META-DESIGN CONSIDERATIONS IN DEVELOPING MODEL MANAGEMENT SYSTEMS*

Ting-peng Liang
Department of Accountancy, University of Illinois at Urbana–Champaign, Champaign, IL 61820
Christopher V. Jones
Department of Decision Sciences, The Wharton School, University of Pennsylvania, Philadelphia, PA 19104

ABSTRACT

This paper examines cognitive considerations in developing model management systems (MMSs). First, two approaches to MMS design are reviewed briefly: one based on data-base theory and one based on knowledge-representation techniques. Then three major cognitive issues—human limitations, information storage and retrieval, and problem-solving strategies—and their implications for MMS design are discussed. Evidence indicates that automatic modeling, which generates more complicated models by integrating existing models automatically, is a critical function of model management systems.

In order to discuss issues pertinent to automatic modeling, a graph-based framework for integrating models is introduced. The framework captures some aspects of the processes by which human beings develop models as route selections on a network of all possible alternatives. Based on this framework, three issues are investigated: (1) What are proper criteria for evaluating a model formulated by an MMS? (2) If more than one criterion is chosen for evaluation, how can evaluations on each of the criteria be combined to get an overall evaluation of the model? (3) When should a model be evaluated? Finally, examples are presented to illustrate various modeling strategies.

Subject Areas: Decision Support Systems, Information Management, Information Processing, and Management Information Systems.

INTRODUCTION

Development of model management systems (MMSs) is one of the most important research areas in decision support systems (DSSs). Recently research in this area has increased dramatically. Since 1980, dozens of articles have been published and several frameworks have been proposed (e.g., [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [14], [15], [17], [18], [22], [23], and [36]). Most of this research, however, focused on technical issues and had little concern with the cognitive side of MMS design. Since the objective of an MMS is to support the development and utilization of models in human decision-making processes, it is unlikely that the development of MMSs could succeed without consideration of human cognitive limitations.

The primary purpose of this article is to explore important issues in human cognitive processes and their implications for MMS design. In this article we first will review background issues and current frameworks in terms of their data-base architecture and knowledge-representation techniques. In developing MMSs, we need data-base architectures for model storage and knowledge-representation techniques

*The authors would like to thank James C. Emery and an anonymous associate editor for their helpful comments.

for model representation. Data-base approaches currently adopted for MMS design include the network, relational, and entity-relationship models. The knowledge-representation approaches employed include structures-inheritance networks (SI-nets), predicate calculus, relational algebra, model abstractions, and graph-based and frame-based systems.

Then three major issues in human cognitive processes—human limitations, human information storage and retrieval, and human problem-solving strategies—will be examined. Since human beings are not perfect information processors, many strategies have been used to offset human limitations, such as decomposing a complex problem into subproblems and using rules of thumb to find a satisfactory solution. In order to support human decision making, MMSs must complement these limitations. For example, an MMS must allow the user to decompose a large model into several smaller component models for storage and retrieval; we call this process "model decomposition." Several smaller models also may be combined to create a larger model; this process is called model integration. The model formulated in the model-integration process is an integrated model. Our discussion indicates that the capability of automatic modeling, which performs model integration automatically, is crucial to the usefulness of MMSs. Seven implications for MMS design are derived:

1. The basic modeling unit should be cognitively meaningful chunks of knowledge to the user,
2. MMSs should generate information about the relations of models based on their inputs and outputs,
3. MMSs should be organized in a hierarchy corresponding to the user's cognition,
4. MMSs should provide features for retrieving the models stored in the model base,
5. MMSs should support both satisficing and optimizing strategies,
6. MMSs should support both model integration and model decomposition, and
7. MMSs should support both forward reasoning and backward reasoning in the modeling process.

Finally, a graph-based framework will be adopted to illustrate the issues involved in automatic modeling. The framework describes the processes by which human beings develop models, called "human modeling processes," as route selections on a network of all possible alternatives for a feasible or optimal route (a route represents a model) from the initial state of the problem to the desired final state. Based on the framework, model evaluation becomes crucial to the automatic modeling process. The following three considerations for evaluating an integrated model will be discussed:

1. What are appropriate criteria for evaluating a model? Five criteria will be introduced: (1) accuracy of the models, (2) user's preference for the model, (3) distance from the goal state, (4) number of models integrated, and (5) total cost for modeling and implementation.

2. If more than one criterion is implemented, how can evaluations on each of the criteria be combined to achieve an overall evaluation? The algorithm that combines evaluations on different criteria is called a validity calculus. One validity calculus will be presented in the paper.

3. When should a model be evaluated? Models can be evaluated at three different points: before the data for running the model are retrieved, after the data are retrieved, and after the model is executed.

OVERVIEW OF CURRENT RESEARCH

Background of Model Management Systems

A model management system is a software system which provides a friendly environment for developing new models and managing previously created models. In this paper, a model is defined as a system which provides an abstraction of a real-world problem and is developed to support human decision making. For example, an economic order quantity (EOQ) model is an abstraction of a specific category of inventory problems and can be employed to help decision makers determine the optimal quantity that should be ordered periodically for a particular product. Models managed by an MMS contain data needed to describe the real-world problem and the implicit or explicit relationships among the data. Such models will be assumed to be implemented on a computer. A model base is a repository for the computer-based decision models.

There are at least three major motivations for MMS development: to offset inherent human limitations, to reduce the modeling cost and time, and to keep pace with the evolution of information systems. With regard to the first of these, human beings as information processors have many inherent limitations, including limited short-term memory [28], bounded rationality [33], and other significant biases in decision making [21]. Therefore, many decision models have been developed to support their decision making. A good MMS that facilitates the application of these decision models can improve the quality of decisions significantly.

The second motivation is to reduce the cost and time for developing models. Modeling is an expensive part of the problem-solving process. Based on data collected from a comprehensive survey on 220 federally supported models, the average time and cost for developing one equation was three weeks and six thousand dollars [20]. Another study [31] reported that one-half billion dollars was spent annually on developing, using, and maintaining models in the U.S. government alone, not to mention the money spent in the private sector. Although these estimates are not up to date, they do indicate the importance of model management in an organization. An MMS can simplify the creation and implementation of decision models and hence provide significant economic benefits.

The third motivation is to keep pace with the evolution of information systems. The increased use of data-base management systems has made the integration between models and their associated data critical to most modeling processes. Developing MMSs to provide this integration has been considered a key characteristic of emerging fourth-generation software tools [19].

Overview of Current Frameworks

Early pioneering efforts in MMSs focused on introducing the concept of model management and describing the functions an MMS should have: model description, generation, restructuring, scheduling, execution, and report generation [41] [35]. However, these early works did not provide a general framework for MMS design. Nor did they consider human cognitive aspects of modeling.

Recent research in MMSs follows two principal paths: one borrows from the experience in data-base management system development and focuses on developing an MMS similar to current data-base management systems; the other employs artificial-intelligence techniques, especially knowledge-representation techniques, and concentrates on automating the modeling process. Actually, the two approaches are not exclusive. We need both the data-base architecture to store models and knowledge-representation techniques to represent models appropriately. In Table 1, current frameworks are compared in terms of the data-base model and the knowledge-representation technique adopted.

The Data-Base Approach

The first work in integrating models with data bases is the Generalized Management Information System (GMIS) proposed by Donovan [16]. In this framework, a manager virtual machine (VM) handles all communications between VMs. Although the term "model management" was not mentioned explicitly in this paper, the manager VM functioned as a generalized MMS that managed all machines, including the analytical machine (equivalent to a model), the interface machine, and the data-base machine.

Stohr and Tanniru's framework [36] divided a modeling process into two phases: the model-definition phase and the model-running phase. Models are developed in the model-definition phase and utilized in the model-running phase. Stohr and

Table 1: Summary of current frameworks.

Research	Data-Base Model	Knowledge Representation
Blanning [1] [2] [3] [4] [5] [6]	Relational	Relational calculus
Bonczek, Holsapple, and Whinston [7] [8] [9] [10]	Entity relationship	Predicate calculus
Dolk and Konsynski [14] [15] [22]	Network	Knowledge abstractions
Donovan [16]	Network	Conventional language
Elam [17] and Elam, Henderson, and Miller [18]	Entity relationship	SI nets
Liang [23] [24]	Relational and network	Alternate AND/OR tree
Stohr and Tanniru [36]	Network	Model definition language
Watson [40]	—	Frames

Tanniru considered a model as a composition of many processes. Each process involves the use of a functional group in the context of a model. A functional group is equal to a program that executes a sequence of computations on data and reflects a computational step.

Blanning's framework [1] [2] [3] [4] [5] [6] was based on the relational data model. A model in the framework is defined as a properly restricted subset of the Cartesian cross product of its inputs and outputs. In a series of papers, Blanning described the organization and the implementation of a model bank and the relational completeness of his model query language. Basically his framework represents and manipulates models by their inputs and outputs. For example, a demand model can be expressed as "PRICE→VOLUME."

Liang [23] proposed a multi-level framework which included both relational and network concepts. He argued that the relational representation was better for the external level because it reflected a decision maker's concern about the relationship between the desired and the available information, whereas the network model was better at the internal level with which a model builder or a programmer interacted.

Knowledge Representation in MMSs

Bonczek, Holsapple, and Whinston [7] [8] [9] [10] introduced the application of predicate calculus to model management. They proposed a generalized problem processing system (GPPS) which was not oriented toward a particular application area but designed for building DSSs in various areas. The GPPS contains a knowledge system that integrates a modeling language with a data base containing environmental knowledge.

Elam [17] began with an entity-relationship model and developed a framework containing five components: analyzer, builder, interrogator, processor, and knowledge base. Elam, Henderson, and Miller [18] represented models by SI nets, a graph-based language composed of nodes and links for describing concepts and their interrelationships. They argued that an MMS knowledge base could be represented by four distinct but coupled SI nets: the technical net, the application net, the language net, and the model net. Using different definitions of nodes and edges, Liang employed an alternate AND/OR inference tree to represent the model integration process [24]. An alternate AND/OR tree is a tree on which the AND node and OR node appear alternately.

Dolk and Konsynski [14] [15] [22] proposed a vehicle they called "model abstractions" for model representation. They decomposed a model into three components: data objects, operators (procedures acting upon the data objects), and assertions about how the data objects and operators interact.

Watson's [40] research was an extension of Dolk and Konsynski's model-abstraction approach. Watson proposed the application of frames to model management where a frame is defined as a data structure for representing stereotyped information about a situation. He argued that model abstractions are better suited for cataloging models, but the integration between frames and model abstractions would provide better management of the model base.

This brief review shows that most of the previous research focused on the technical issues involved in developing MMSs, such as how a model should be represented in the model base. Since the objective of MMSs is to support the modeling process, which is more art than science, the development of MMSs is by no means a technical issue only. Cognitive issues pertinent to the process also must be considered.

COGNITIVE ISSUES IN MMS DESIGN

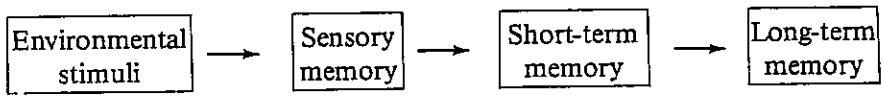
Cognition can be considered a modeling process in which a model is created to draw deductions. A cognitive model is developed by a person when carrying on a discourse with another person or a computer. An invalid or inaccurate cognitive model will lead to user dissatisfaction, a lack of confidence in the system, and hence inefficient use of the system and human resources [32]. Therefore the process by which an MMS manipulates models should support its users' cognitive models.

There are at least three cognitive issues that should be considered in developing an MMS:

1. Human cognitive limitations
2. Human information storage and retrieval
3. Human problem-solving strategies.

Human Cognitive Limitations

The process by which a human absorbs and uses information can be briefly described by the following diagram [11]:



Sensory memory retains environmental stimuli for a very short time (one second), screens out some of the stimuli, and then passes selected information to short-term memory for further processing. Short-term memory (STM), often referred to as working memory, is an active memory that accepts information passed from sensory memory for processing. The processed information is converted to a more durable form and then transferred into long-term memory (LTM). STM loses information within about 15 seconds.

The major human limitation in information processing is the STM capacity. Based on the well-known research findings of Miller [28], STM can handle only about seven "chunks" of information at any given time. Some researchers even have argued that seven is too large [26] [29]. A chunk is defined as a "maximal familiar substructure of a stimulus" [34, p. 80]. In general, a chunk is equivalent to a stimulus that has a unitary representation in LTM. For example, "CAT" can be considered as one chunk for most people. But for a child who has not learned what "CAT" means, it consists of three chunks, "C," "A," and "T."

Since solving a problem involves both retrieval of information from LTM as well as processing and maintenance of current information in STM, the limited STM capacity indicates that human beings do not have sufficient means for storing information in memory to enable them to apply the efficient strategy for problem solving [34]. To offset this limitation, human beings store and retrieve information in chunks and organize chunks of knowledge in a hierarchy. Grouping information into larger chunks can reduce significantly the information load on STM and hence increase information-processing capabilities.

The limited capacity of STM indicates that to support human decision making MMSs should store and manipulate models in a way that is compatible with the user's cognitive models. That is, all operations supported by an MMS—including model storage, model retrieval, model integration, model aggregation, and model decomposition—should be performed based on cognitively meaningful chunks rather than on mathematically meaningful units. For example, in storing and manipulating the EOQ inventory model, the whole model rather than the square-root function for computing the EOQ must be considered as a basic unit. To most decision makers, the EOQ model as an inventory control model is cognitively meaningful, whereas the square-root function is not.

The primary implication of this concept for MMS design is that large-scale models can be decomposed into smaller component models and stored or manipulated as such by the MMS. Each component model must be a cognitively meaningful unit. For example, the production scheduling model described in Exhibit 1 can be decomposed into many component models. The inputs, outputs, and relationships among those component models are illustrated in Figure 1. In the figure, each component model is both a stand-alone model in the model base and a component of the large-scale model. This process is called "model decomposition."

The circled area in Figure 1 represents a production model in which the overall production is a function (in this case, a summation function) of the regular production and the overtime production. On the one hand, the model can be used in isolation to determine the overall production. On the other hand, it also is a component of the production-scheduling model. Once it has been executed, the overall production can be fed into the succeeding component models to generate outputs including the sales volume, ending inventory, and monthly material cost. In the following sections, we shall find decomposition and integration critical to most modeling processes.

The decomposition of a large-scale model into smaller cognitively meaningful units is by no means a unique process. Different users may decompose the same model differently. For most models, however, there is a baseline of decomposition. For instance, the previously described model for calculating the overall production is a nondecomposable basic unit because further decomposition is impossible without losing its semantic meaning. The EOQ model also is a basic cognitive unit. Basic cognitive units may be called basic models or basic modeling units. MMSs must perform operations based on these basic models and at the same time support different ways of "chunking" them to meet the requirements of different users. Here "chunking" means the process by which a large model is decomposed into

Exhibit 1: A production-scheduling model.

Mr. Rosselli, president of the Victor Chemical Company, needs a decision support system that can provide the following information, broken down by month along with annual totals:

1. Production schedule
   - Regular-time production
   - Overtime production
   - Total production
   - Demand
2. Inventory report
   - Beginning balance
   - Monthly supply
   - Average inventory
   - Expected stockout cost
   - Warehousing cost
3. Production-change report
   - Change in production
   - Cost of change
4. Operations-cost report
   - Direct labor
   - Direct material
   - Indirect cost
   - Overtime cost
   - Production-change cost
   - Stockout cost
   - Warehousing cost
   - Total operations cost
   - Average cost per pound
5. Summary financial plan
   - Sales revenue
   - Cost of goods sold
   - Administrative cost
   - Interest
   - Total cost
   - Net profit before taxes
   - Taxes
   - Net profit after taxes
6. Cash-flow analysis
   - Beginning cash balance
   - Cash receipts
   - Operation expenditures
   - Net from operations
   - Interest payments
   - Taxes
   - Ending cash before adjustment for debt
   - Change in debt (+ or −)
   - Ending cash balance
   - Ending debt

small components for storage and then aggregrated to support a particular decision-making task. We shall call these different ways of chunking knowledge different "views" of models. Different views reflect different cognitions of different users.

For example, there are two ways to treat the model that forecasts sales by computing forecasted demand based on historical data and then multiplying the forecasted demand by the sales price. One considers the model as the integration of two chunks of knowledge: demand forecasting and calculation of sales. The other considers the model as a single chunk of knowledge. To be useful an MMS must be able to support both. By integrating existing basic models, many different views of a large model can be created.

Information Storage and Retrieval

In addition to complementing the chunking and hierarchical nature of human knowledge, an MMS should support two other features of the process by which human beings store and retrieve information. First, human beings store not only data but also the relations among the data. Second, human memory is richly indexed for access in the presence of appropriate stimuli.

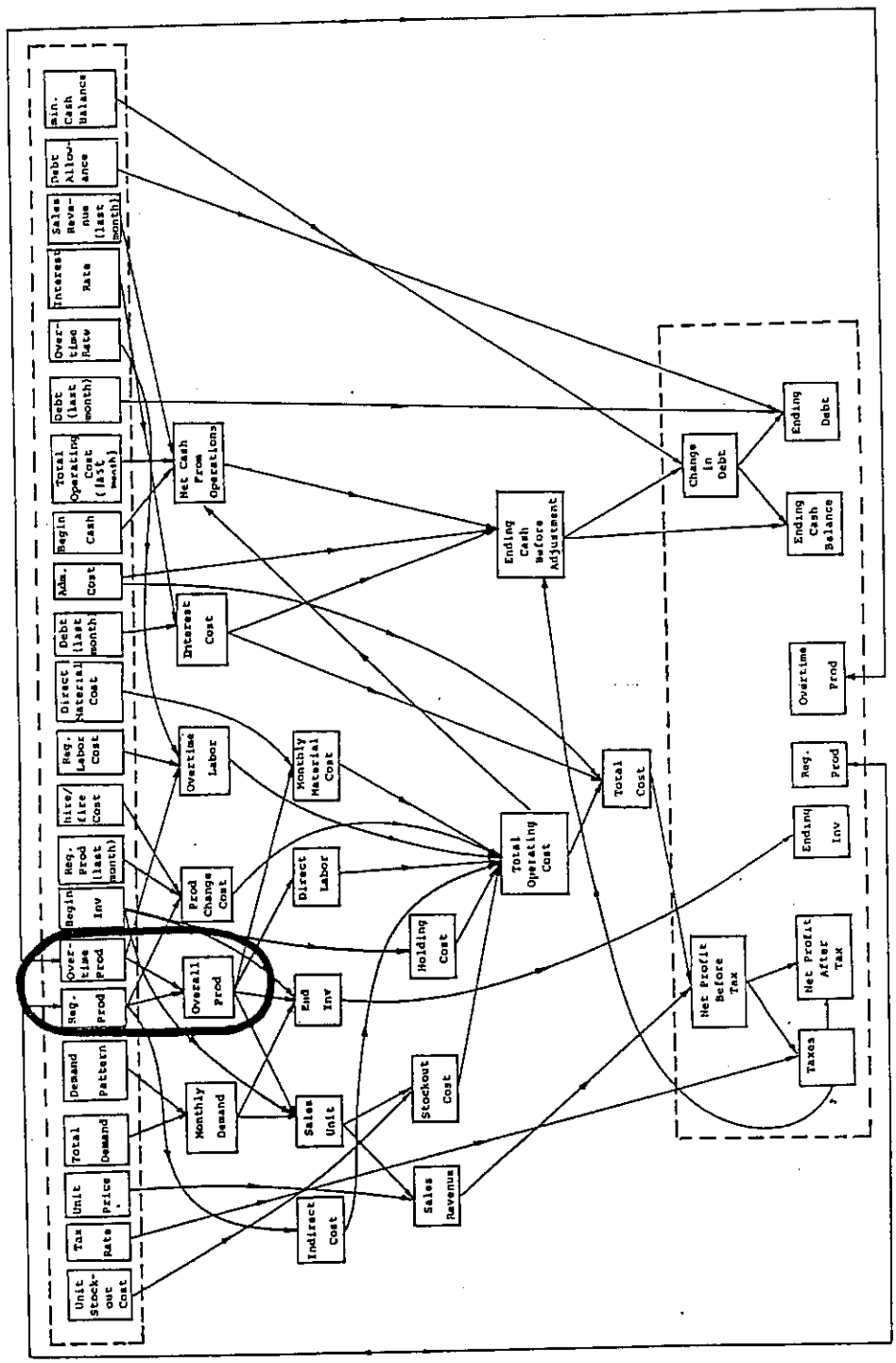


Figure 1: Decomposition of the production scheduling model.

Storage of Data and the Relations among Data

A well-known experiment on chess perception conducted by De Groot [13] provided evidence to support the argument that human beings store not only data but also the relations among the data. In the experiment, chess masters presented with chess positions taken from actual games significantly outperformed novices in reconstructing the positions. The masters' performances however, were no better than those of the novices when they were shown a chess board with randomly arranged pieces. Simon [34] therefore concluded that information about the board is stored in the form of relations among the pieces, rather than as a "television scan" of the 64 squares.

Since people store both relations and entities, an MMS should handle not only individual models but also relations among the models. Here a model in the model base is similar to a chess piece on a board. Since most models are composed of functions that convert a set of input data to a set of output data, the inputs, outputs, and functions for data conversion are three major features that must be managed. The relations among models can be created by matching the outputs of a model with the inputs of another model [24]. By capturing the relations among models, an MMS will be able to provide information about effective use of the models available in the model base. This point is particularly important to the development of the automatic modeling capability in MMSs, which automatically can create more complicated models for producing the desired information by integrating existing basic models.

Indexing Memory for Access in the Presence of Stimuli

The indexed structure of memory enables a human being to draw upon a large repository of information and accounts for the complexity of human behavior. In a series of experiments, Tulving [37] and his associates [38] [39] found that factors affecting human problem solving included both the availability and the accessibility of information. In the experiments, subjects giving category names as cues recalled more items previously shown to them than the subjects in the control group. However, the latter recalled almost the same number of items when they were supplied with the category names. This result indicates that human beings benefit from comprehensive indexing and cues.

The implication of this feature for MMS design is that MMSs should provide a comprehensive indexing function in order to reduce the workload on STM. In other words, an MMS should provide not only useful functions to support the processes by which human beings develop computerized decision models (called "human modeling processes" in this paper) but also useful cues to help users to access those functions. In short, an MMS should allow users to retrieve models or functions by specifying features recognized by users. For example, a user having some knowledge of mathematical programming should be able to retrieve a goal programming model by specifying features such as optimization, multiple objectives, or linear constraints.

Problem-Solving Strategies

In solving a complex problem, three strategies frequently are employed to offset human limitations: (1) decomposition and integration, (2) forward and backward reasoning, and (3) satisficing and optimizing. To help a user develop and work with decision models, an MMS should facilitate the use of these strategies and, in fact, implement these strategies.

Decomposition and Integration

One strategy human beings use to reduce the cognitive burden is to decompose a complex problem into semi-independent components corresponding to its functional parts, solve each component individually to achieve a partial solution, and then integrate those partial solutions to reach a solution for the whole problem. In previous sections, we stated that an MMS should support decomposition and integration.

The process of integration could be either manual or automatic. In other words, the user may specify the process of integration (user-assisted modeling) or request the system to do it automatically (automatic modeling). A powerful MMS, however, must support both.

Developing the capability of automatic modeling in MMSs is important but difficult. Models in the model base must be represented appropriately, and the MMS must be able to search models for solving a particular problem and then select the proper model according to pre-specified criteria. Issues pertinent to the implementation of automatic modeling will be discussed in later sections.

Forward and Backward Reasoning

In most problem-solving situations, the initial state of the problem and the desired final state are known. A problem solver must determine a way to transform the initial state into the final state. Two strategies generally are used: forward reasoning and backward reasoning. Forward reasoning requires the decision maker to reason from the initial state to the final state. Backward reasoning is the reverse: the decision maker reasons from the final state to the initial state.

Forward and backward reasoning strategies are especially important in model integration. If the model for solving a given problem does not exist in the model base but can be created by integrating existing models, then the user must decide whether to start the model integration process from the final state or from the initial state. MMSs should support both strategies.

Satisficing and Optimizing

No matter whether the backward or forward reasoning strategy is adopted, the strategy for selecting one model among all candidates can be either satisficing or optimizing. Although optimization is the common objective in many sciences,

a satisficing strategy is widely adopted in human decision making. In a satisficing strategy, alternatives are examined as they become available, and the first which satisfies all decision criteria is accepted for implementation. As argued by Simon [33], most human decision making, whether individual or organizational, is concerned with the discovery and selection of satisfactory alternatives; only in exceptional cases is it concerned with the discovery and selection of optimal alternatives. Therefore, MMSs should support both satisficing and optimizing strategies of human modeling.

In addition to the human cognitive limitations, time and cost considerations are major reasons for the satisficing strategy. In some situations, the decision maker simply does not have time to wait for the development of a complex optimization model. This strongly supports the need for automatic modeling. That is, MMSs should provide capabilities for *ad hoc* support that automatically formulates a satisfactory model for decision makers in a short time period.

Summary of Cognitive Considerations

In summary, the development of MMSs should consider both technical issues and the following cognitive issues:

1. All operations in MMSs, including model integration and model retrieval, should be performed based on cognitively meaningful chunks rather than mathematically meaningful units.
2. Each model in the model base should be represented as a chunk of knowledge that includes associated inputs and outputs, and the MMS should be able to generate information about the relations of the models based on the inputs and outputs.
3. Models in the model base should be organized in a hierarchy and indexed comprehensively.
4. MMSs should handle not only models but also features for retrieving them.
5. MMSs should support both satisficing and optimizing strategies for modeling.
6. MMSs should support both model decomposition and model integration.
7. MMSs should support both forward reasoning and backward reasoning.

CONSIDERATIONS IN AUTOMATIC MODELING

Automatic modeling includes two major processes: searching the model base to find appropriate models and integrating those selected models automatically. It attempts to perform some of the chores human beings undertake in assembling larger models from basic models. Our discussions in previous sections have indicated that automatic modeling is an important function for model management. Developing an automatic modeling capability, however, involves some difficult issues:

1. how to represent the models to be integrated and the interactions (relations) among the models
2. how to use heuristics (rules of thumb) to reduce the search space

3. how to develop a metric (an evaluation function) for measuring the applicability of an integrated model to a user's problem.

In this section, issues surrounding automatic modeling will be illustrated in the context of a graph-based framework. First, the graph-based framework for representing models and the interactions of models will be described briefly. A detailed description can be found in [24]. Then heuristic search and the development of evaluation functions will be discussed. One of the most significant advantages of the graph-based framework is that many heuristics developed in graph theory and artificial intelligence can be applied with minimum modification to the development of MMSs. Adoption of the graph-based framework, however, does not imply that this is the only appropriate framework, nor are the issues to be discussed specific to the graph-based framework.

The Graph-Based Framework for Representation

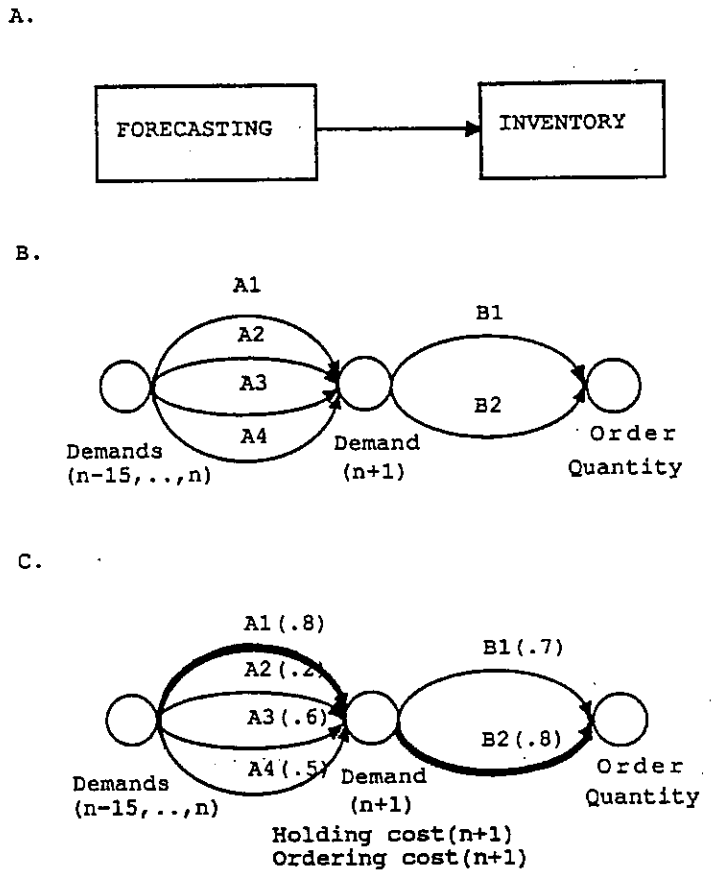
As "problem solving is often described as a search through a vast maze of possibilities" [34, p. 66], so can the process by which human beings develop decision models to support their decision making be described as a search through a number of possible relationships in order to find a route that can convert the initial state (available information) of a problem to the desired final state (output information). By this definition, we can represent models to be integrated and the relations among the models as a network. Each node on the network represents a set of data attributes, and an edge between two nodes represents a set of functions that convert the input data to their associated output. A model is composed of two nodes and one edge connecting the two nodes.

Since there usually is 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, there may exist more than one model for solving a problem. For example, if one wants to forecast sales for the next year by using the sales of the last 15 years, one can use the moving average, exponential smoothing, regression, or the Box-Jenkins approach. Different approaches provide different forecasted sales and have different validities. Here validity is a value representing the applicability of a model to a particular problem. It usually is generated by a model-evaluation function. Detailed discussion will be provided in the next section. The objective of modeling, therefore, is to find a route which starts at the initial state and ends at the final state. The route may be a satisfactory one (satisficing strategy) or the one with the highest validity (optimizing strategy).

Since a set of sales forecasting models is considered a chunk of knowledge for most decision makers, the problem may be classified as a one-stage problem. Here, a stage is defined as an application involving one cognitively meaningful chunk of knowledge. For example, the functions for converting the previous years' sales (initial state) to the next year's sales (desired state) are cognitively meaningful to most persons who have a background in forecasting. They therefore are counted as a single chunk.

Not all problems are as simple as this example, however. For instance, Figure 2B illustrates a two-stage problem. In graphical terms, a stage is defined as two nodes

Figure 2: A two-stage modeling process.



and the associated edges connecting the two nodes. The combination of the node representing demand (n+1), the node representing demands (n−15, . . ., n), and edges A1, A2, A3, and A4 in Figure 2B constitutes one stage.

According to these concepts, we can define various strategies for modeling in graphical terms.

1. Model decomposition and integration. The decomposition of a model is a process that decouples a multistage model into a set of submodels, and each submodel is part of the decomposed model. Model integration is a process that connects more than one model by the data set they share to create a multistage model.

2. Forward and backward reasoning. In the model-integration process, there are two strategies for searching a route on a network. The forward reasoning strategy is defined as a process that searches from the input node to the desired output node. The backward reasoning is the reverse; it searches from the output node to the input node.

3. Satisficing and optimizing. The satisficing strategy is a trial-and-error process for searching a route that can convert the initial state to the desired final state; the process continues until one route, which fulfills all of the pre-specified criteria, is found or all routes are proved faulty. The optimizing strategy is the process of maximizing the validity of the searched route under the constraints of time and cost.

Search and Evaluation of Models

Regardless of which strategy is employed, the implementation of automatic modeling in MMSs requires algorithms for searching and integrating appropriate models in the model base and functions for evaluating various component models. Since these two issues are highly related, our discussions will be interwoven in this section.

If the size of the model base is small, an exhaustive search may be possible. In some cases, however, the model base may be huge. Since the time required to find the optimum model increases exponentially with the size of the model base, an exhaustive search is not feasible because of combinatorial explosion, and the modeling process becomes intractable. In this case, heuristics are required to reduce the search space.

A heuristic is a rule of thumb used in problem solving. Functions for evaluating models in order to reduce the search space in the modeling process are called model-evaluation functions. The value generated by a model-evaluation function is called validity because it represents a measure of the applicability of a model to a problem. Here validity is not necessarily associated with the accuracy of the model. For instance, although distance from the goal state is not a measure of the quality of a model, it is a good metric for offsetting the combinatorial explosion problem in a model-integration process. This kind of search usually is called a heuristic search because the evaluation functions typically are based on heuristics [12].

Validity usually is a multiattribute measure. To implement a multiattribute model-evaluation function, the following three issues must be considered:

1. What are proper criteria for evaluating models?
2. How can several evaluations on each of the criteria be combined to obtain the validity of the integrated model?
3. When should the validity be determined?

All these issues are difficult. Hence, the following discussions should be considered as a starting point rather than a complete solution.

Criteria for Evaluation

There are at least five possible criteria for model evaluation in a modeling process: (1) accuracy of models, (2) the user's preference for a model, (3) distance from the goal state, (4) number of models integrated, and (5) total cost.

The accuracy of models perhaps would be the best measure since it guarantees that the model selected is the most accurate one. Evaluating the accuracy of models, however, is difficult. Different categories of models may need different evaluation

methods. For instance, goodness of fit is a good criterion for evaluating forecasting models but is useless in evaluating different cost-allocation methods.

In order to implement this criterion in the modeling process, the MMS must keep track of what evaluation function is suitable for what categories of models and apply the function appropriately. For example, if a modeling process involves sales forecasting models, then a function for measuring the degree that the forecasted sales data fit real sales data may be executed on each model to compute its validity value. The major difficulties of this approach are (1) how to find good evaluation functions for all models in the model base and (2) how to combine the validities of component models evaluated based on different evaluation functions into a meaningful measure of the integrated model.

If we want a unified measure for all kinds of models, then measuring a user's preference for each model is a good candidate. User preference is a subjective criterion and can be implemented based on a predefined scale such as from 0 to 1. The basic rationale behind this criterion is that an MMS must provide users with models they trust. Since different users may prefer different models for solving a specific problem, the MMS must maintain a data base of user preference. Every time a model is used, the data base must be updated. It certainly is possible that the model preferred by a user is not the best one in terms of accuracy or other technical criteria. In this case, the MMS may provide this information to the user by executing another evaluation function which uses the first criterion.

There are two major difficulties in implementing this criterion: (1) for many users, their preferences for different models may not differ significantly; and (2) users may not have enough knowledge or information for making an intelligent judgment on the applicability of a particular model to their problems.

The third criterion is distance from the goal state. The number of stages an integrated model has is considered a measure of such distance. Because overall validity of an integrated model is a function of the validities of its component models, it is plausible to assume that the fewer stages in a network, the higher the overall validity of the integrated model. For example, a sales forecasting model that forecasts future sales directly from historical sales data (a one-stage model) is likely to be better than the model that integrates one model for forecasting sales volume and another model for forecasting future price (a two-stage model).

A measure similar to the distance from the final state is the number of models involved in an integrated model. Instead of minimizing the number of stages, this measure minimizes the total number of component models.

Total cost that occurs in the modeling and implementation processes is another possible criterion for model evaluation. Applying this criterion requires certain functions to assess the modeling and implementation costs. This turns out to be the major difficulty. It is especially difficult to estimate system implementation costs accurately.

A Validity Calculus

Since an integrated model combines more than one component model, a validity calculus, which calculates the validity of the integrated model from the validities

of its component models, is required. The definition of a calculus depends on what criteria are employed for evaluating models. For example, if a user's preference has been used to measure validities of models, then the validity calculus may be defined as follows:

1. The validity at each stage equals the maximum validity of the component models at the stage. For instance, the validity of the forecasting stage in Figure 2C equals .8.
2. The overall validity of an integrated model equals the product of its component models at each stage. For instance, the validity of the integrated model A1-B2 in Figure 2C equals .64 (.8×.8).

This certainly is not the only legitimate validity calculus. Depending on the adopted criteria for model evaluation and the requirements of a particular organization, various validity calculi may be defined in different MMSs. A complete discussion and justification of evaluation functions, however, are outside the scope of this paper.

A Search Heuristic

With these measures of validity, a simple heuristic as described below will be able to reduce the search space effectively. More heuristics can be found in [12] or [30].

1. Calculate validity values for all candidate models.
2. If more than one model is available at a particular stage, eliminate dominated models (models with lower validities).
3. If more than one model in the stage survives the second step, choose one arbitrarily or ask the user to select one.
4. Integrate the model selected in steps 2 and 3 to formulate a network graph.
5. Select the path with the highest validity in the network.

When the Model Evaluation Function Should Be Performed

In an automatic modeling process, the evaluation function can be performed at three different points: (1) before any data for model execution are retrieved, (2) during the process of model integration, and (3) after the selected model is executed.

Since a model base usually contains a huge amount of models, it is important to evaluate these models before retrieving data from the data base. The primary purpose of applying the model evaluation function at this stage is to reduce the search space. If a model is found unsatisfactory, it should be dropped from further consideration. The result of this process is a set of selected models for model integration.

After screening the model base, the selected models can be integrated to produce the desired information. Because more than one integrated model may be created in this process, a function is required to evaluate these integrated models and select the appropriate one. Hence, the primary purpose of applying the model-evaluation function here is to provide a basis for model selection.

Finally, the evaluation function may be applied to evaluate the performance of the selected model after it is executed. This step provides feedback information important for possible improvement in the future.

Although applying model-evaluation functions complements the search heuristic discussed in the previous section, it is not without risk. An inappropriate model-evaluation function may omit good models and provide the decision maker with misleading information. Therefore it is important that, on the one hand, the user is aware of the strengths and limitations of the model-evaluation function employed in a particular MMS. On the other hand, the designer continuously must evaluate and update the model-evaluation function.

An interesting issue associated with the risk of using model-evaluation functions is whether the feedback obtained from executing the selected model can be used to update the model-evaluation function automatically. In order to have this capability, an MMS must have a powerful learning mechanism. Currently, development of such mechanisms is still in its infancy. Interested readers may refer to [27].

ILLUSTRATIVE EXAMPLES

In order to illustrate the process of model integration and various human problem-solving strategies, examples are presented in this section.

Given the demands for the last 15 months, the carrying cost, and the ordering costs, the user wants to determine the order quantity of part *P* for the next month. In order to obtain the desired information, two chunks of knowledge may be required: one for demand forecasting and the other for inventory control (see Figure 2A).

If there are four approaches for demand forecasting and two approaches for inventory control, the modeling process involves a search among eight (2×4) possible models, as shown in Figure 2B. Suppose the validities of those component models (the number associated with the edges in Figure 2C) are preferences of the user for those models; the strategies for modeling can be described as follows.

Decomposition and Integration

Since no cognitively meaningful tool is directly available for calculating the order quantity for the next month, the problem first is decomposed into two subproblems: forecasting and inventory (Figure 2A). Then models for solving the two subproblems are retrieved from the model base. Finally, the two sets of models are integrated to formulate all candidate models (Figure 2B).

Forward and Backward Reasoning

In searching for a route, there are two strategies: forward reasoning and backward reasoning. If the system takes the forward reasoning strategy, then the process will be as follows:

1. Search all forecasting models that can convert past demands (the initial state) to the forecasted demand (the goal state).

2. Search all inventory models that convert demand to order quantity (the final state).
3. Combine all models and get eight feasible alternatives.
4. Select a model automatically based on pre-specified objective functions.

The backward reasoning strategy is the reverse. The system first searches the inventory model to provide an order quantity; it then searches the forecasting model whose output is the input of each selected inventory model. Those combinations that match both the initial state and the final state are feasible alternatives for selection.

Satisficing and Optimizing

After determining the strategy for reasoning, the strategy for selecting a model must be determined. In the satisficing strategy, if the minimum satisfactory value is .5, the user will accept model A1-B1 without further examining other alternatives because its validity ($.56=.8\times.7$) is higher than .5. In the optimizing strategy, however, the user will search all paths and select A1-B2 ($.64=.8\times.8$), the darkened path shown in Figure 2C.

By implementing the search heuristic described in the previous section, the formulated graph becomes much simpler. In this example it contains only one path. Since validity data already are available, the system first eliminates models A2, A3, and A4 in Figure 2C because they are dominated by A1. Model B1 in the second stage also is dropped because B2 has higher preference value. Then the system integrates A1 and B2. This combination has the highest validity.

CONCLUSION

Given our view of the modeling process as a cognitive process, the development of MMSs is not merely a technical puzzle. The primary objective of this article has been to investigate cognitive issues that must be considered in developing MMSs.

In this paper, we first explore human cognitive limitations, including human information storage and retrieval, human problem-solving strategies, and their implications for MMS design. Since evidence indicated that automatic modeling is crucial to model management, three issues for implementing automatic modeling were discussed in the context of a graph-based framework. The framework describes the process by which human beings develop decision models to support their decision making as a path-finding process on a network which captures all feasible alternatives. These issues are

1. What are proper criteria for evaluating models?
2. How can several evaluations based on different criteria be combined to obtain the overall validity of an integrated model?
3. When should the validity be determined?

Five criteria for model evaluation and a validity calculus for computing the validity of an integrated model were introduced. Three potential points for applying model evaluation functions were suggested. They are (1) before the required data are retrieved, (2) after the data are retrieved but before the model is executed, and (3) after the model is executed. A heuristic for reducing the search space was also presented.

[Received: September 10, 1985. Accepted: January 29, 1987.]

REFERENCES

[1] Blanning, R. W. Model structure and user interface in decision support systems. In *DSS-81 Transactions.* Providence, RI: The Institute of Management Sciences, 1981.

[2] Blanning, R. W. A relational framework for model management in decision support systems. In *DSS-82 Transactions.* Providence, RI: The Institute of Management Sciences, 1982.

[3] Blanning, R. W. Issues in the design of relational model management systems. In *AFIPS Conference Proceedings.* Reston, VA: American Federation of Information Processing Societies, 1983.

[4] Blanning, R. W. Conversing with management information systems in natural language. *Communications of the ACM,* 1984, *27,* 201-207.

[5] Blanning, R. W. Language design for relational model management. In S. K. Chang (Ed.), *Management and office information systems.* New York: Plenum Press, 1984.

[6] Blanning, R. W. A relational framework for join implementation in model management systems. *Decision Support Systems,* 1985, *1,* 69-82.

[7] Bonczek, R. H., Holsapple, C. W., & Whinston, A. B. The evolving roles of models in the decision support systems. *Decision Sciences,* 1980, *11,* 337-356.

[8] Bonczek, R. H., Holsapple, C. W., & Whinston, A. B. *Foundations of decision support systems.* New York: Academic Press, 1981.

[9] Bonczek, R. H., Holsapple, C. W., & Whinston, A. B. The evolution from MIS to DSS: Extension of data management to model management. In M. J. Gintzberg, W. Reitman, & E. A. Stohr (Eds.), *Decision support systems.* Amsterdam: North Holland, 1982.

[10] Bonczek, R. H., Holsapple, C. W., & Whinston, A. B. Specification of modeling and knowledge in decision support systems. In H. G. Sol (Ed.), *Processes and tools for decision support.* Amsterdam: North-Holland, 1982.

[11] Bourne, L. E., Dominowski, R. L., & Loftus, E. F. *Cognitive processes.* Englewood Cliffs, NJ: Prentice-Hall, 1979.

[12] Charniak, E., & McDermott, D. *Introduction to artificial intelligence.* Reading, MA: Addison-Wesley, 1985.

[13] De Groot, A. D. Perception and memory versus thought: Some old ideas and recent findings. In B. Kleinmuntz (Ed.), *Problem solving.* New York: Wiley, 1966.

[14] Dolk, D. R. *The use of abstractions in model management.* Unpublished doctoral dissertation, University of Arizona, 1982.

[15] Dolk, D. R., & Konsynski, B. R. Knowledge representation for model management systems. *IEEE Transactions on Software Engineering,* 1984, *SE-10,* 619-628.

[16] Donovan, J. J. Database system approach to management decision support. *ACM Transactions on Database Systems,* 1976, *1,* 344-369.

[17] Elam, J. J. Model management systems: A framework for development. In *Proceedings of the 1980 Southeast American Institute for Decision Sciences.* Atlanta, GA: Decision Sciences Institute, 1980.

[18] Elam, J. J., Henderson, J. C., & Miller, L. W. Model management systems: An approach to decision support in complex organization. In *Proceedings of the First International Conference on Information Systems.* Chicago, IL: Society for Information Management, 1980.

[19] Emery, J. C., Knapp, H. W., & Zisman, M. D. *An advanced business system integrating fourth generation tools and concepts* (83-06-02). Unpublished working paper, University of Pennsylvania, Wharton School, 1983.

[20] Fromm, G., Hamilton, W. L., & Hamilton, G. E. *Federally supported mathematical models: Survey and analysis.* Washington, DC: National Science Foundation, 1974.

[21] Huber, G. P. *Managerial decision making.* Glenview, IL: Scott, Foresman, 1980.

[22] Konsynski, B. R., & Dolk, D. R. Knowledge abstractions in model management. In *DSS-82 Transactions.* Providence, RI: The Institute of Management Sciences, 1982.

[23] Liang, T. P. Integrating model management with data management in decision support systems. *Decision Support Systems,* 1985, *1,* 221-232.

[24] Liang, T. P. A graph-based approach to model management. In *Proceedings of the Seventh International Conference on Information Systems.* Chicago, IL: Society for Information Management, 1986.

[25] Maier, N. R. F., & Burke, R. J. Test of the concept of "availability of functions" in problem solving. *Psychological Reports*, 1966, *19*, 119-126.
[26] Mandler, G. Organization of memory. In K. W. Spence & J. T. Spence (Eds.), *The psychology of learning and motivation* (Vol. 1). New York: Academic Press, 1967.
[27] Michalski, R. S., Carbonell, J. G., & Mitchell, T. M. *Machine learning: An artificial intelligence approach.* Los Altos, CA: Morgan Kaufmann, 1983.
[28] Miller, G. A. The magic number seven, plus or minus two: Some limitations on our capacity to process information. *Psychological Review*, 1956, *63*, 81-97.
[29] Murdock, B. B., Jr. *Human memory: Theory and data.* Potomac, MD: Lawrence Erlbaum, 1974.
[30] Rich, E. *Artificial intelligence.* New York: McGraw-Hill, 1983.
[31] Roth, P. F., Gass, S. I., & Lemoine, A. J. Some considerations for improving federal modeling. In *1978 Winter Simulation Conference Proceedings.* New York, NY: The Institute of Electrical and Electronics Engineers, 1978.
[32] Saja, A. D. The cognitive model: An approach to designing the human-computer interface. *ACM SIGCHI Bulletin*, 1985, *16*(3), 36-40.
[33] Simon, H. A. *The new science of management decision.* New York: Harper, 1960.
[34] Simon, H. A. *The sciences of the artificial* (2nd ed.). Cambridge, MA: MIT Press, 1981.
[35] Sprague, R. H., & Watson, H. J. Model management in MIS. In *1975 Proceedings.* Atlanta, GA: Decision Sciences Institute, 1975.
[36] Stohr, E. A., & Tanniru, M. R. A database for operations research models. *International Journal of Policy Analysis and Information Systems*, 1980, *4*, 105-121.
[37] Tulving, E. Episodic and semantic memory. In E. Tulving & W. Donaldson (Eds.), *Organization of memory.* New York: Academic Press, 1972.
[38] Tulving, E., & Pearlstone, Z. Availability versus accessibility of information in memory for words. *Journal of Verbal Learning and Verbal Behavior*, 1966, *5*, 381-391.
[39] Tulving, E., & Thomson, D. M. Encoding specificity and retrieval processes in episodic memory. *Psychological Review*, 1973, *80*, 352-373.
[40] Watson, G. W. *Knowledge based management for model management.* Unpublished master's thesis, Naval Postgraduate School, 1983.
[41] Will, H. J. Model management systems. In E. Grochla & N. Szyperski (Eds.), *Information systems and organization structure.* Berlin: Walter de Gruyter, 1975.

Ting-peng Liang is Assistant Professor in the Department of Accountancy, College of Commerce and Business Administration, University of Illinois at Urbana-Champaign. He received an MBA degree from National Sun Yat-sen University, Taiwan, Republic of China, and an M.A. and a Ph.D. in information systems from The Wharton School, University of Pennsylvania. Dr. Liang's research interests include decision support systems, expert systems, model management systems, and implementation issues of information systems.

Christopher V. Jones is Assistant Professor in Decision Sciences, The Wharton School, University of Pennsylvania. He received an M.A. and a Ph.D. from Cornell University. Dr. Jones's research interests include computer graphics, decision support systems, and computer simulation.