Association for Information Systems
AIS Electronic Library (AISeL)

ICIS 1980 Proceedings

International Conference on Information Systems (ICIS)

1980

MODEL MANAGEMENT SYSTEMS: AN APPROACH TO DECISION SUPPORT IN COMPLEX ORGANIZATIONS

Joyce J. Elam
University of Pennsylvania

John C. Henderson
University of Pennsylvania

Louis W. Miller
University of Pennsylvania


• N E W D O C •

MODEL MANAGEMENT SYSTEMS: AN APPROACH TO DECISION SUPPORT IN COMPLEX ORGANIZATIONS

JOYCE J. ELAM

JOHN C. HENDERSON

LOUIS W. MILLER

The Wharton School
University of Pennsylvania

This research was supported in part by ONR Contract No. N00014-75-0440.

1. INTRODUCTION

Recent years have seen an increased interest in developing interactive computer-based systems for supporting decisions that must be made in complex environments. Many of these systems are designed and built for decisions that relate to a specific problem (1,10) -- portfolio management, manpower planning, etc. Each of these systems center around a single model that a decision maker can use to explore various problem characteristics and solutions. The model, the user interface, and the model solution process are tightly coupled into a self-contained system. As a result, such systems lack flexibility and are difficult to adapt when there are changes in the problems they are designed to deal with. Modifications due to changes in the environment or because of learning on the part of decision makers may introduce the need to incorporate new policies, goals, or data into the analytic framework of the decision support system. This paper discusses an extension of the decision support system concept that we term "Model Management Systems" (MMS). These systems support decisions relating to a variety of problems that arise in a complex decision making environement.

In particular, the major objectives of a MMS are:

1. to facilitate the structuring of a decision so that analytical tools, possibly several in combination, can be used in generating possible solutions,

2. to facilitate the use of the analytical tools that have been brought together through a structuring process.

Thus, rather than being a predefined decision aid, a MMS can be viewed as a system that dynamically constructs a decision aid in response to a particular problem. This is accomplished by drawing on a knowledge base of models that reflects the technical expertise of a management scientist and the organizational experience with the activities involved in a given decision making environment.

This knowledge can be diffused throughout the decision making environment and adapted as necessary to support a decision maker in structuring as well as analyzing a problem. Knowledge representation, diffusion of knowledge, and adaptation of this knowledge in solving problems are basic characteristics of a MMS.

The remainder of this paper will discuss the MMS concept in depth. Organizational factors that have created a need for a MMS are discussed in the next section. Section 3 discusses the user roles involved with the MMS, and Sections 4, 5, and 6 build upon our experiences with prototype systems to discuss a structure of each MMS component. Section 7 presents our conclusions.

2. MODEL MANAGEMENT: WHY IS IT NEEDED?

A common characteristic of all decision makers is the use of a "model" as a basis to gather data, analyze this data, and eventually make a choice. These models may be intuitive or externalized, i.e., formulated in some symbolic manner. Even those models that have been externalized may not be in a form that

allows the utilization of computer technology to aid in processing. Nevertheless, a primary task of all decision makers involves the building and/or storing and/or recalling and/or executing of models.

Issues concerning the support of individuals in their roles as model builders and users is the focal point of Decision Support Systems (DSS). Keen and Scott Morton (10) define DSS as a system, normally computer based, which supports decision makers who are dealing with semi-structured problems (13). The focus on semi-structured problems is crucial. In effect, it requires the role of the DSS to be one of enhancing a process rather than providing a "product" or an answer. The goal is to create a more effective decision maker by facilitating his or her ability to search, create alternative solutions, and evaluate these solutions through the use of models. To achieve this goal, DSS are often designed to model the specific decision or problem under consideration, to exploit unique aspects of the decision process, and even to match the style of the manager. As a result, there is little capability to use the decision support system for other applications, even if these applications are closely related.

Most of the recent technological developments in DSS have been directed toward providing software and hardware that reduce the cost and time necessary to build and implement problem specific decision aids. While the benefits from problem specific aids have been demonstrated (1, 10), we contend that, in may situations, significant benefits can be realized by developing more generalized decision aiding systems that access, utilize, adapt, and integrate models.

For example, commercially available interactive financial planning systems allow non-technical managers throughout the corporation to build specific decision aids for analyzing their problems. The decentralization of modeling activities made possible by these systems has several benefits: expertise that exists in various functional areas can be utilized directly in the building, analysis, and interpretation of models; managers gain insights into their particular planning problems and thus, it can be argued, make better decisions.

While decentralization has its advantages, it creates difficulties when one attempts to develop an overall corporate model, particularly when one wishes to use the knowledge represented in the corporate model to analyze problems that cut across organizational boundaries. The decentralization of model building, ownership, and use creates a potential for inconsistent definitions and representations leading to circumstances wherein an appropriate model exists but is not recognized, or an existing model is inappropriately applied.

The development of a system for managing the model resource is not a simple task; but the costs of not managing this resource are real and are becoming more significant. Perhaps the most obvious cost is "reinventing wheels." A more significant cost is that the knowledge gained in developing a model is often not available to others in the organization. The result can be a decision not to use a model because there is no memory of organizational modeling activities and/or because no mechanism to diffuse and adapt such knowledge exists.

Even when one has knowledge of separate models that can be integrated to form a particular decision aiding system, the integration of models often requires tedious, time consuming manual interfacing. There is often little or no concrete basis for estimating development costs, assigning priorities to subtasks, or selecting personnel (perhaps from another part of the organization) for a modeling activity. The inability to resolve such issues may result in an inflated perception of costs and development time, leading to a decision not to develop a model.

All this argues strongly for viewing models as a corporate resource that needs to be effectively managed. The MMS provides a tool to aid this management task.

3. USER'S ROLES IN MMS

A MMS supports the user community in its attempts to interact with the computer for problem solving. But, who is the user community? Rather than focus on individuals or their specific organizational titles or functional responsiblities, we will focus on organizational roles often identified with the design and implementation of a DSS. Alter (1) identified 5 key roles associated with a DSS, the user, decision maker, intermediary, maintainer, and feeder. Adapting his classification, we identify 4 key roles associated with the MMS: decision maker, model user, model maker, and model implementor.

The decision maker is the client for whom the particular DSS application is

being designed. It is his or her effectiveness that the DSS is ultimately supposed to serve. As such, the world view of the individual(s) in this role will have a primary impact on problem definition and eventual success of the DSS.

The model user interacts directly with a dynamically constructed decision aid, running cases and interpreting outputs for the decision maker. The model user also requests the model maker to revise the decision aid as necessary.

The model maker uses knowledge of problem solving technology, the application of this technology to problems within the organization, and the particular decision context to conceptualize an appropriate logical structure for the problem. Based on this conceptualization, the model maker oversees the creation of a decision aid. If this conceptualization requires a model that does not currently exist in the knowledge base, the model maker calls upon the model implementor to create an appropriate model, which is then added to the knowledge base.

The model implementator is responsible for translating the specifications supplied by a model maker into appropriate computer programs and for integrating these programs into the MMS.

In some environments, one can expect a single individual to fill more than one role. In other environments, a single role will be filled by several individuals. In any event, a person in one role will invaribly be concerned with some aspects of other roles. The MMS provides mechanisms to support each role as well as interfaces between roles.

The major activities supported by the MMS are shown in Figure 1. Also depicted in Figure 1 are the outputs from the activities and the roles associated with the activities. Figure 2 illustrates the five basic components of a MMS: knowledge base, interrogator, builder, analyzer, and executor. The remainder of this paper discusses how each of these components are structures in order to support the activities outlined in Figure 1.

4. KNOWLEDGE BASE

A major objective of a MMS is to facilitate the structuring of a decision in a particular problem domain so that analytical tools can be used in generating possible solutions. The MMS supports this process by accessing knowledge on general problem structuring techniques as well as knowledge specific to the problem domain. In addition, when the MMS acquires new knowledge about the problem domain from its interaction with users, the MMS captures this knowledge so that is can be accessed at a later time. Thus, a key component of the MMS is a knowledge base. In this section, we discuss issues of representation of this knowledge base.

The MMS must represent the technical problem structuring knowledge that one would expect a management scientist to have (traditional view of the analyst). This knowledge involves information on mathematical model types (linear programming models, network models, deterministic simulation models, etc.), on the parameters that define one model type and distinguish it from other model types (constraints, decision variables, objectives, etc.), and the structural relationships between parameters. The MMS must also have an understanding of the basic activities involved in the problem domain (allocation, scheduling, production, etc.) and the way in which these activities interrelate (traditional view of the user). The MMS must also have knowledge of an appropriate vocabulary that can be easily incorporated into the user interface. Finally, the MMS must have knowledge about models that have been developed for specific applications (model instances) in order to support the execution and analysis of these models. The knowledge base, thus, contains four types of information: technical, application, language, and model.

More importantly, the knowledge base contains information on how all of this information is related, e.g., how is a user-supplied description of a problem related to basic activities that are known to exist in the problem domain and how can the activities be structured into an appropriate model to which some algorithm can be applied. It is this information that allows the MMS to bridge the gap between user and analyst.

A major design issue in developing a MMS is how to represent the knowledge that is required. Existing knowledge-based systems use either a predicate calculus approach or a graphical approach in representing knowledge. In choosing between these approaches, we considered the fact that the knowledge to be used by the MMS is highly interconnected and forms a network of concepts, facts, and perceptions. Each concept must be represented as a unit of knowledge that can be independently accessed and manipulated. Relationships that exist

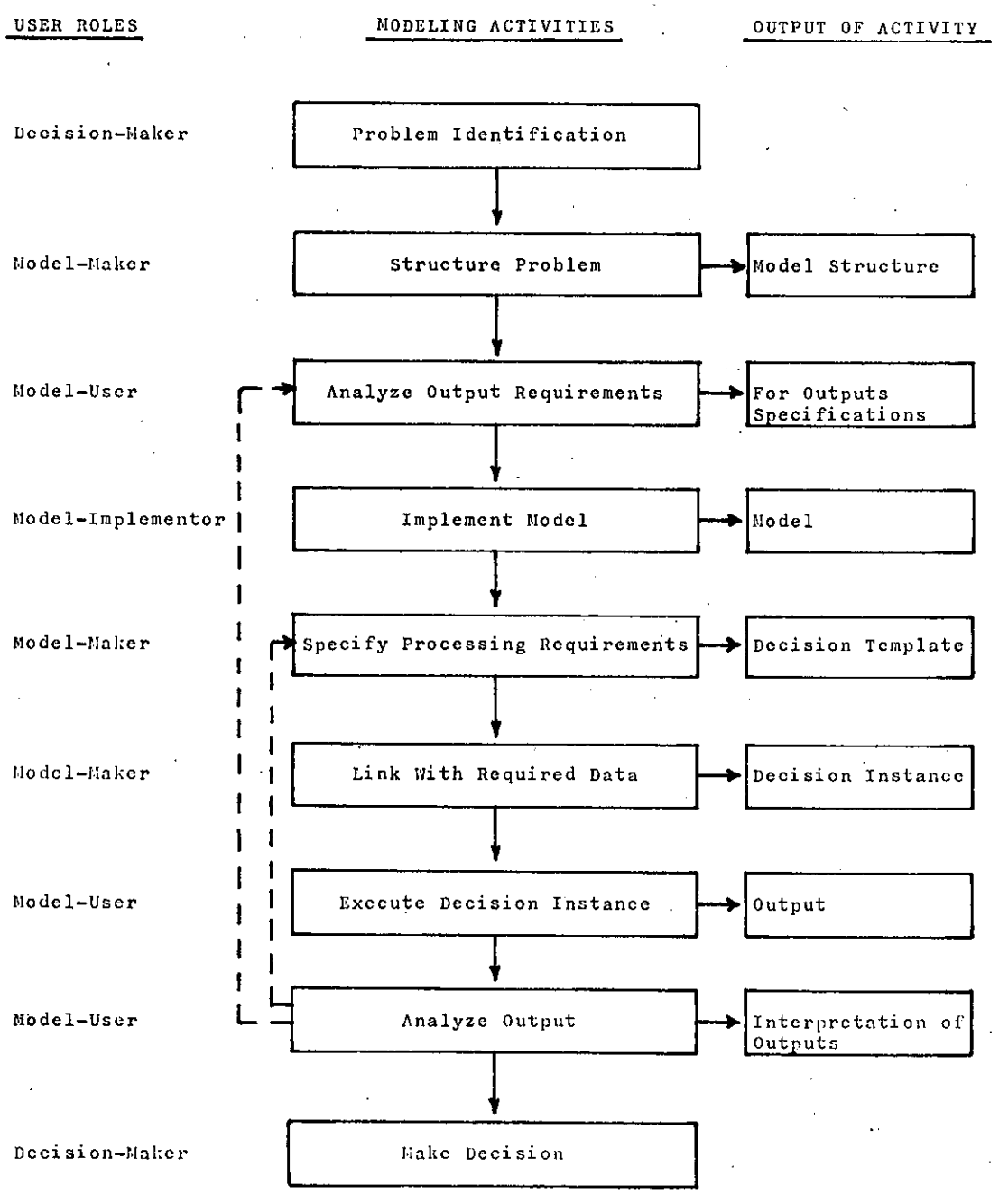

Figure 1
Modeling Activities

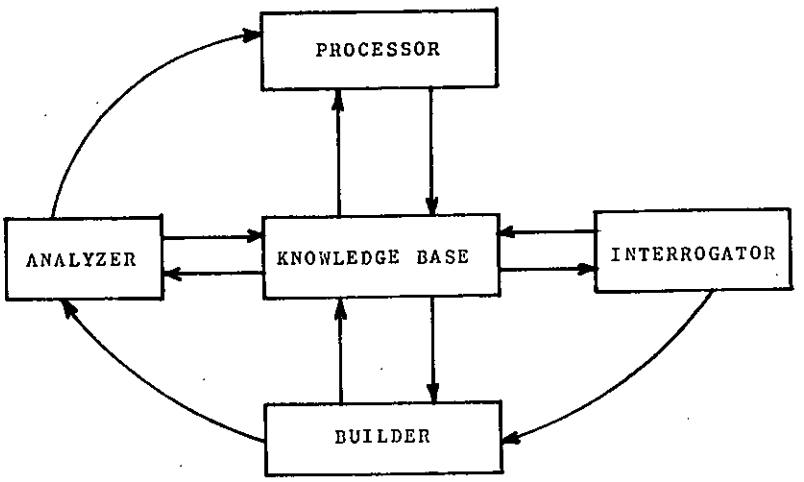

Figure 2

Model Management System

between these concepts can be used to define facts and perceptions and can provide the interrogation process with the "meaning" necessary to support problem structuring.

The predicate calculus approach has been successfully used in MYCIN (16), a knowledge-based consulting program for the diagnosis and therapy for infectious diseases. The primary source of domain-specific knowledge in this system is a set of approximately 200 production rules, each of the form: IF premise THEN action, where the premise is a Boolean combination of predicate functions.

Given below is an example of a rule from the MYCIN knowledge base:

IF

1) the infection is primary-bacteremia, and

2) the site of the culture is one of the sterilesites, and

3) the suspected portal of entry of the organism is the gastrointestinal tract

THEN there is suggestive evidence (.7) that the identity of the organism is bacteroides.

A production rule is an independent chunk of knowledge that is operationalized as a modular piece of code. The production rule methodology provides a simple and uniform way to represent facts. It is highly modular, allowing facts to be added and deleted easily. It also offers powerful mechanisms for judgmental reasoning of the form "A suggests B" or "A and B tend to rule out C."

The major drawback to using production rules and the predicate calculus approach in general as a representation is the difficulty in expressing complex concepts and describing the way in which these concepts are related. For example, it would be awkward or even impossible to use production rules to generate answers to questions such as: What is production? How is distribution different from production? Is distribution a type of resource allocation problem?

Graphical representations, on the other hand, allow one to answer such questions in a relatively straightforward manner. Graphical representations, the most widely known form being a semantic net, contain nodes and arcs, where the arcs represent relationships which can be used to answer questions like that posed above. For example, a common type of relationship is labeled "is a" and is used to convey such knowledge as a distribution problem is a type of resource allocation problem. Combining an "is a" relationship with another type of relationship labeled "has a part of" allows the semantic net to define an individual concept in detail. For example, shipping could be defined as a type of process that has an initial state, an ending state, and involves the movement of some type of good between states. These relationships allow knowledge to be "chunked" into groups of descriptions about a concept rather than

"chunked" into independent rules. As will be shown when we discuss the interrogation process, organizing knowledge in this manner is more appropriate for the MMS. These relationships also provide the basis for the development of rules that can be used to make inferences, much in the same way as production rules are used.

The major limitation of a semantic net is the inability to represent the wide range of conditions that can easily be represented by production rules. For example, the condition -- if the initial state and the ending state of a process are distinct and the process involves the movement of goods from one state to another, then the process is probably shipping -- cannot be represented in semantic nets.

An expanded graphical representation based on a semantic net that allows a condition of the type described above to be represented is the Structures Inheritance Network (SI-Net)(2). The SI-Net representation brings together the advantages of graphical and production rule representations. For these reasons, the SI-Net representation was chosen for the MMS knowledge base.

As mentioned above, a SI-Net is a graphical language composed of nodes and links for describing concepts and the interrelationships between these concepts. A "concept" is defined as a set of functional roles tied together with an explicit structuring relationship. Two basic types of relationships are involved in defining a concept "is a part of," which is represented by a DATTR link and describes the functional roles in a concept, and "are structured as," which is represented by a STRUCTURE link and describes how these roles are put together. It is through the structure link that one defines IF/THEN rules. Each role is described by 1) a role name, 2) a value restriction that represents the types of things that can fill this role (referred to as a role filler), 3) a number which represents how many role fillers there are, and 4) a modality value. Modality represents whether the role must have a filler supplied externally (necessary), or a role filler is not required (optional), or the role filler is to be derived from a processing routine or from the structural relationship defined for the concept (derived). Figure 3 illustrates the basic SI-Net notation for the concept "shipping point."

A SI-Net also represents relationships between concepts through a fixed set of link types. These relationships include such things as "is analogous to," "is a subconcept of," "is the same as except," "is an individual of," etc. These links allow a concept to inherit all properties as necessary. Each link has a "meaning" that provides the basis for a fixed and well-defined interrogation process.

The MMS knowledge base can be thought of as four distinct, but coupled SI-Nets: the technical net, the application net, the language net, and the model net. Figure 4 illustrates a small portion of a technical net and an application net. In the technical net, the concept "network" represents technical information about a type of mathematical programming model. It is defined in terms of the roles "nodes," "arcs," and an "objective function." Note that the value assigned to objective function is derived; this represents the fact that the role filler for objective function must be derived from some processing routine (e.g., a network optimization routine). The concept "network" also has a structural condition which specifies among other things that the network model must be connected. The concept "arc" is further defined by roles such as "from node," "to node," "cost," "flow," etc. Like the technical net, the application net also consists of concepts and structural conditions. These concepts are linked together in a hierarchy of less-to-more abstract concepts. The less abstract concepts relate to fundamental activities, while the more abstract concepts focus on problem systems. In Figure 4, the activities of storing the shipping are represented. Structural conditions are used to cluster related activities into a problem system. For example, a distribution system involves both shipping and storing, where the shipment of goods from locations must also involve the storing of the goods at some of the same locations.

The language net directly confronts the particular jargon, terminology, etc. associated with an organizational environment. The language net provides the capability for translating between a user's description of a concept and the system-defined label associated with the same concept. Thus, a manufacturer speaks of factories, products, and warehouse, while the military communicates in terms of depots, spares and loses. They both may be referring to identical problem systems and technical structures. The language net allows us to communicate with the user in a language that is both familiar and more precise.

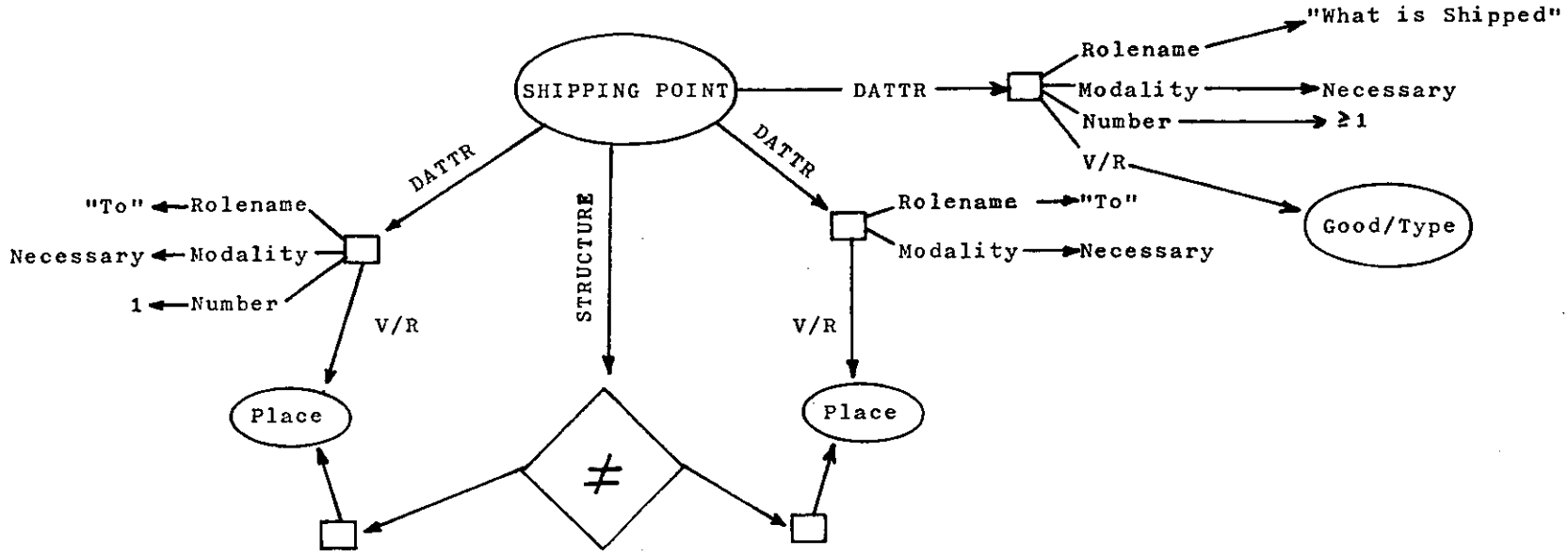

Figure 3

Simplified SI-NET Representation of "Shipping Point"

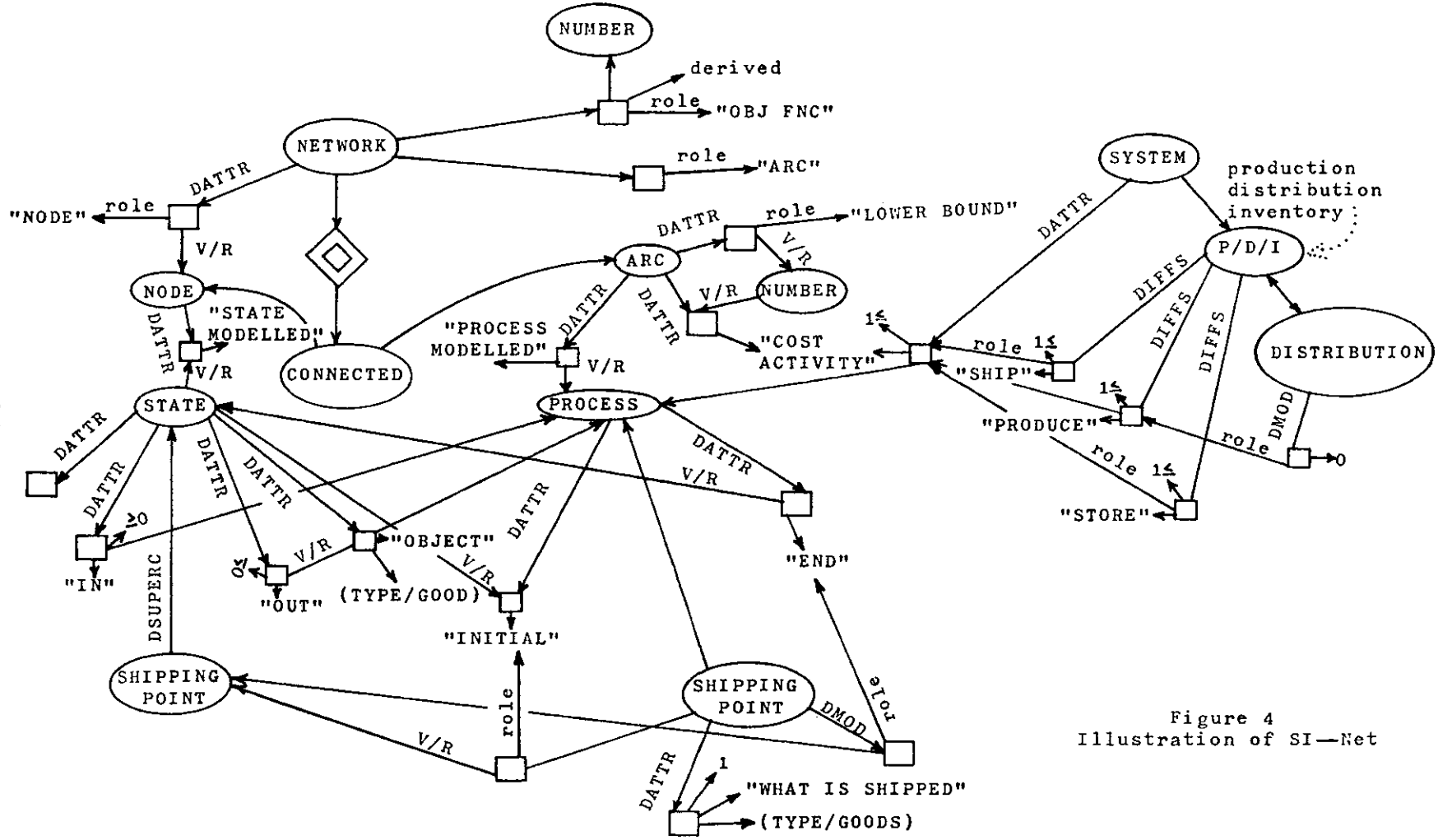

Figure 4
Illustration of SI—Net

5. THE INTERROGATOR

The knowledge base is made available to the organization via the interrogator component. This component functions somewhat like a data retrieval system but has enhanced capabilities due to the SI-Net. The basic objectives for this component of the MMS are fourfold:

1) to recall actual model instances

2) to support the problem definition and model conceptualization process

3) to identify necessary modifications for existing models

4) to initiate the actual process of model building

The attainment of these objectives is realized through an interactive stimuli/response sequence with a user. The system attempts to identify basic concepts, verify these concepts and infer potential logical structures that could be used to model the problem. Further, user generated lables for concepts are captured and used to dynamically adapt the semantics of the interaction so as to become increasingly "friendly."

For example, the interaction may begin with a user responding to a general question such as "What are the important activities/actions associated with your problem?" The response is free format, with the system processing the response to identify candidate concepts, generating questions to verify the interpretation of concepts, and using the SI-Net to infer possible problem systems. Verifying questions are generated by processing structural conditions to determine an appropriate set of related concepts that can be fed back to a user as a verification mechanism. The system tracks its verification steps so as to eliminate redundant questions. Further, if all concepts involved in a structural condition are verified, the current concept having that stuctural condition are verified, the current concept having that structural condition is automatically verified. The set of verified concepts provide the basis to infer more abstract concepts; for example, activities are used to infer problem systems. The selection of a particular problem system is made when the structural condition associated with that problem system has been investigated and shown to hold. For example, suppose that the concept of "ship" has been verified by the user as being relevant concept for his or her problem.

The inference process begins (Figure 4) by initially moving vertically in the net using a super concept link to the more abstract concept "process". The role fillers for "process" can be investigated to further verify the appropriateness of the concept "ship." "Process," in itself, is a value restriction for several problem systems. Each of these problem systems must be investigated to determine their relevance for both shipping and the user's problem. One possible problem system is production/distribution/inventory. The concept "ship" is a role filler in this problem system. At this point, the MMS must investigate other role fillers (represented through the DIFF links in Figure 4) to verify the appropriateness of production/distribution/ inventory. The user may indicate that "store" is an appropriate concept but "produce" is not. This insight allows the MMS to infer a problem system of distribution.

One advantage of inititating the interaction via the application net is the ability to use labels for basic action concepts and their roles. For example, during the verifying process for the concept "ship," the user may indicate that the "places" (a basic concept) from which goods are shipped have labels of "warehouse," "plant," etc. This allows the system to adapt the interface dialogue to utilize terminology which is more concrete in the user's mind.

The labels generated are linked through the SI-Net structure to more general concepts. Thus, "warehouse" is associated with "shipping location." This permits the user to create an instance of his or her problem by entering examples of relationships between indentified concepts. For example, the user may be asked to generate shipping routes (i.e., matched to and from labels). If there exists shipping routes into and out of a common shipping location, a transshipment model can be inferred. At this point, the conceptual relationships defined in the technical net are used to verify the appropriateness of the transshipment model.

The problem structuring process is iterative. It may require returning to the application net to verify new concepts or to clarify inconsistencies in the use of concepts. As interaction process proceeds, it results in an evolving model definition.

The Interrogator contains primitives that are the basis of commands allowing users to recall, store, compare, and contrast concepts. The store and recall

procedures allow the SI-Net to be stored, maintained, and reconstructed in ways compatible with current database management systems. The compare primitive operates in two modes: hypothesizing and discovery. In the hypothesizing mode, the "comparison" is between the MMS's understanding of concepts (i.e., model structures, activities, etc.) and the user's perceptions of the same concepts. In the discovery mode, the "comparision" is between two concepts that are already represented in the knowledge base. Hypothesizing supports problem structuring. The problem structuring process is initiated by the MMS with questions such as "what are the actions associated with your problem?" Discovery permits a user to determine if the system's understanding of a concept is the same as his own. The discovery mode is initiated by user commands such as "recall an activity like production" or "compare model 1 with model 2." Finally, constrast primitives support the use of commands such as "find a counter example" or "negate assumptions" or "contrast model 1 with model 2."

To this point, the MMS has been operating interactively with a user to define a problem system. In the example discussed above, the interaction began by verifying with the user the concept "ship." The user began with a problem definition centered on shipping, expanded that definition to include storing and exclude production. The MMS reasoned that the problem system being defined could be distribution. The relationships and the structural conditions in the technical net were used to investigate and verify the appropriateness of a transshipment model for this distribution system. At this point, the MMS can retrieve actual applications of transshipment models applied to distribution problems. These application examples not only help to verify problem system definitions but also can identify further information necessary to implement a decision aid. For example, the examination of a transshipment model may indicate the need to obtain a forecasting model for estimating the supply of goods to be shipped, the need to access an external database for freight costs, and the need to access an internal database for plant capacities. Further examination of these applications may clarify the need for certain reports. After these application examples have been examined, the user may wish to cycle back through the interrogation process or may wish to proceed to the building, execution, and analysis phase. The requirements to execute the building and analysis components are discussed in the next section.

6. BUILDING, EXECUTION, AND ANALYSIS

Interrogation of the knowledge base results in the model maker having identified a potentially useful model or set of models that will comprise the decision aid. To create and use the decision aid, however, involves the activities of building, execution, and analysis. Building is a process that resolves issues relating to compatibility between models, control of model execution, and communication of data between models and between users and models. The result of building is a decision template, which is a file containing human and machine readable specifications to instruct the executor (operating system) component of the MMS in preparing a computer program to implement the decision aid.

The execution phase, based on the specification in the decision template, includes physical accessing of computer code from libraries, linking together models (in the usual sense of preparing a load module), executing the program, and possibly saving the outputs for further processing. Somewhere between building and execution, a top level control procedure for the decision aid must be generated. We prefer to make this part of execution rather than building in order to keep decision templates simple.

Analysis means presentation of results to the user and possibly includes processing outputs of models by statistical and graphical procedures. For some applications, software support for analysis is an integral part of models. But frequently, it is advantageous to perform analysis separately from executing models. Therefore, the MMS contains an analysis subsystem.

By now there are many software systems that support building, execution, and analysis and supply some services similar to those of the MMS. The several commercially available financial planning systems comprise a class of typical systems. The MMS is more general in that it is not meant to be restricted to a particular problem domain, and models are implemented in standard programming languages (such as FORTRAN and PASCAL). Moreover, the special modeling systems frequently are intended to encourage a single individual to take on the roles of decision maker, model user, model maker, and model implementor described above, and tend to support the concept of the "disposable model." With the MMS, however, we expect many models to be large and complex, so that the services of professional implementors are required,

and we expect that model makers will do very little that resembles computer programming.

6.1 BUILDING

The purpose of building is to organize enough information so that the models chosen in the interrogation process can be run. In order to do this, four kinds of issues have to be dealt with. They are: 1) assuring the compatibility of models, 2) inter-model communication, 3) sequencing and control of models' executions, and 4) obtaining data from the user or external sources.

To help resolve these issues, the MMS has access to a collection of information beyond that already discussed as components of the knowledge base. We call this information "model documentation," and it exists specifically to support building. Documentation for a model is written in a special "documentation language" by the implementor as part of the model coding process. Documentation language statements are interpreted and used by the MMS, and their content can be displayed in order to support the model maker role. Documentation language statements mainly describe the inputs and outputs of a model, but they deal with other topics as well (such as providing names for accessing program libraries).

Inter-model communication is a problem because the outputs of some models are inputs to others, and part of the building activity is to identify the sources of each model's inputs as coming from another model, from an outside data base, or from the user. In the example of a decision aid to be composed of a forecasting model and a transshipment algorithm, the forecasting model requires inputs regarding exogenous variables from a data base, and control parameters from the user. The outputs of the forecasting model will be estimates of demands at several locations, which in turn, are inputs to the transshipment model. In order for the two models to be compatible, the transshipment model requires the forecasted demands, and it must be assured that the output of the forecasting model is in a format that the transshipment model can understand. Reorganizing and reformatting the output of one model to be consistent with the input requirements of another model is done by incorporating programs that appear to the MMS's executor as though they are simply additional "models" to be included in the decision aid.

Organizing data flow between models is supported by the documentation language statements for the models. If the documentation for the transshipment model specifies that the model needs a vector of DESTINATION REQUIREMENTS, the MMS will try to find another model, among those identified by the interrogation process, that has an output with a similar description. If a match is found, the model maker has an opportunity to verify that the linking of these particular items is appropriate. The model maker has a good idea of what the correct linkage should be because of information gained from the interrogation process. The model maker also has access to the documentation language statements, which can include descriptive material for the model maker's benefit. If the MMS is unable to match the input of a model to another model's output, the model maker can assist.

Sequencing and control of models means deciding in what order the models should be called upon and arranging for iterations of model executions, should that be required. Requirements for sequencing models are usually derived from considerations of inter-model communication. (In the example of the forecasting and transshipment models, it is clear that the forecasting model has to be run first.) Iteration means repeating the execution of models or groups of models in order to achieve some kind of convergence or to generate results for sensitivity analysis. Specifying requirements for iteration is the closest that the model maker comes to performing an activity that resembles computer programming.

In our example, the two models may not always be run together. The user may wish to experiment with the forecasting model alone before using it to obtain the inputs to the transshipment model using a single forecast. The MMS has to allow these options without placing a burden on the user.

6.2 USER'S DATA

In addition to models communicating with each other, models frequently require information from users. Inputs from users can include parameters representing problem data (e.g., choices of transshipment points allowed or disallowed), choices among options available within models (e.g., designation of criteria used by the forecasting model), or names of files that will serve as sources of input data (e.g., file of source-destination shipping charges). Managing users' input data is an important part of the model management concept in that programs run under the MMS should not interact directly with the user through

normal read statements. Instead, the model implementor describes in documentation language statements what is wanted from the user. The description includes mode and dimensioning information, text for prompts, help texts, and error conditions that should be checked for. Then the model management system will prompt the user and make the information available to the model through "get" procedures called from the model program.

There are several reasons for taking this indirect approach to handling users. First is that programming high quality user interactions is difficult, and the MMS can provide better and more uniform interactions than most programmers would be willing to implement. Another benefit is that models may require a great deal of data from the user, but most of the data would not be changed over a series of model executions. The values of such data items can be given by the model maker to become part of the decision template. The remaining data inputs are defered until the model is executed. The third benefit is that by having users' data pass through the MMS, the data can be saved (and even annotated) as a form of documentation.

6.3 CHECKING

Checking the consistency of the linkages in a model implied by a decision template is a valuable service provided by the MMS. The checking procedure verifies that the models are being executed in a feasible sequence and that every model's inputs can be found and are in the proper format. Reports produced by the checking process can be saved as additional documentation of the decision aid, and can be examined for further verification that the decision aid makes sense in terms of the problem to be solved. Checking is a dry run for executing the model, and it employs procedures that are part of the MMS's facility for generating the decision aid's controlling routine.

6.4 SAVING AND MODIFYING DECISION TEMPLATES

Once the effort has been expended to produce a decision template, the template can be saved in the knowledge base. Facilities are provided to make modifications to old decision templates so that models specified by them can be removed or added and user's data can be changed.

6.5 ANALYSIS OF MODEL OUTPUTS

As mentioned earlier, in many applications it is advantageous to carry out analysis in a process separate from running models. As an example, we have applied many model management concepts to micro-analytic simulations (11) that simulate the interaction of events and policies on samples of individuals. The models take a disaggregated view and produce voluminous outputs, giving a record of attribute values for each of the simulated individuals. By having aggregation and analysis done on the results of models rather than by the models themselves, users can explore results in many different ways without having to rerun models. Further, having a separate analysis subsystem makes it possible to employ analytic procedures to make comparisons across the results of several model runs. The analysis system used in this work resembles an interactive, extensible statistical package, and its design is consistent with many of the general model management concepts, including a library of modules that can be added to, a template to describe what is to be done to what data, and user interactions as described previously.

Many of the ideas in this section have been operationalized in WHIMS (9), which was developed in order to provide flexibility and modularity in working with micro-analytic simulation models. Although WHIMS is weak in supporting the interrogation process and it is quite restricted in the kinds of model structures that it supports, the building, execution, and analysis subsystems are highly developed.

REFERENCES

1. Alter, Steven L. Decision Support Systems: Current Practices and Continuing Challenges, Addison Wesley, (1980).

2. Bonczek, R. C., Holsapple, W., amd Whinston, A. Computer based support of organizational decision making. Decision Sciences, (April 1979).

3. Brachman, R.J. A structural paradigm for representing knowledge. Report No. 3605, Bolt, Beranek, and Newman, Inc. (1978)

4. Donovan, John. Database system approach to management decision support, TODS, 1,4, (1979), pp. 344-369.

5. Dill, L. and Hilton, W. Reitman. The New Managers, Prentice Hall, N.J., (1962).

6. Elam, J. Model management systems: a framework for development. Working Paper 79 02 04, Department of Decision Sciences, University of Pennsylvania, (1979).

7. Finuller (Ed). Associative Networks: Representation and Use of Knowledge by Computers, Academic Press, (1979).

8. Gorry, G.A., and Scott Morton, M.S., A framework for management information systems. Sloan Management Review, 13, 1, (Fall 1971), pp. 55-70

9. Katz, N. and Miller, L. An interactive modeling system. Working Paper 77 09 02, Department of Decision Sciences, University of Pennsylvania, (1977).

10. Keen, P. and Scott Morton, M. Decision Support Systems: An Organizational Perspective, Addison Wesley, (1978).

11. Orcutt, G.H., Greenberger, M.H., Korbel, J., and Riolin, A.M. Microanalysis of Socialeconomic Systems: A Simulation Study, Harper and Brothers, (1961).

12. Quillian, Ross, Semantic memory. Semantic Information Processing, Marvin Minsky, (Ed.) MIT Press, (1968).

13. Simon, H.A., The Shape of Automation for Men and Management, Harper & Row, (1965).

14. Stohr, E.A. and Tanniru, M.R. A data base for operation research models. Policy Analysis and Information Systems, 4, 2, (December 1980).

15. Thompson, J. Organizations in Action, McGraw Hill Book Company, (1967).

16. Davis, R., Buchanan, B., Shortliffe, E. Production rules as a representation for a knowledge-based consultation program. Artificial Intelligence, 8, (1977).