node is a node that is the ending node of more than one basic model; each model produces the entire set of required information. An OR node is true if at least one edge ending at the node is true. In this paper, an OR node is represented as a square.

[Example] Node D in Figure 2b is an OR node because there are four models ending at node D and each of which can produce the forecasted demand.

In the human modeling process, an AND node represents a union point where more than one set of output data is combined to formulate the required output; and an OR node represents a decision point where one or more models are selected among those available models.

Because all of the four forecasting models represented in Figure 2b produce the same set of outputs, and no output of a model becomes an input of another model in the graph, it can be called a one-stage graph. However, not all modeling problems are as simple as this example. Rather, many problems may need integration of various kinds of models. By "integration", we mean that two or more models are combined to become an integrated model in which the output of a model is fed into another model. For example, Figure 2d illustrates a two-stage graph. It

represents an integration of the EOQ and demand forecasting models described in the previous section. In the figure, the output of the demand forecasting models (A1, A2, or A3) are fed into the EOQ model (B1). Because the model base has one model for EOQ computation and three models for demand forecasting, there are three paths (1*3), i.e., three different models, for producing the desired information. Formal definitions of the path, integrability and integrated model are as follows.

Definition 7: Path

A Path, P, is a finite sequence of edges of the form that

(1) these edges are connected,
(2) at each OR node, only one edge that enters the node is true,
(3) at each AND node, all edges that enters the node are true.

Definition 8: Integrability

Two basic models are integrable if the input of one model and the output of the other share common data attributes.

Definition 9: Integrated model

An integrated model, Mi, is a model which integrates a set of basic models.

According to the definitions previously described, the concept of a model graph and the modeling process can be defined.

Definition 10: Model graph

A model graph, G, is a graph which represents all possible models, including basic models and integrated models, for producing the requested information. Each path in a model graph represents a model. A model graph must be acyclic.

[Example] Figure 2d is a model graph which represents models for computing EOQ. The model graph is composed of three integrated models. For example, path A1-B1 is the model which forecasts demand by using the moving average technique (edge A1) and then computes the EOQ by using the EOQ model (edge B1).

Definition 11: Modeling process

A modeling process is a process that includes two phases: the formulation of a model graph and the selection of one or more paths in the formulated model graph.

The modeling process is a logical process that formulates a model graph capturing all possible paths for producing the requested outputs and makes selection in the graph. Because a model graph clearly represents the relationships among relevant basic models, it becomes much easier for the system to provide advice regarding model integration and selection. Based on the model graph, an MMS may either perform model integration and selection automatically (the automatic modeling mode) or provide advice about model integration to the user and then allow the user to create integrated models (the user-assisted modeling mode). For implementation purpose, each model graph must be acyclic. Otherwise, the modeling process may become an infinite process.

Each path in the graph implies an appropriate model; but it does not guarantee that the model will generate a feasible solution. For example, if a model base contains a capital budgeting model which uses the integer programming technique to determine the best combination of projects for investment, the model graph only indicates the existence of this model, but it will not be able to tell the user whether the model can produce a feasible solution until the model is actually executed.

After formulating a model graph and choosing a path in the graph, the MMS also needs a process for executing the selected path. In graphical terms, the model execution process can be defined as follows:

Definition 12: Execution process

The model execution process is a process that activates a path and then executes the models constituting the path in an appropriate sequence in order to generate the output.

3.2 Implementation of the Graph-based Representation

Concerning the implementation of a model representation method, a model can be portrayed by the following five categories of information:

- The output of the model,
- The input required to produce the output,
- The computational procedures used in the model,
- The integrity constraint of the model, and
- The validity of the model.

In other words, a basic model can be represented by a set of five relations: relations between the model and its inputs, outputs, integrity constraints, validity evaluation, and computational subroutines, as follows:

```
INPUT (Modelname, Inputs)
OUTPUT (Modelname, Outputs)
OPERATION (Modelname, Functions)
INTEGRITY (Modelname, Constraints)
VALIDITY (Modelname, Evaluation)
```

Each relation in the scheme represents a unique characteristic of a model. They should be read as "the inputs of <modelname> include <input1, input2,..>", "the outputs of <modelname> include <output1, output2,..>", and so forth. The first four relations are important to the formulation of a model graph, and the fifth relation (validity relation) is important to the selection of models.

The advantages of this scheme are two-fold. First, it is

non-procedural, i.e., the model builder specifies what the model is rather than how the model computes data. Second, it can be implemented easily in a symbolic language, such as PROLOG.

Corresponding to the graph-based representation, the input and output relations are nodes for formulating a model graph. The operation relation is represented as an edge. It specifies computational functions used in a model and is part of the interface between the logical integration of models indicated in a model graph and the actual execution of the selected model. A basic model, identified by a unique name, is a combination of one operation relation (an edge) and its associated input relation and output relation (two nodes).

The integrity relation of a model specifies constraints that must be satisfied before the model can be considered applicable to a specific problem. For example, the least squares linear regression technique requires that the number of cases must be larger than the number of independent variables plus two. Unless this constraint is satisfied, the sales forecasting model using the regression approach should not be considered in formulating the model graph.

The validity relation indicates a measure of the fitness of a model to a particular problem. Because the validity of a model can only be assessed after it has been implemented, the validity value in the relation usually represents the historical validity of the model in a specific context. In other words, it represents a kind of subjective confidence in the model, based on a pre-defined model evaluation function or previous experience in that specific context. For example, in the case of forecasting

future sales, our experience indicates that the accuracy of the moving average technique is poor for identifying the turning point in a trend (Chambers, Mullick, and Smith, 1971), the model should have a low validity value when it is considered for forecasting the turning point.

Measuring validity is important to the automation of model selection. If more than one model is applicable to a specific problem, the MMS needs their validity values to make selection automatically. For implementation considerations, we need a quantitative measure of validity. A model evaluation function that determines the validity of a model based on a set of pre-determined criteria is required. The reason for quantifying validity is that the validity value can be manipulated and calculated to facilitate model selection in a model graph. For example, different modeling strategies, including optimizing and satisficing, can be implemented in the automatic modeling process. The optimizing algorithm selects the best model among all alternatives for the user based on the expected validities of available models; whereas the satisficing strategy selects the first model whose validity is higher than the satisfactory level.

Because of space limitation, it is difficult to provide a complete discussion on the development of model evaluation functions in this paper. In general, the following three issues should be considered:

1. What are proper criteria for determining validity value? There are at least five possible criteria: (1) accuracy of the model, (2) the user's preference for the model, (3) distance from producing the desired information, (4) number of models integrated, and (5) total cost.

2. How can several validity values be combined to get the overall evaluation of an integrated model?

3. When should a model be evaluated?

Further discussions about these issues can be found in (Liang and Jones, 1986).

Figure 3 is a sample representation of the EOQ model. The model has an integrity constraint that both the holding cost and ordering cost must be constants in the period. If the integrity constraints are satisfied, the validity of the model is 0.8 on a 0.0 to 1.0 scale.

INSERT FIGURE 3

3.3 Mechanism for Formulating a Model Graph

The major motivation for developing the graph-based model representation scheme is to build consulting capabilities in MMSs, which can provide advice concerning model integration and selection. To develop such capabilities we need a mechanism to formulate model graphs, the basis on which advice is generated.

Formulation of a model graph involves extensive search in the data base and the model base. Many heuristics have been developed for creating and traversing a search tree (see Busacker and Saaty, 1965, Carre, 1979, Rich, 1983 or other books in graph theory or artificial intelligence for a review). These include depth-first search, breadth-first search, and best-first search. For creating a model graph the depth-first search and the best-first search strategies are better than the breadth-first search

strategy because they support both the optimizing and the satisficing modeling strategy. An MMS generates a satisfactory model in the satisficing strategy but selects the model with the highest validity in the optimizing strategy. In this section, an algorithm for formulating the model graph will be presented. This algorithm is based on the depth-first search strategy and compatible with the graph-based model representation scheme. The application of different modeling strategies will be discussed in the next section.

The basic idea of the depth-first search is to pick up an alternative at every node arbitrarily and work forward from that alternative. Other alternatives at the same level are completely ignored as long as there is any hope of reaching the destination using the orginal choice. If the original choice is proved impossible to lead to a solution, then go back one level to work on another alternative.

Suppose a user has placed a query and the requested information is not directly available in the database, procedures for applying the depth-first search to formulate a model graph are as follows:

Step 1: Search OUTPUT relation in the model base to see whether there is a model that produces the output.

Step 2: If no model is found, then stop searching and report that no model is available in the model base. The system may ask the user to develop a new model.

Step 3: If a model is available, then search INPUT relation of the model to find the input data required for execution.

Step 4: Repeat the following process until all inputs are obtained or one input is proved unavailable:

4.1: Pick up an input, check whether it is an output of its

preceding models (check for acyclicity).

4.1.1: If it is true, then drop this model and go to step 3.

4.1.2: IF it is not true or the model does not have any preceding model then skip this procedure.

4.2: Search the data base for availability.

4.2.1: If the input is available in the data base, then retrieve its value and go to step 4.1.

4.2.2: If the input is not available in the data base, then go to step 4.3.

4.3: Search OUTPUT relation of the model to see whether it can be produced by a model in the model base.

4.3.1: If no model is available, then go to step 4.4.

4.3.2: If a model is found, then go to step 4.5.

4.4: Prompt the user for the input.

4.4.1: If it is provided by the user, then obtain its value and go to step 4.1.

4.4.2: Otherwise, drop the model.

4.5: Search the INPUT relation of the model to find input data required for execution. Repeat step 4 until all input data have been obtained or one input is proved unavailable.

Step 5: If all input data are available, then check integrity constraints.

5.1: If any integrity constraint is not satisfied, then drop the model.

5.2: If all constraints are satisfied, then add the model to the model graph.

Step 6: Check whether there is another model for producing the desired information.

6.1: If there is another model, then go to step 3.

6.2: Otherwise, stop the process and then provide advice based on the formulated model graph.

Figure 4 illustrates the process for formulating a model

graph. The circled numbers in the figure are corresponding steps. A prove of the generality of this mechanism is presented in Appendix 1.

The procedures of the best-first search are basically the same as that of the depth-first search, except that the former employs an evaluation function to evaluate the potential of all possible paths before further investigation and gives higher priority to better paths in order to make sure that models with higher validities will be examined earlier. There are certainly other possible approaches for building model graphs. They will not be discussed here, however, because they may be derived from the procedures described before.

INSERT FIGURE 4

In this mechanism, if the operation that picks up an input of a model and searches for the availability of the specific input is considered a basic operation in the model base and represented as an edge, then the formulated model graph will be an alternate AND/OR tree.

Definition 13: Tree

A tree, T, is a graph containing one or more nodes such that

(1) there is a specially designed node called root,
(2) the remaining nodes are partitioned into n (n>=0) disjoint sets T1,...,Tn where each of these set is also a tree. T1,...,Tn are called the subtrees of the root.

Definition 14: AND/OR tree

An AND/OR tree is a tree that includes both AND nodes and OR nodes.

Definition 15: Alternate AND/OR tree

An alternate AND/OR tree is a tree in which the AND node and OR node appear at alternate levels. In other word, if nodes at level m are AND nodes, then the nodes at level m+1 must be OR nodes.

[Example] Figure 5 illustrates an alternate AND/OR graph. It is the model graph formulated by the above algorithm for providing advice about the EOQ and demand forecasting problem described in the first section.

INSERT FIGURE 5

Proposition 1: The model graph formulated in the above algorithm is an alternate AND/OR tree.

A prove of this proposition is in Appendix 2.

3.4 Strategies for Model Selection

Given the formulated model graph for producing the desired information, there are two different strategies for providing advice: optimizing and satisficing. The optimizing strategy requires that an MMS formulate a model graph and then evaluate all paths in the graph to find the best alternative. The satisficing strategy, on the other hand, requires that each path be evaluated immediately after it is found and accepted if it is satisfactory. Therefore, a complete model graph may not be necessary in the satisficing strategy.

If validities of all models in the model graph are available, then the optimizing strategy is simply to maximize the validity of the selected path. This can be formulated as a maximum validity flow problem subject to the constraints of

modeling time, modeling costs, and other considerations. In this case, most algorithms for finding the best path in graph theory can be applied to solve the problem.

Although the optimizing strategy guarantees that, given the criteria, the formulated model is the best available, the combinatorial explosion sometimes makes it unrealistic and forces a system to adopt the satificing strategy. In the satisficing strategy, the MMS follows the same procedure to formulate a model graph except that every path is evaluated at the time it is formulated. If a satisfactory path has been found, the process for formulating the model graph will be terminated and the path will be chosen to produce the desired information. Figure 6 briefly illustrates the modeling process for implementing the satisficing strategy.

INSERT FIGURE 6

No matter whether the satisficing or optimizing strategy is chosen, heuristics must be used to reduce the complexity of the process. The following is a sample heuristic for model selection:

Step 1: Determine the validity of each edge (a model) in the model graph.

The system retrieves the validity of each selected member model by searching the VALIDITY relation of the model or executing the evaluation function if appropriate.

Step 2: Simplify the problem by removing dominated alternatives.

If more than one edge is connecting two nodes, i.e. more than one model is available to convert a set of inputs to its associated outputs, then select the one with the highest validity and ignore the rest.

Step 3: Calculate validities for all possible paths from the initial state to the final state.

The validity of a path equals the product of the validities of its member edges.

Step 4: Select the path with the highest validity.

The selection may be constrained by some other non-technical constraints, such as the computational cost, modeling time and so forth. Therefore, it may also need an integer program to determine which path is the best one. However, because of the screening procedures described in steps 1-3, the new formulation should be more efficient than the original one.

4. IMPLEMENTATION OF THE FRAMEWORK

The graph-based framework described in the previous section has provided a sound basis for developing model integration and selection capabilities of a knowledge-based MMS. In this section a prototpye implemented in PROLOG, called TIMOS (The Integrated MOdeling System), is presented to demonstrate the feasibility of the framework.

TIMOS is an integrated system with the model integration capability. By "integrated system" we mean it supports the following three functions:

1. Retrieval of data in the data base.
2. Retrieval and execution of models in the model base.
3. Formulation and execution of an integrated *ad hoc* models.

4.1 Architecture of TIMOS

The architecture of the system is quite similar to that

illustrated in Figure 1. In other words, TIMOS has two major subsystems: one is the model utilization subsystem, and the other is the modeling subsystem. A graph-based inference engine drives the integration among models and between the model base and the data base.

4.2 Model Utilization Subsystem

The model utilization subsystem has three major functions: query processing, report generation, and help. TIMOS provides an interactive query language, TQL (The Query Language), in which a user can access both the data base and the model base without the need to identify where the information is stored beforehand.

TQL is a SEQUEL-like language. The system maintains a data dictionary to facilitate understanding of terms used by users. In using TQL, a user first specifies the category of the required output, such as "sales". Then, the system will retrieve the associated attributes from the data dictionary (for "sales" these might be "product" and "year"), prompt for their values and process the query. For example, the query for getting the information, "sales for toy for 1987", is as follows:

```
SELECT: sales
WHERE
   PRODUCT = toy
   YEAR = 1987
```

In addition to TQL, TIMOS has a report generator for producing formatted information, and a help module to provide helpful information if requested to do so.

4.3 Modeling Subsystem and The Inference Engine

TIMOS employs the graph-based framework as its inference engine. Since it is implemented in PROLOG, a logic-based programming language, the theoretical foundation for inference is predicate logic. The graph-based inference mechanism is on top of predicate logic.

In order to implement the alternate AND/OR tree, TIMOS uses a recursive structure named "path". A technical description of the path structure is in Appendix 3. By implementing the path structure, the system supports the formulation of model graphs and a simple satisficing strategy that the system will provide advice based on the first alternative available. If the user does not like the first piece of advice and asks for more, the system will then provide the next alternative, if available, to the user. This process can go on until no alternative is available.

INSERT FIGURE 7

Figure 7 is a sample session of consultation for integrating the EOQ and demand forecasting models to produce the EOQ for product "a" for 1987. The user specifies the desired information, the system first searches the database and reports that it is not available in the database. Then, the system searches the model base, formulates a model graph as previously illustrated in Figure 5, and then informs the user that the

integration of the EOQ model and a demand forecasting model will be able to generate the desired information. The user may accept that advice and execute the integrated model, as shown in the session, or request more advice. The path structure for this query is illustrated in Appendix 3.

5. CONCLUDING REMARKS

One of the most important but difficult research issues in developing model management systems is how to develop model integration and selection capabilities. This paper has presented a graph-based approach to building such capabilities in MMSs. With these capabilities a knowledge-based MMS can integrate basic models in the model base to formulate ad hoc decision models. The models in the model base are not only considered as basic models for decision support but also treated as building blocks for creating complex models.

In order to build such capabilities, a software architecture adopted from expert systems and a graph-based inference framework were developed. The architecture consists of three major components: model utilization subsystem, modeling subsystem, and inference engine. The inference engine is the core of the system. It adopts the graph-based framework to drive model integration and selection processes. In the framework, a set of data is defined as a node, a set of functions for converting data is defined as an edge, and a basic model is represented as a combination of two nodes and one edge connecting the two nodes. Based on this graphical representation scheme, mechanisms for model integration and selection have been presented. Successful

implementation of the architecture and the graph-based framework in PROLOG has indicated a promising integration of operations research, expert systems and DSSs research.

Appendix 1: Prove of the Model Integration Mechanism

To be useful the mechanism must fulfill two requirements:

1. Completeness: it must be able to build a graph that capture all models for producing the desired information; and
2. Termination: the process must stop after finding all candidates.

1. Completeness

(i) Assume a model base, MB, has one model (M(1) = <I1,P1,O1>) for producing the desired output Df, and the mechanism cannot find it. If we use C1 to represent the integrity constraints of M(1), MB to represent all models in the model base, DB to represent the data in the data base and UR to represent the data provided by the user, then the following statements are true:

(1) $M(1) \in MB$,
(2) $Df \subset O1$,
(3) $I1 \subset DB \cup UR$,
(4) C1 is satisfied, and
(5) M(1) is not in the model graph.

By examining the mechanism, statement (5) is true if and only if one of the following conditions is true:

(6) $M(1) \notin MB$, (step 2)
(7) $Df \not\subset O1$, (steps 2 & 4.3)
(8) $I1 \not\subset DB \cup UR$, (step 4.2 & 4.4)
(9) C1 is not satisfied. (step 5)

These conflict with our original assumptions. Therefore, if there is one model in the model base and it fulfills conditions (1) - (4), then the mechanism must be able to add it to the model graph.

(ii) Assume a model base has m models (M(j) = <Ij,Pj,Oj> where j = 1..m) for producing the desired output Df, and the mechanism can build a model graph capturing all m models. However, when one more model for producing the desired output is

added into the model base, the mechanism will not be able to find m+1 models. There are two situations under which the mechanism will not be able to find m+1 models:

1. The mechanism fails to add one of the models, M(1),..,M(m), to the model graph; and
2. The mechanism fails to add model M(m+1) to the model graph.

Since we assume the mechanism can build a model graph capturing all m models, the first situation is not true. In addition, we have proved in (i) that if there is one model for producing the desired information Df, then the mechanism will be able to add it to the model graph. The second condition is not true either. Therefore, we can conclude that given the condition that the mechanism can build a model graph capturing m models, it will be able to build a model graph including m+1 models if the model base has m+1 models for producing the desired output.

Based on mathematical induction, we can conclude that the mechanism can build a model graph that captures all models for producing the desired output in the model base.

2. Termination

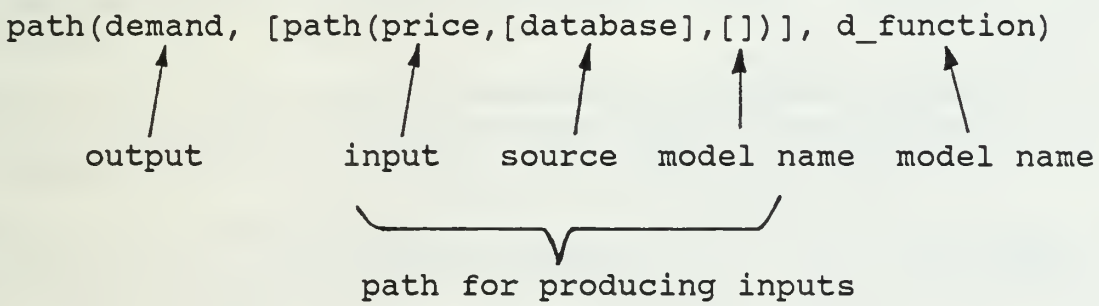
Assume we have a model base containing finite number of models and the mechanism will not stop in the course of formulating a model graph. Since the number of models in the model base is finite, there is only one situation under which the assumtion is true: a model needs the output of its preceding models as input. In this case, a cyclic graph is formulated. Since the mechanism includes a procedure to detect cyclicity (step 4.1), a cyclic path will be dropped as soon as it is detected. Therefore, we can conclude that the mechanism is a finite process.

Appendix 2: Prove of Proposition 1.

The algorithm employs two kinds of operations: one is picking up an input, the other is finding possible models that produce the input. If the former operation is performed on a node, then the node becomes true only if the operation has been successfully applied to all inputs (i.e., this node is an AND node). If the latter operation is performed, then the node becomes true if any model in the model base is available (i.e., this node is an OR node). Since these two kinds of operations are applied alternately in the propagation process of the model graph, the formulated graph must be an alternate AND/OR tree.

Appendix 3: Path Structure

"Path" is a list composed of three elements: the desired output, name of the model for producing the output, and the paths for providing the required inputs of the model (these paths are again represented in the "path" structure; this situation is called recursive). For example, a model, that forecasts future demand (its output) by the price of the product (its input), can be represented as follows:



In the example, the "d_function" is the name of the model. The path in the square bracket represents a well-formed list. A list is a common data structure which has two elements: head and tail. A well-formed list is a special kind of list whose tail is a list; the tail of the tail is, in turn, also a list, and so on. The well-formed list, "[database]", in the input path means that the price is retrieved from the data base; whereas the "[]" (nothing in the square bracket) means that no model execution is required since no model is included in the input path. The path structure illustrated in the following represents the model graph shown in Figure 5.

```
[path(eoq(a,1987,X),
     [path(h_cost(a,5),[database],[]),
      path(o_cost(a,20),[database],[]),
      path(demand(a,1987,Y),
          [path(demand(a,1986,158),[database],[]),
           path(demand(a,1985,145),[database],[]),
           path(demand(a,1984,132),[database],[])],
      M4)],
 M1),                        /* First Path */
 path(eoq(a,1987,X),
     [path(h_cost(a,5),[database],[]),
      path(o_cost(a,20),[database],[]),
      path(demand(demand(a,1987,Y),
          [path(demand(a,1986,158),[database],[]),
           path(demand(a,1985,145),[database],[]),
           path(demand(a,1984,132),[database],[]),
           path(demand(a,1983,123),[database],[])],
      M3],
 M1),                       /* Second Path */
 path(eoq(a,1985,X),
     [path(h_cost(a,5),[database],[]),
      path(o_cost(a,20),[database],[]),
      path(demand(a,1987,Y),
          [path(price(a,10),[user],[])],
      M2)],
 M1)].                      /* Third Path */
```

This structure includes three paths that produce EOQ for product "a" for 1987: integrating models M4 and M1, integrating models M3 and M1, and integrating models M2 and M1. In the path structure, "[user]" means that the information is provided by the user.

REFERENCES

Alter, S. 1980. Decision Support Systems: Current Practices and Continuing Challenges. Addison-Wesley, Reading, MA.

Blanning, R. W. 1982. A Relational Framework for Model Management in Decision Support Systems. DSS-82 Transactions. 16-28.

Blanning, R. W. 1983. Issues in the Design of Relational Model Management Systems. AFIPS Conference Proceedings. 395-401.

Blanning, R. W. 1984. Conversing With Management Information Systems in Natural Language. Communications of the ACM. 27:3, 201-207.

Blanning, R. W. 1985. A Relational Framework for Join Implementation in Model Management Systems. Decision Support Systems. 1:1, 69-82.

Bonczek, R. H., Holsapple, C. W., and Whinston, A. B. 1980. The Evolving Roles of Models in the Decision Support Systems. Decision Sciences. 11, 337-356.

Bonczek, R. H., Holsapple, C. W., and Whinston, A. B. 1981. The Evolution from MIS to DSS: Extension of Data Management to Model Management. in M. J. Gintzberg and E. A. Stohr(eds.), Decision Support Systems. North-Holland, amsterdam.

Bonczek, R. H., Holsapple, C. W., and Whinston, A. B. 1981. Foundations of Decision Support Systems. Academic Press, New York.

Busacker, R. G. and Saaty, T. L. 1965. Finite Graphs and Networks: An Introduction with Applications. McGraw-Hill, New York.

Carre, B. 1979. Graphs and Networks. Oxford University Press, Oxford, UK.

Chambers, J. S., Mullick, S. K., and Smith, D. D. 1971. How to Choose the Right Forecasting Techniques. Harvard Business Review. 49:4, 55-64.

Codd, E. F. 1970. A Relational Model of Data for Large Shared Data Banks. Communications of the ACM. 13:6, 377-387.

Dolk, D. R. 1982. The Use of Abstractions in Model Management. Ph.D. Dissertation, University of Arizona.

Dolk, D. R. and Konsynski, B. R. 1984. Knowledge Representation for Model Management Systems. IEEE Transactions on Software Engineering. SE-10:6, 619-628.

Donovan, J. J. 1976. Database System Approach to Management

Decision Support. ACM Transactions on Database System. 1:4, 344-369.

Donovan, J. J. and Madnick, S. E. 1977. Institutional and Ad Hoc DSS and Their Effective Use. Data Base. 8:3, pp.79-88.

Elam, J. J. 1980. Model Management Systems: A Framework for Development. Proceedings of the 1980 SEAIDS. 35-38.

Elam, J. J., Henderson, J. C., and Miller, L. W. 1980. Model Management Systems: An Approach to Decision Support in Complex Organization. Proceedings of the First International Conference on Information Systems. 98-110.

Geoffrion, A.M. 1985. Structured Modeling. Monograph. Graduate School of Management, University of California, Los Angeles.

Jones, C. V. 1984. Graph-based Models. Ph.D. Dissertation, Cornell University.

Keen, P. G. W. and Scott Morton, M. S. 1978. Decision Support Systems: An Organizational Perspective. Addison-Wesley, Reading, MA.

Konsynski, B. R. and Dolk, D. R. 1982. Knowledge Abstractions in Model Management. DSS-82 Transactions. 187-202.

Liang, T. P. 1985. Integrating Model Management With Data Management in Decision Support Systems. Decision Support Systems. 1:3, 221-232.

Liang, T. P. and Jones C. V. 1986. Meta-design Considerations in Developing Model Management Systems. Decision Sciences Working Paper. The Wharton School, University of Pennsylvania.

Little, J. D. C. 1970. Models and Managers: the Concept of a Decision Calculus. Management Science. 16, B466-B485.

Rich, E. 1983. Artificial Intelligence. McGraw-Hill, New York.

Simon, H. A. 1981. The Science of the Artificial. MIT Press. Cambridge, MA.

Sprague, R. H. and Carlson, E. D. 1982. Building Effective Decision Support Systems. Prentice-Hall, Englewood Cliffs, NJ.

Sprague, R. H. and Watson, H. J. 1975. Model Management in MIS. Proceedings of the 7th National AIDS Meeting. 213-215.

Stohr, E. A. and Tanniru, M. R. 1980. A Database for Operations Research Models. International Journal of Policy Analysis and Information System. 4:1, 105-121.

Watson, G. W. 1983. Knowledge Base Management for Model Management. Master Thesis, Naval Post Graduate School.

Will, H. J. 1975. Model Management Systems. in E. Grochla and Szyperski (eds.) Information Systems and Organization Structure. Walter de Gruyter, Berlin, 467-482.

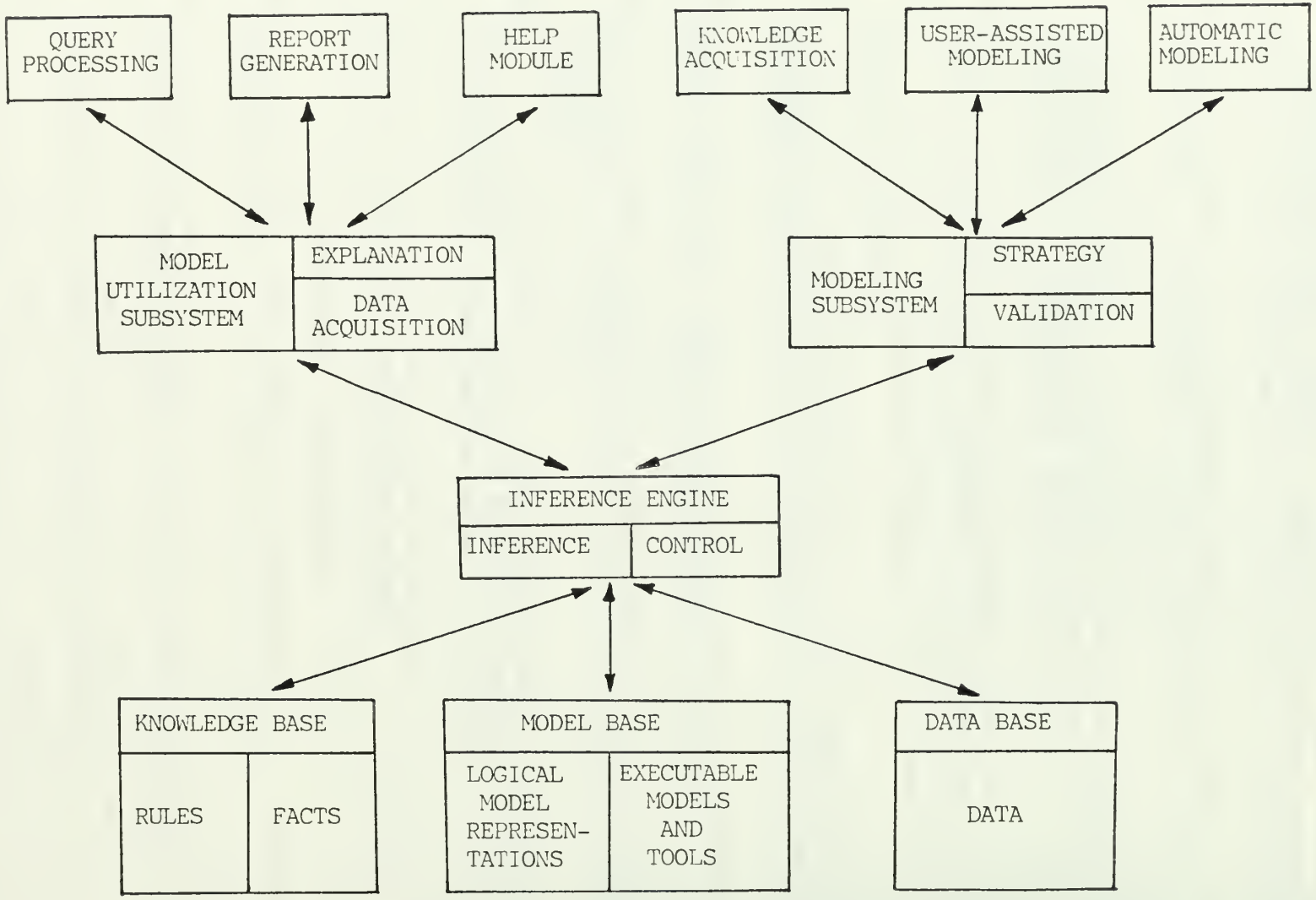
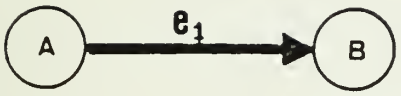


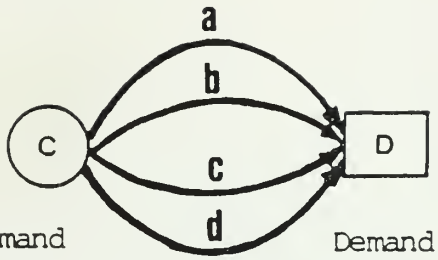
Figure 1. A Software Architecture for MMS



Demand,
Holding Cost,
Ordering Cost

EOQ

(a) Graphical Representation of the EOQ Model

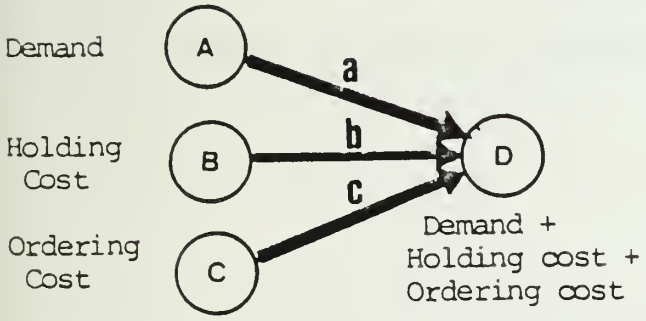


Demand (N-15,..,N)

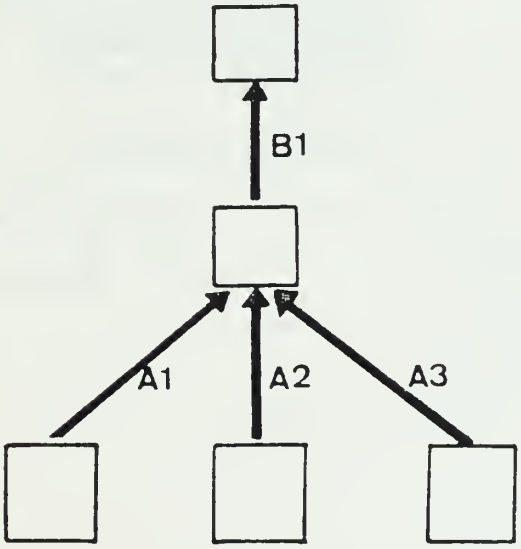
Demand (N+1)

where: a: Moving average
b: Exponential Smoothing
c: Regression
d: Box-Jenkins

(b) A one-stage Modeling Process



(c) An Example of AND nodes



Where: A1: Moving average
A2: Regression
A3: Demand function
B1: EOQ

(d) A Two-stage Modeling Process

Figure 2. Graph-based Representations

```
OUTPUT(EOQModel, [Economic order quantity])
INPUT(EOQModel, [Demand, Ordering cost, Holding cost])
OPERATION(EOQModel, [EOQ subroutine])
INTEGRITY(EOQModel, [Constant(Ordering cost, Holding
                                cost)])
VALIDITY(EOQModel, [0.8])
```

Figure 3. Representation of the EOQ model

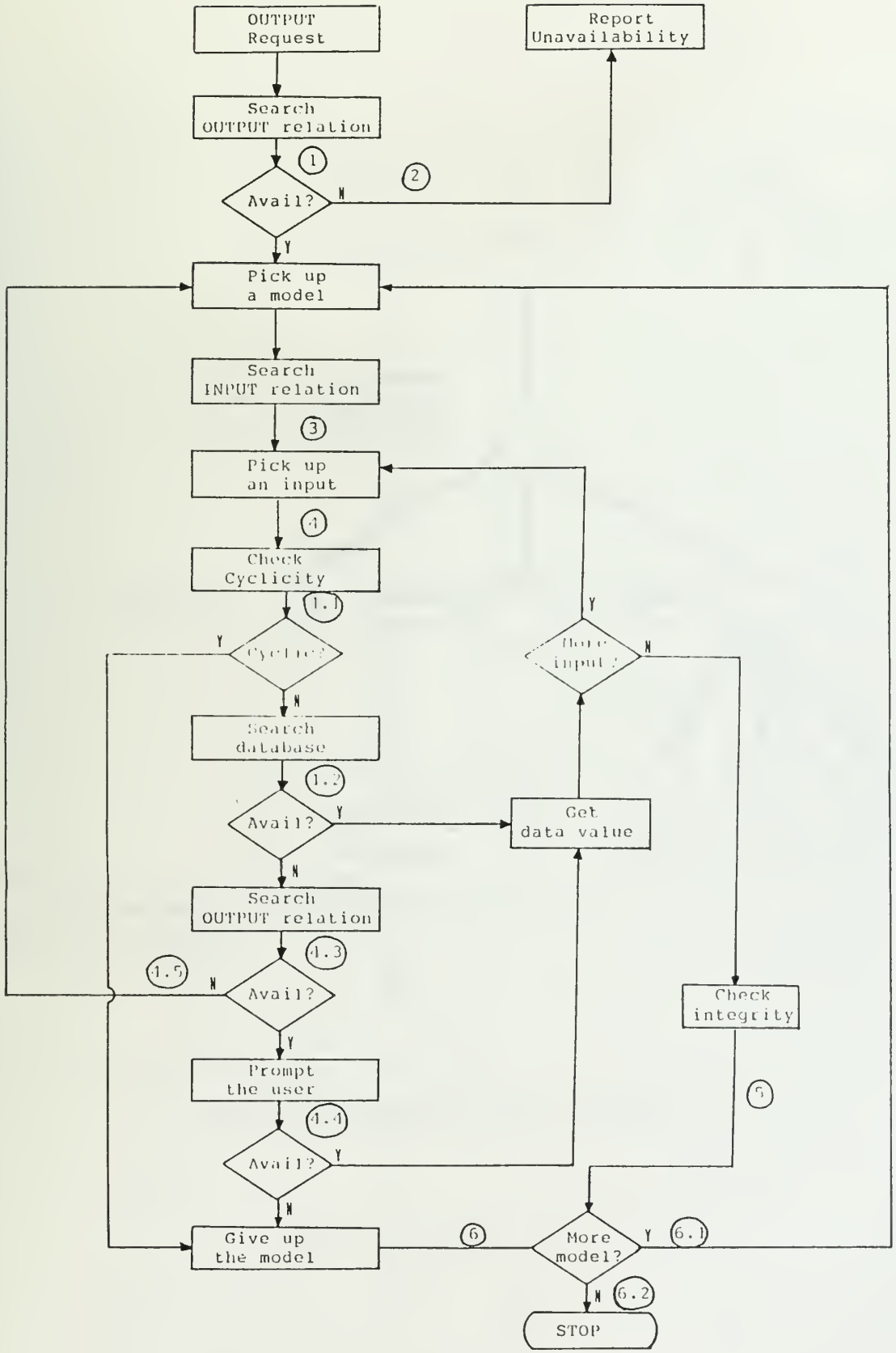


Figure 4. Process for Formulating a Model Graph

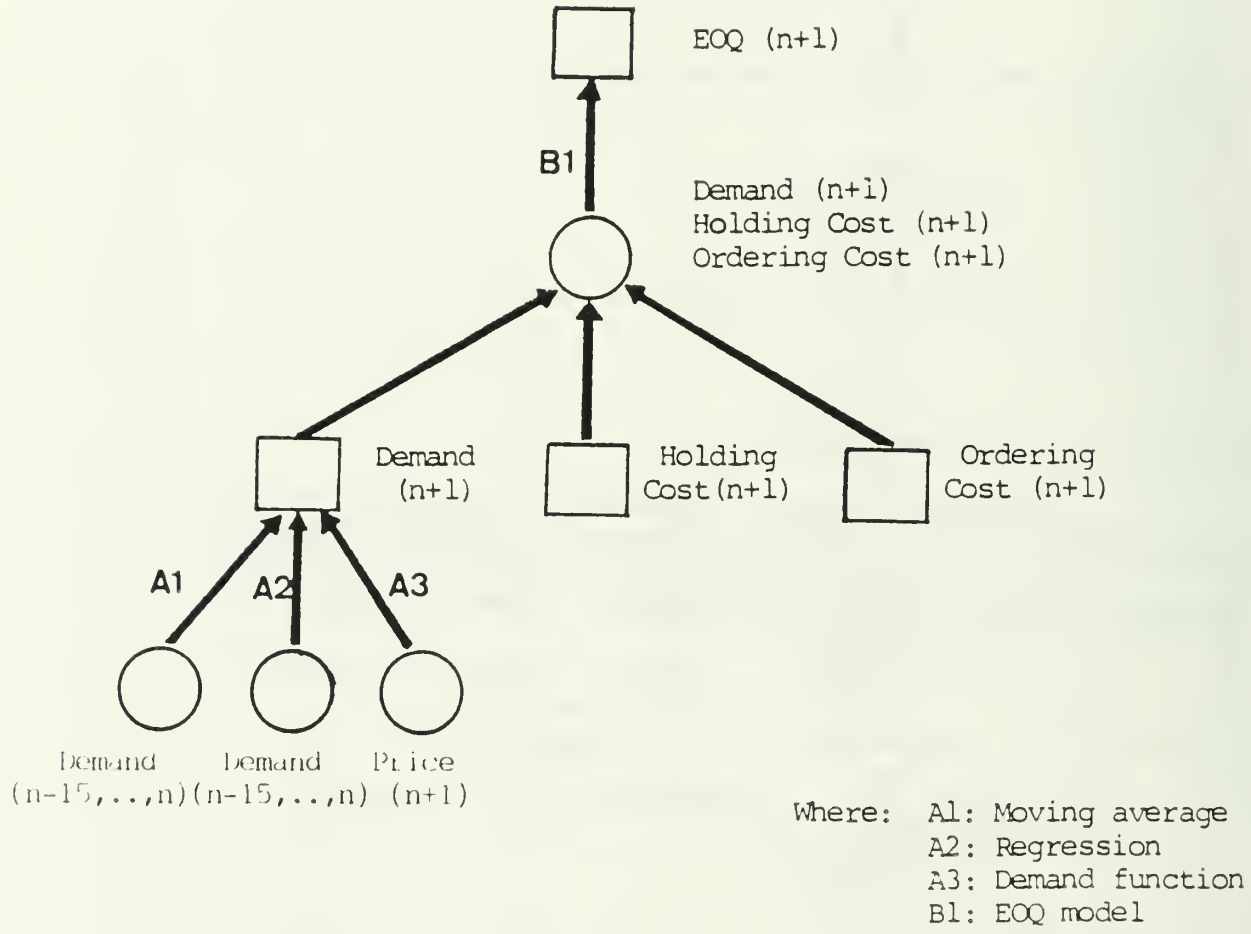


Figure 5. An Alternate AND/OR Graph

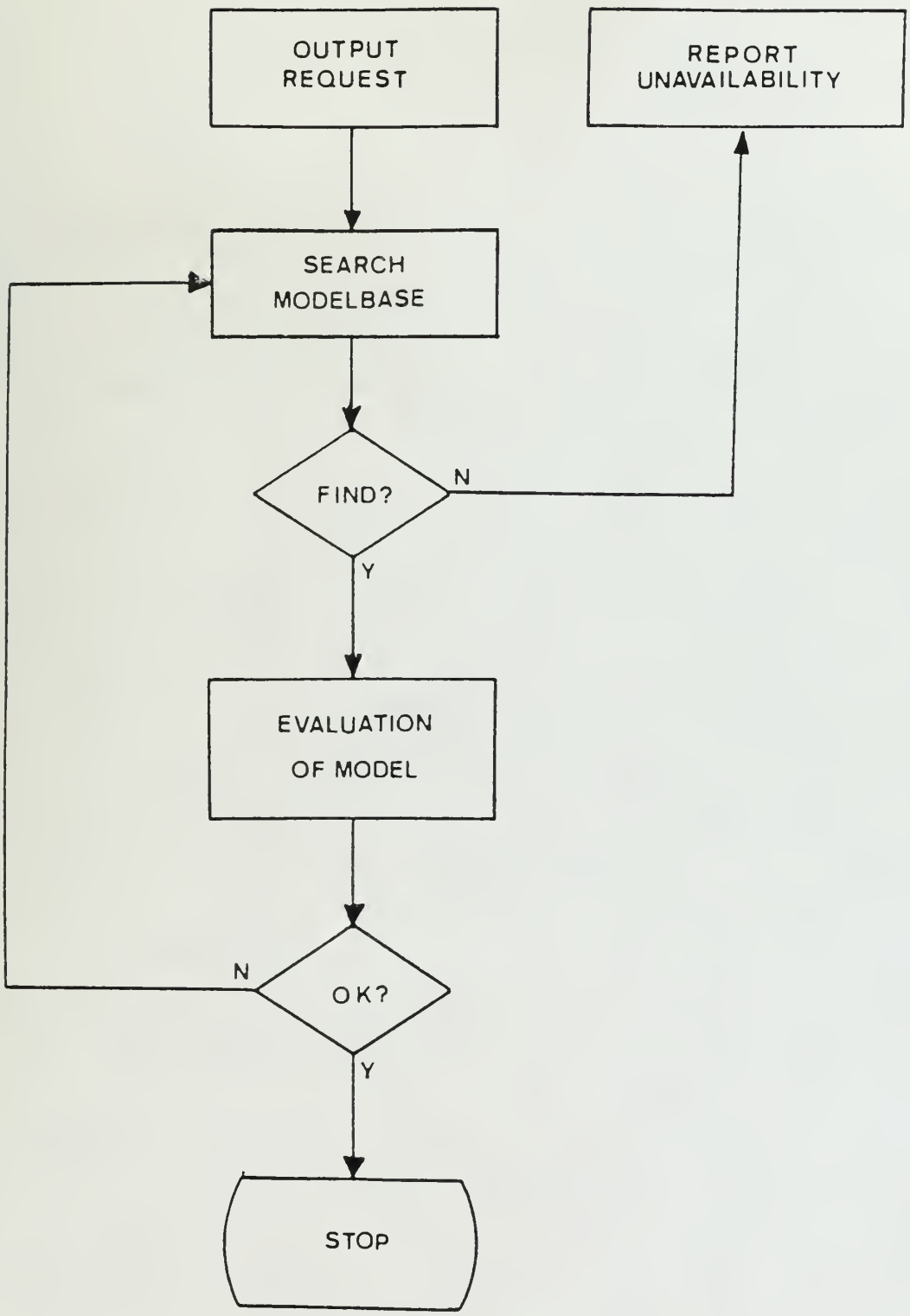


Figure 6. Process of the Satisficing Strategy

```
**********************************************************
**                                                      **
**            TIMOS : QUERY PROCESSING SUBSYSTEM        **
**                                                      **
**********************************************************

Please specify the information you need:

        OUTPUT: eoq
        WHERE:
            product = a
            year = 1987

Please wait while checking the database
'eoq for a for 1987' is not available in the database

I am checking the model base

'demand for a for 1987' is needed but not available in the database
Could you provide it (y/n)? n

'price for a for 1987' is needed but not available in the database
Could you provide it (y/n)? y

Please enter the value: 10

------------------------------------------------------------
                      MY SUGGESTIONS
------------------------------------------------------------

There are three ways to produce the requested information

The first is:

Integrating model 'M1' and model 'M2'

Model 'M1' can generate 'eoq for a for 1987'

The execution of 'M1' needs the following 3 inputs:

      -- holding_cost of a
      -- ordering_cost of a
      -- demand for a for 1987

The database has ..... holding_cost of a = 5
The database has ..... ordering cost of a = 20

'demand for a for 1987' can be produced by executing model 'M2'

The execution of 'M2' needs the following 1 input:

      -- price of a

You provided ..... price of a = 10

Do you want to execute this model (y/n)? y

** eoq of a = 12

More suggestions (y/n)? n

------------------------------------------------------------
                      THANK YOU
------------------------------------------------------------
```

Figure 7. A Sample Session