Model Management Systems: Proposed Model Representations and Future Designs

Lynda M. Applegate
University of Arizona

Gary Klein
Southern Methodist University

Benn R. Konsynski
University of Arizona

Jay F. Nunamaker
University of Arizona

Model Management Systems: Proposed Model Representations and Future Designs

by

Lynda M. Applegate*[*]
Gary Klein[**]
Benn R. Konsynski[*]
Jay F. Nunamaker[*]

[*] Department of Management Information Systems
College of Business and Public Administration
University of Arizona
Tucson, Arizona 85721

[**] Department of Management Information Sciences
Edwin L. Cox School of Business
Southern Methodist University
Dallas, Texas 75275

ABSTRACT

The availability of microcomputers, modeling langauges and general purpose spreadsheets has resulted in an increase in the use of models for decision making within organizatons. Decision makers with microcomputers on their desks and spreadsheet and modeling software can create models rapidly. Problems with model redundancy, consistency, integrity and security have prompted an increased interest in the design of model management systems (MMS).

Several model management designs have been discussed in the literature. Different model representation techniques have been proposed. These include formal logic, semantic inheritance networks, frames, and relational representations. The approaches to model management are evaluated in respect to their model manipulation and model storage functions. A framework for the design of MMS is proposed based on the system design objectives and the system domain complexity. Advantages and disadvantages of each model representation method are identified. Application domains for the classifications are proposed which focus on the strengths and weaknesses of the model representation for supporting model storage and model manipulation functions. An example of the design of a MMS using the classification is presented.

Introduction

The use of models within organizations has increased dramatically over the past few years (Ulvila and Brown, 1984). The availability of microcomputers, modeling languages (e.g., IFPS), and general purpose spreadsheets (e.g., Supercalc3, Lotus 1-2-3) has provided decision makers with the necessary development tools for rapid model implementation and design. These models, developed by different decision makers, can lead to different and often conflicting results. Problems of model redundancy, inconsistency, integrity and security have been noted. These problems are similar to the data management problems that prompted the design of database management systems.

Model management systems (MMS) have been proposed as a possible solution to the need for support of the modeling activities of organizations (Will, 1975). An MMS is a software system which provides for the creation, manipulation and access of models. The ability to store and manipulate models is a critical function in the design and implementation of an MMS. Model storage functions include model representation, model abstraction, physical model storage and logical model storage. Model manipulation functions include model instantiation, model selection and model synthesis. The MMS accesses the centralized database of the organization and external databases to obtain the data necessary to solve a given problem. The capability for interactive input and correction is also provided. Two objectives for model management have been identified: (1) to expand and enhance decision support system (DSS) capabilities by improving the modeling component of the system; and (2) to enable organizations to centralize model management functions and insure the integrity, consistency, currency and security of model bases.

Significant progress has been made on MMS design for support of organizational modeling activities (Elam et.al., 1980; Bonczek, Holsapple and Whinston, 1981; Konsynski and Dolk, 1982; Blanning, 1982). A variety of knowledge representation schemes (e.g., formal logic, semantic inheritance networks and frames) from the artificial intelligence literature have been proposed to implement the model storage and model manipulation components of the system. A relational model representation scheme, using relational database concepts, has also been proposed.

The purpose of this paper is to review several model representation schemes that have been addressed in the literature. Advantages and disadvantages are discussed for each representation. A framework for classifying MMS based on the system design objective (DSS support or centralized model base management support) and the complexity of the system domain is proposed. Recommendations for the choice of a suitable model representation based on the class of MMS are presented.

Evolution of the Model Management Concept

Automation of the modeling process began with the first computer programs. Models were used to provide the algorithmic, problem-solving logic which formed the heart of most computer programs. These programs provided a single-model, deterministic perspective to the problem solution. This approach was effective for traditional data processing applications where the manipulation of data was the primary objective of the system and structured logic could be applied. It soon became apparent, however, that this approach was less effective for decision support applications where the manipulation of the model was the primary objective of the system and structured logic was not applicable.

The need to manage organizational models followed an evolutionary path similar to the data management concept. Three distinct stages can be identified in the evolution of the data and model management concepts. These are: (1) file management; (2) data base and model base management; and (3) generalized data base and model base management.

EVOLUTION OF THE DBMS CONCEPT

Data management began in the late 1950s and early 1960s with the advent of data file management systems. Organizational data, which were previously tightly coupled to a single application program, were stored in centralized data files that could be accessed by a number of application programs. Problems arose with this form of data management as improved computer technology resulted in the increasing use of the computer by more and more users attempting to access, update and create data. Problems of data redundancy, inconsistency, integrity and security soon became widespread. This prompted managers within organizations to demand centralized control of data resources.

The second stage in the evolution of the data management concept was marked by the introduction of database management systems (Codd, 1970; Database Task Group, 1971). These systems, popularized during the late 1960s and early 1970s, were designed to provide the centralized management and control of data required to meet increased organizational use. Early database management systems were static systems with little flexibility for responding to the changing needs of the organization. Managers soon realized that the time and programming effort required to restructure a database and reprogram application programs was unacceptable for providing timely and accurate data to organizations undergoing growth and change.

Generalized database management systems (GDMS) were developed to provide flexible access to organizational data stores while maintaining the centralized control afforded by the database management system (Minker, 1969; Angell and Randall, 1969). These systems allowed the manipulation of newly defined data and files using the existing application programs and systems. These systems also provided access to data by name instead of physical location, which helped to insulate the user and application program from the physical implementation of the system.

EVOLUTION OF THE MMS CONCEPT

The development of centralized subroutine libraries marked the beginning of the model management concept. These subroutine libraries were similar to the early data file management systems in that they allowed a model, which was previously hard-coded into a single application program, to be separated from the program and accessed by a number of application programs.

These libraries of subroutines were soon upgraded to "utility packages" to allow decision makers to access groups of related models. Examples of these utility packages are the Statistical Package for the Social Sciences—SPSSx—(SPSS Inc., 1983) and the Experimental Mathematical Programming package—XMP—(Marsten, 1981). These utility packages provided model management functions which were analagous to the early database management systems. Frequently used models could be centrally controlled for widespread access but the systems suffered from a static design which required considerable programming effort and time to revise if organizational modeling needs changed.

This approach to model management met organizational needs until the advent of the microcomputer and the rapid proliferation of modeling languages and spreadsheets. Prior to that time, the modeling activities of an organization were fairly static. The time and expense necessary to create new models prevented organizations from using computer models for all but the most complex decision problems. A 1974 survey by the National Science Foundation, reported by Bisschop and Meerhaus (1981), found that the average time needed to create a new mathematical model was approximately seventeen months. The average cost of these models was approximately $100,000. As a result, they found that models were rarely used for policy-making decisions.

Over the last few years this situation has changed dramatically (Ulvila and Brown, 1982). Decision makers with microcomputers on their desks and modeling languages and general spreadsheet packages on floppy disks can create models rapidly. These models, however, can produce conflicting results since they are often developed and used by a specific decision maker with a specific conceptualization of the problem. The problems of model redundancy, inconsistency, integrity and security mirror the data mangement problems which prompted the design of DBMS.

A second major information systems trend over the past few years has provided further impetus to the development of the model management concept. The increased use of decision support systems (DSS) to support organizational decision making has paralleled increased organizational modeling activities. These two developments are clearly related since the focus of a DSS is the modeling component of the system (Sprague and Carlson, 1982).

Design of early DSS was approached in a similar manner to the development of early computer models and computer programs. The data, dialogue and model components were tightly integrated, leading to a static system design that was costly and time-consuming to develop and difficult to change. The widespread use of DSS within organizations was hampered by this static system design. The concept of an MMS was proposed to enable the DSS designer to offer a wide range of models within the system and to allow for flexible access, update and change of the model base. It was hoped that the addition of an MMS to the basic DSS design would enhance the flexibility of the DSS and allow for the support of more complex decision problems.

Generalized model management systems have been proposed to provide for flexible and dynamic model management within organizations. The early forerunners of the generalized model management concept can be found in the research on generalized modeling techniques and database management systems. Examples of this research include the General Problem Solver—GPS, developed by Shaw, Newell, and Simon and described in detail by Ernst and Newell (1969); the Numerical Analysis Problem Solver—NAPSS (Rice and Rosen, 1966); and the Generalized Database Planning System—GPLAN (Nunamaker, Swenson and Whinston, 1973).

The GPS research began in 1957 with the dual intention of (1) developing a machine which would solve problems requiring human intelligence; and (2) developing a general theory of how humans solve problems. GPS was the first problem-solving system that separated its general problem-solving methods (models) from the knowledge specific to the problem to be solved. NAPSS, designed and implemented at Purdue University, is also a generalized problem-solving system in which the modeling knowledge is independent of the decision problem. A user, unskilled in numerical analysis, can solve relatively complex decision problems by describing the decision parameters. The system selects the models, performs the analyses, and gives diagnostics of possible difficulties and meaningless results. These two systems were forerunners of the use of a generalized model component to improve the flexibility and scope of DSS design.

The GPLAN system extended the generalized data management system concept to provide for the automatic setup of models from a database as instructed by the user through a special query language. The GPLAN system functioned to generalize the organizational database management system to provide some control over organizational modeling activities. This system is a forerunner of the use of generalized model management systems to provide centralized control and management of organizational models.

The design of an MMS developed to enhance the flexibility and scope of a DSS is similar to the design of an MMS developed to provide centralized control of organizational modeling activities. Both types of systems require the ability to store and manipulate models for access by users with a variety of decision problems. There is a major difference, however, in the design. An MMS that is developed as a component of a DSS requires a model representation which focuses on the relationship of the models to the decision process that the system is designed to support. An MMS that is developed for centralized control of organizational modeling activities requires a representation that focuses on allowing the user to access the model without knowledge of its physical limitation. In addition, the design must support the ability to store a widely diverse set of models unrelated to any specific decision problem.

Several designs have been proposed for implementing a generalized MMS (Elam et.al., 1980; Bonczek, Holsapple and Whinston, 1981; Konsynski and Dolk, 1982; Blanning, 1982) based on different model representation schemes. The model manipulation and model storage components for each model representation are reviewed in an attempt to develop a framework for selecting a MMS design that is best suited to the purpose of the system.

Review of Proposed Model Representation Schemes

The majority of MMS designs proposed in the literature make use of artificial intelligence concepts and techniques for the manipulation and storage of models in the system. The similarity between the problem-solving systems of artificial intelligence research and the decision-making process provides support for using similar design and implementation criteria and techniques. Model representations have been proposed which utilize predicate calculus, semantic networks and frames to implement an MMS. In addition, one representation has been proposed which utilizes a relational framework for implementation of an MMS.

FORMAL LOGIC MODEL REPRESENTATION

Formal logic, in the form of production rules and predicate calculus logic, is used in some form for every MMS that has been proposed. There are two main components of formal logic techniques. The first is the axiomatic structure of a system which specifies relations and implications within the system. The second component is the deductive structure of a system which specifies the rules of inference which can be applied if certain axioms are assumed to be true. The determination of the truth of a statement, a fundamental concept of formal logic, is accomplished by evaluating the syntax of an individual statement and the syntactic manipulation of formulae.

The most commonly used formal logic techniques for use in automated problem-solving systems (both AI systems and DSS/MMS) are predicate calculus and production rules. Predicate calculus is a formal language for expressing assertions, axioms and the rules of inference about a problem domain (Nilsson, 1980). The syntax of the predicate calculus language includes symbols for predicates, variables, constants, functions, operators and quantifiers. The rules of inference determine the operations that can be applied to given well-formed functions (wffs) and sets of wffs through a variety of techniques (especially resolution) to produce new derived wffs. These derived wffs are called theorems.

Production rules are condition-action pairs of the form IF [condition] THEN [action] ELSE [action]. These rules are stored in a rule base which forms the central core of a production system which also includes a context component and an interpreter. The context component functions as a short-term buffer and specifies the data for the particular problem at hand. The interpreter functions to control the flow of system logic.

Bonczek, Holsapple and Whinston (1981) have proposed a generalized DSS architecture in which the application-specific modeling knowledge is represented as predicate calculus axioms, clauses and wffs and is stored along with a variety of models in a model base called a "module pool." See Figure 1.

This representation uses predicate calculus logic in the form of wffs, clauses, axioms and rules of inference to implement the model storage component of the system. Resolution techniques and other state-space search methods are used for model manipulation.

The primary advantage of the formal logic representation for an MMS is that it employs a well-researched and powerful reasoning process, especially for model manipulation functions. In the artificial intelligence field, there have been many successful implementations of formal logic for problem solving systems and expert systems.

The primary disadvantage is that the use of formal logic without an organizing framework can result in an extremely large search space of rules for complex problems. This can make the resolution process cumbersome, slow and in many cases infeasible. A second problem with the use of predicate calculus formal logic for the knowledge and model representation is that the relationships between the concepts are lost. Because this representation stresses knowledge as independent facts, it is difficult to decompose problems through categorization

```
-----------------------------
| MODEL MANIPULATION COMPONENT|
|  -------------------------- |
|     Resolution Techniques   |
|State-Space Search Techniques|
-----------------------------
              |
-----------------------------
|  MODEL STORAGE COMPONENT    |
|  -------------------------- |
|  Predicate Calculus Wffs,   |
|  Clauses and Rules of Infer.|
-----------------------------
```

Figure 1

Formal Logic Model Representation

and classification (chunking) unless a separate knowledge representation is used along with the formal logic representation.

SEMANTIC INHERITANCE NETWORK MODEL REPRESENTATION

The semantic inheritance network knowledge representation concepts evolved from the literature on the psychological models of human associative memory (Quillian, 1968). Semantic inheritance networks are composed of nodes and links between those nodes. The nodes are the storage structures for the knowledge while the links represent the interrelationships of the knowledge.

There are four major types of nodes (Brachman, R.J., 1978). These are concept nodes (predicates for objects and actions), role description/role instance nodes (items which are related to the concepts), structural condition nodes (relations between concepts and role description/role instances), and structural reference nodes (access to the complex descriptions through decomposition).

There are two types of links which connect the concept node to the nodes representing its internal structure. "Dattr" links connect the concept nodes to the role description/role instance nodes and denote the attributes of a given object or action predicate. "Structure" links connect the concept nodes to a structural condition or structural reference node and denote how the specific attributes and concepts are tied together.

Elam et.al. (1980) have proposed an MMS utilizing semantic inheritance network knowledge representation concepts. In this MMS framework, the model is stored in a network which links together concept nodes with their associated structural and role description nodes in a hierarchy from the least abstract to the most abstract. Four networks are used to store the modeling knowledge.

The technical net represents the technical information about each model to be stored in the MMS. The model net contains the objective function values for the specific model derived from a network optimization routine. The language net contains user-defined labels of the specific concepts in the network, which allows the user to access the models using familiar terminology. The problem-specific information is stored in a separate semantic inheritance network, the application net.

The model selection function is accomplished through the "interrogator" component. This component functions like a date retrieval system to select, classify, modify and build models. Production rules and predicate calculus resolution principles are used to search the tree and select the appropriate model. The semantic inheritance network MMS framework is presented in Figure 2.

An advantage of the semantic inheritance network representation for an MMS is that the relationships among concepts are preserved, making it possible to categorize and classify knowledge to promote problem decomposition. The use of resolution and production rules in the model manipulation function provides a powerful inferencing structure for model selection and query processing.

```
-------------------------------
| MODEL MANIPULATION COMPONENT|
|         Interrogator        |
|  -------------------------- |
|Predicate Calculus Resolution|
|     State-Space Search      |
-------------------------------
               |
-------------------------------
|  MODEL STORAGE COMPONENT    |
|       Technical Net         |
|         Model Net           |
|        Language Net         |
|      Application Net        |
|  -------------------------- |
|  Semantic Inheritance Nets  |
-------------------------------
```

Figure 2

Semantic Inheritance Network Model Representation

The major limitation with this representation is that it is difficult to represent a wide range of conditions in a complex problem. The volume of information and the lack of direct support for multiple levels of logic are serious constraints on the applicability of this approach. However, with minor modifications, the semantic network approach has proven quite useful in the PLEXSYS system for the specification of a knowledge base that evolves during the system design process and supports a dynamic, frame-oriented system. (Konsynski, Kottemann, Nunamaker, Stott; 1984). This system uses the semantic network approach at the knowledge management level and uses a frame approach (to be discussed in the next section) for scripting and context structuring.

ABSTRACTIONS/FRAME MODEL REPRESENTATION

The most recently developed knowledge representation to be suggested for the implementation of an MMS is the frame representation. This knowledge representation technique was described by Minsky (1975). A frame is basically a data structure that includes declarative and procedural information in pre-defined internal relations. This information can be stored as predicate calculus well-formed functions, production rules or other knowledge representation types. The important thing about the frame representation is that the knowledge is classified according to the types of information which must be remembered for a given circumstance. This can include concept properties and structure (as in the semantic inheritance network) as well as related information and heuristics such as how to use the frame, what actions can take place inside the frame and which interactions are allowed.

As a data structure, the frame can be viewed as a set of slots which contain fixed information about the nature of the frame itself and variable information which can change depending on the specific problem attributes. Individual frames are linked together by an information retrieval network which allows for a more efficient retrieval and search process.

Konsynski and Dolk (1982) have proposed the concept of a model abstraction for model representation within an MMS. The model abstraction is a hybrid of several knowledge representation schemes, but most resembles the frame representation concept presented above. The outward structure of the abstraction is based on the programming language concept of a data abstraction and consists of data objects, procedures and assertions expressed in first-order predicate calculus.

The data objects section enumerates the data items and types comprising the structure (model) being described. The procedures section lists each procedure, the data object(s) it accesses, and the data object(s) it returns. The assertions section specifies information about the data objects and procedures and their various relationships. Data items, data types and procedures are assumed to be predicates while assertions are well-formed functions. Further work in this area by Klein, Konsynski and Beck (1984) has presented a goal programming formulation to store the model/problem attributes as a linear function that describes the relationship between attributes for previous successful solutions. The frame, therefore, includes information regarding the problem and model attributes of previous successful problem solutions. This information is used during the model selection process to provide additional support for the choice of a given model from the model base. The model abstraction/frame representation for a MMS is presented in Figure 3.

The model abstraction/frame representation for an MMS combines the positive features of the formal logic and semantic inheritance network knowledge representations for the implementation of an MMS. The frame serves to organize the specific models in the model base from a conceptual, structural and operational standpoint while also providing significant detail on the model/problem attributes. This functions to maintain a modular structure in the MMS that enhances the flexibility of the system and allows for more efficient solution of a wider range of complex problems. The use of goal programming to store a linear representation of the model/problem attributes also serves to simplify the model selection and model storage functions and, therefore, enhances the ability to represent more complicated decision problems.

The primary disadvantage of this representation for the design and implementation of an MMS is that the frames must be pre-defined in the context of the problems to which they are applied. This may be quite inconvenient in the absence of an ability to "learn" these structures. Another problem is that the frames offer constraints that may preclude alternative useful paths in the resolution of problems.

RELATIONAL MODEL REPRESENTATION

Blanning (1982) has proposed an MMS architecture which utilizes a relational model representation. The model is represented and stored as a relation which includes the relevant input and output criteria for the model. Tuples in a model relation differ from the tuples in a data relation in that they do not exist in stored form. Instead the model tuples are generated on demand, based upon the input and output attributes which are required to answer a given query.

The model relations are normalized and decomposed in a similar manner to the data relations in a relational database but utilize different types of dependency relations based on the anomalies that could arise in using the models to solve problems. The anomalies of interest in data management are primarily update anomalies—addition, deletion or changes to the specific tuple in the database. Since model tuples are not stored, these anomalies do not exist for an MMS. The primary anomalies for model management are called processing anomalies (Blanning, 1982b). Three types of processing anomalies have been identified. These are: (1) input anomalies, (2) search anomalies, and (3) output anomalies.

The model manipulation function is implemented using a model query language (MQL) which is similar to the SEQUEL (SQL) relational query language for a DBMS. The only operation in relational data management that is also useful for model management, however, is the selection operation (Blanning, 1983). The join and projection operations are not needed because the model tuples are not stored within the system. Instead of relational database joins, based on the relations among the data items, the models are "joined" based upon an ordering of their calculation processes. This occurs when the output from one model becomes the input to another model. The relational framework for an MMS is presented in Figure 4.

The ability to integrate the data storage component and model storage component of an MMS is significantly enhanced by using a relational MMS representation. This is a key advantage of this representation over the representations presented above. In addition, the power of the relational algebra query language has been well documented for database management systems and it is expected that it would provide an excellent model manipulation component for an MMS.

A disadvantage of this approach is that the relationships between the model and problem are difficult to fully describe. The tabular form of representation is not well-suited to problem description.

A summary of the major advantages and disadvantages of the four MMS model representations is presented in Table 1.

Framework for MMS Design

As discussed earlier in this paper, organizations are becoming increasingly aware of the need to manage models. This need has arisen as a result of (1) a desire to utilize a variety of models within general purpose DSS to support decision-making activities at all levels of the organization and for all classes of decision problems; and (2) the rapid proliferation of the use of models within

```
--------------------------------
| MODEL MANIPULATION COMPONENT|
  -------------------------
     Predicate Calculus
         Resolution
|  Goal Programming Selector  |
--------------------------------
               |
--------------------------------
|  MODEL STORAGE COMPONENT     |
   -------------------------
   Model Abstraction/Frame
   Predicate Calculus Wffs,
   Clauses,Axioms and Rules
   of Inference
   Goal Programming Attribute
|  Storage                     |
--------------------------------
```

Figure 3

Abstraction/Frame Model Representation

```
--------------------------------
| MODEL MANIPULATION COMPONENT|
  -------------------------
     Model Query Language
           (MQL)
--------------------------------
               |
--------------------------------
|  MODEL STORAGE COMPONENT     |
   -------------------------
|   Relational Model Base      |
--------------------------------
```

Figure 4

Relational Model Representation

Table 1

Advantages and Disadvantages
Proposed Model Representations for Model Management

FRAMEWORK	ADVANTAGES	DISADVANTAGES
Formal Logic Production Rules	Powerful search and selection functions. Past success in AI problem solving systems.	Poor for handling large volumes of data/models. Large search space. Loss of concept relations.
Semantic Inheritance Networks	Powerful classification and categorization functions. Maintains relations between problem, model and data.	Lack of direct support for multiple levels of logic. Poor for handling complex model and/or decision domains.
Frames	Improved ability to represent a complex decision and/or model domain. Permits multiple representations of model characteristics and logic. Predicate calculus formal logic can be used to store a representation of problem/model characteristics in the frame.	Frames must be predefined in the context of a problem environment. Constrains the selection of alternative useful solutions not defined at the time of the design.
Relational	Improved ability to integrate data and model bases. Excellent management and control functions. Powerful relational query language. Past experience with DBMS systems.	Less suitable for decision processing. Difficult to adapt a data management technique to storage of models that are inherently more complex, dynamic structures.

organizations as a result of the increased availability and decreased cost of modeling languages, general purpose spreadsheets and microcomputers.

These two factors suggest that MMS may be implemented as either a component of a problem-solving DSS or as a centralized system to manage and control an organizational model base. The objective of an organization in developing a MMS is a critical factor in determining the model representation which provides the most effective and efficient model storage and model manipulation. The advantages and disadvantages of each of the four model representations, presented in Table 1, suggest that different model representations may be selected for the implementation of an MMS depending on the design objectives of the system and the number and variety of models to be stored in the model base.

SYSTEM DESIGN OBJECTIVE

The first factor which must be considered in the design of a model management system is the primary objective for the system design. Two functional categories of MMS have been identified. These are: (1) decision processing MMS and (2) model processing MMS.

A decision processing MMS is developed to function as the modeling component of a problem-solving DSS. The primary system design objective for this type of MMS is to enable the system to choose the most appropriate model or string of models to solve a given decision problem. The system must support a portfolio of models that are applicable to the problem environment of the system. Access to these models is based primarily on the characteristics of the given decision problem.

The abstraction/frame representation appears to be the ideal model representation for implementing this type of MMS. The ability to store the relationship of the model and problem within the abstraction/frame structure allows the system to select a model based on a description of the problem by the user. Storage of the model/problem characteristics in the form of well-formed functions and predicates or a linear representation permits the use of predicate calculus resolution techniques and production rules for model selection and manipulation. The power of these techniques for implementing problem-solving logic has been well documented in the artificial intelligence and expert system literature.

This representation is appropriate for DSS designed to support a variety of decision situations involving structured, semi-structured and unstructured problems. The basic abstraction/frame representation must be augmented by the ability to learn, however, if the problems are unstructured or the range of problems cannot be defined prior to system implementation.

A model processing MMS is developed to function as a centralized system for management and control of an organizational model base. The primary system design objective for this type of MMS is to assure the integrity, consistency, currency and security of the model base.

The relational representation appears to be the ideal model representation for implementing this type of MMS. The ability to store and access models based upon their input/output characteristics provides for optimal manipulation and control of a large number of models for a variety of problem situations. The power of the relational algebra query language has been well documented for use in data management. Blanning (1982) has made significant progress in identifying the key factors for management of models within this framework. Identification of the major processing anomalies which can affect the integrity of the model base and the strategies to avoid these processing anomalies has been proposed. The compatibility of the MMS with centralized organizational DBMS is also enhanced by this representation.

COMPLEXITY OF THE SYSTEM DOMAIN

A second factor that affects the choice of a model representation for the design of a MMS is the complexity of the system domain. This includes the number, variety and structure of the decision problems and models that will be supported by the system. Tha ability of a model representation to categorize and classify models based on user-defined model/problem characteristics is essential for improving the efficiency of an MMS that must support a large number and variety of problems and/or models.

The semantic inheritance network provides a powerful representation for the categorization and classification of knowledge. The strength of this representation lies in its inheritance property and membership class structure, which allows the system designer to represent assertions and knowledge about many entities at one time. This is operationalized by linking information to a node representing an entire class. Production rules and predicate calculus resolution techniques can then be applied to allow for an efficient search of the network depending on the characteristics of the class. In the case of an MMS this network structure could be designed to represent a class of problems or a class of models.

The semantic inheritance network representation can be used in conjunction with the abstraction/frame representation to improve the functioning of a decision processing MMS if a large number and variety of models are required to support the problem environment of the DSS. Nodes are developed to represent classes of problems or classes of models. Property and membership links connect the concept nodes (denoting a class) to the frame/abstraction nodes which would describe the model attributes, problem attributes, procedures and assertions necessary to describe and implement a given model for a given problem. The powerful logic capabilities of first-order predicate calculus and production rules can be used within the frame/abstraction to store the attributes, procedures and assertions. Predicate calculus resolution techniques and/or production rules could also be used to search the network in order to find the appropriate class of problems/models. Similar search strategies can then be used to search the class subnet and to drive the frame/abstraction logic. Techniques, such as network partitioning into "net spaces" (Hendrix, 1975) and network clustering techniques (Alshawi, 1982), can also be employed to improve the efficiency of the network functioning.

The relational model representation is also extremely useful for management of a model base in which a large number and variety of models must be stored. This representation is especially appropriate for a model processing MMS with a large, complex model base structure.

Table 2 presents a summary of the recommendations for the selection of a model representation for an MMS depending on the functional category of the MMS and the size and complexity of the model base.

Design of a Decision Processing MMS

The framework for classification of an MMS, presented in Table 2, has been used for the design of an MMS that is currently being implemented at the University of Arizona. The MMS is intended to function as a component

Table 2

Summary of Model Representations for
Model Management System Design

FUNCTIONAL CLASS		SYSTEM DOMAIN COMPLEXITY	
		Narrow decision and model domains	*Complex, broad decision and model domains*
DPMMS	Mainpulation Component	Production rules and a menu-driven command language can be be used for model manipulation. A frame representation can be used to organize the model and decision relationships.	A semantic inheritance network, combined with a frame representation, can be used to classify and organize decision and model knowledge to improve model manipulation efficiency.
	Storage Component	Model instances can be stored using subroutine libraries. Storage of decisions is frequently unnecessary due to the narrow decision domain.	Specific models and decisions can be stored as instances of the frame representations. A relational representation can also be used to store decisions for very complex, broad decision domains if the decision and model relations are loosely coupled.
MPMMS	Manipulation Component	Model manipulation can be implemented using a menu-driven command language and production rules. The use of a frame representation is usually not indicated since there is less need to specify the decision and model relations.	A relational query language can be used to implement the model description and model manipulation. An example is the Model Query Language (MQL) proposed by Blanning.
	Storage Component	Models can be stored in subroutine libraries. A sophisticated model representation is usually unnecessary.	A relational representation can be used to store the models within a centralized model base.

DPMMS = DECISION PROCESSING MODEL MANAGEMENT SYSTEM
MPMMS = MODEL PROCESSING MODEL MANAGEMENT SYSTEM

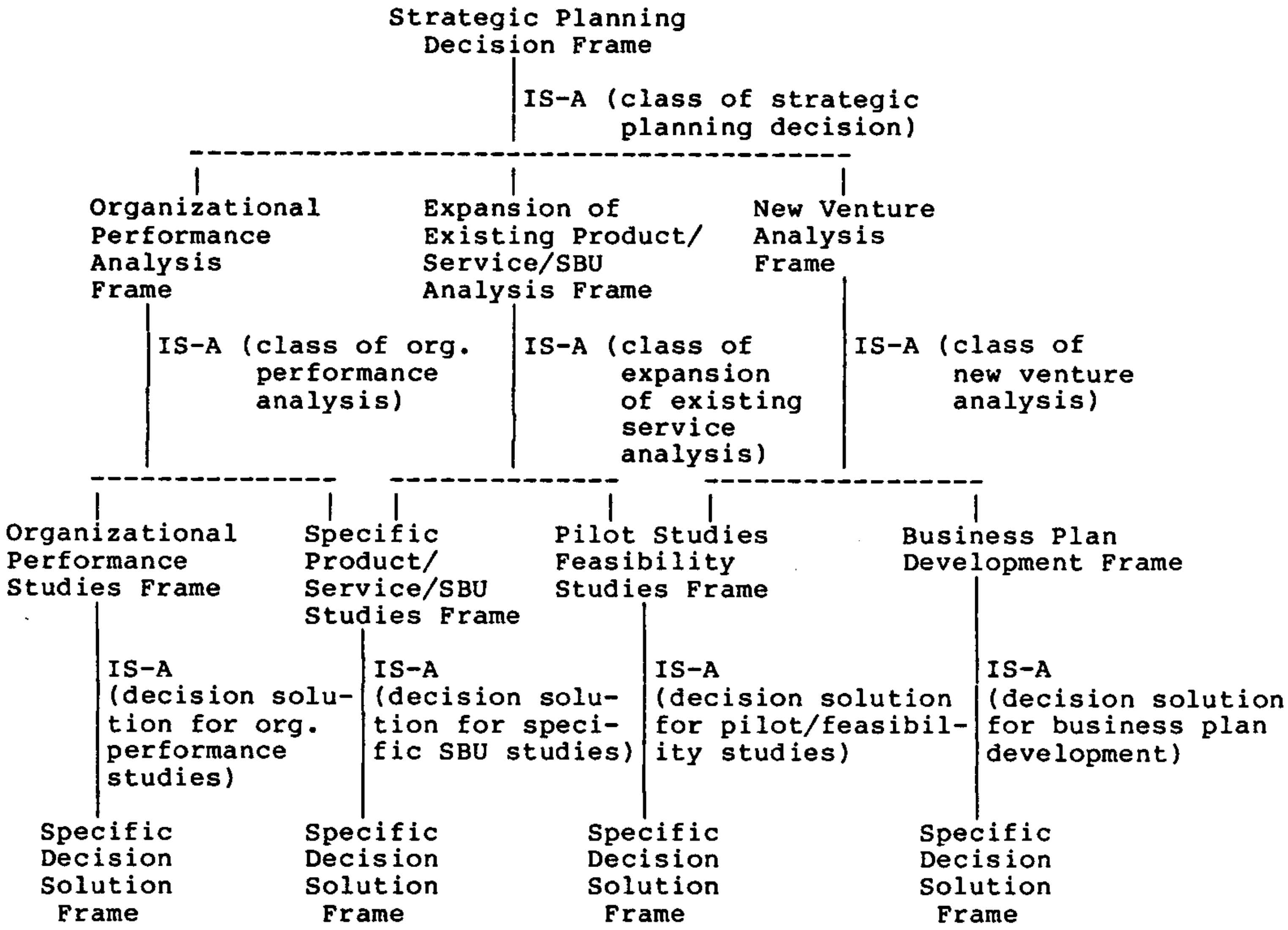
Strategic Planning Decision Frame
IS-A (class of strategic planning decision)
Organizational Performance Analysis Frame
Expansion of Existing Product/ Service/SBU Analysis Frame
New Venture Analysis Frame
IS-A (class of org. performance analysis)
IS-A (class of expansion of existing service analysis)
IS-A (class of new venture analysis)
Organizational Performance Studies Frame
Specific Product/ Service/SBU Studies Frame
Pilot Studies Feasibility Studies Frame
Business Plan Development Frame
IS-A (decision solution for org. performance studies)
IS-A (decision solution for specific SBU studies)
IS-A (decision solution for pilot/feasibility studies)
IS-A (decision solution for business plan development)
Specific Decision Solution Frame
Specific Decision Solution Frame
Specific Decision Solution Frame
Specific Decision Solution Frame

Figure 5

Decision Manipulation Component:
Strategic Planning Model Management System

of a strategic planning DSS. The MMS is classified as a decision processing MMS because the primary design objective of the system is to implement the modeling component of a DSS. A frame representation is used as the basic representation for the model and decision components of the system. A semantic inheritance network representation is also used to organize and classify the model and decision frames to improve the efficiency of the system. The complexity of the strategic planning decision processes and the strategic planning model base influenced the decision to use both a semantic inheritance network and a frame representation for the system. Production rules are used to implement the model manipulation components of the system. An in-depth discussion of the decision processing MMS architecture and the system design of the Strategic Planning Model Management System is presented in a follow-up paper (Applegate, Konsynski and Nunamaker, 1985).

There are three major components of the decision processing Model Management System. These are (1) a decision component, (2) a model component, and (3) a data component. The decision component of the system provides the user with the ability to describe, analyze and store organizational decisions. This component is similar to the problem processing system described by Bonczek, Holsapple and Whinston in their work on the design of a generalized decision support system (Bonczek, Holsapple and Whinston, 1981). The decision component accesses the model component to retrieve, sequence and control the organizational models needed for solving a specific decision problem. The model component accesses the data component to retrieve, sequence and control the organizational data needed for implementing a specific model.

Design of the decision component of the system involves the design of a decision manipulation component (decision net and decision frames) and a decision storage component (decision instances). The semantic inheritance network and frame/abstraction representations are used to represent the decision knowledge within the system. Figure 5 presents an overview of the decision manipulation component of the Strategic Planning Model Management System. It is important to note that the network representation presented in Figure 5 expresses the decision relationships for a *specific* strategic planning decision domain. It is not intended to represent the decision domain for every strategic planning MMS.

The semantic inheritance decision network functions to organize and classify the diverse strategic planning analysis decision frames. The network is defined on four levels. At the first two levels of the network, the frames contain the system control rules for describing the current strategic planning decision and for selecting a class of strategic planning analysis methods. At the third level of the network, the frames contain the system control rules for selecting the specific studies needed for a class of strategic planning analysis and the model classes that will be needed to implement the study. At the fourth level of the network, the frames contain the system control rules for solving the specific decision problem. This includes the rules for selecting and sequencing the specific models needed for the analysis. Network maintenance rules for update, storage and retrieval of the decision problems are present at each level of the network.

The decision history of the organization is stored in the decision storage component of the system. Decision instances are stored as specific implementations of the decision solution frames.

The decision component accesses the model component through the use of production rules that are stored in the decision solution frames in the decision net. Design of the model component of the system involves the design of a model manipulation component (model net and model frames) and a model storage component (model instances). The semantic inheritance network and frame/abstraction representations are also used to represent the model knowledge within the system. Figure 6 presents a subset of the model manipulation component of the Strategic Planning Model Management System. Other classes of models (e.g., financial models, accounting models, mathematical programming models) are implemented in a similar manner.

The semantic inheritance model network functions to organize and classify the strategic planning model frames. These models are classified according to their role within the strategic planning process and their functional analytic class. This provides access to the models for analysis of a specific phase of the strategic planning process and also allows for direct access of the models for functional data analysis (e.g., regression analysis).

The modeling history of the organization is stored in the model storage component of the system. Model instances are stored as specific implementatons of the model solution frames.

Conclusion

This paper has presented a framework for evaluating and selecting a model representation scheme for the design of an MMS based on the objectives for the system and the complexity of the model base structure. A functional classification is used to categorize MMS by the primary

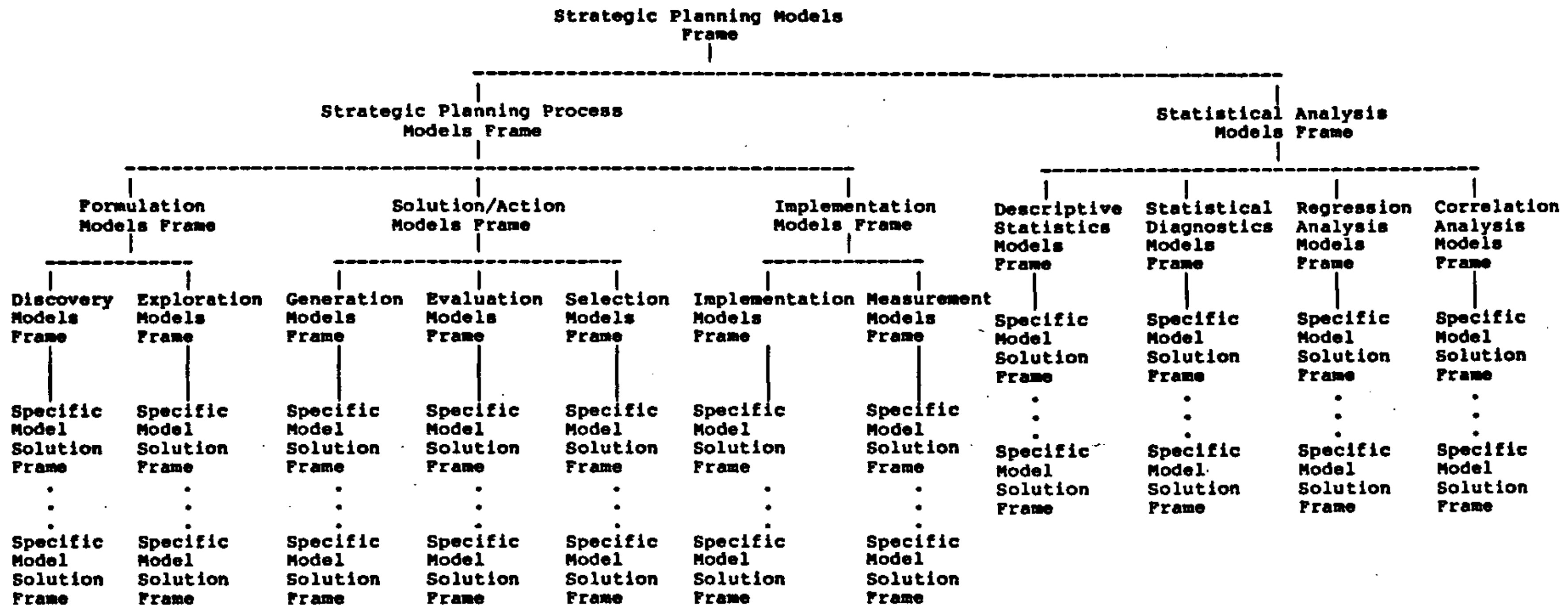
Strategic Planning Models Frame
Strategic Planning Process Models Frame
Statistical Analysis Models Frame
Formulation Models Frame
Solution/Action Models Frame
Implementation Models Frame
Descriptive Statistics Models Frame
Statistical Diagnostics Models Frame
Regression Analysis Models Frame
Correlation Analysis Models Frame
Discovery Models Frame
Exploration Models Frame
Generation Models Frame
Evaluation Models Frame
Selection Models Frame
Implementation Models Frame
Measurement Models Frame
Specific Model Solution Frame

Figure 6

Model Manipulation Component:
Strategic Planning Model Management System

note for figure 6

*NOTE: This is only a partial semantic inheritance network/frame representation of the modeling knowledge stored within the system. Other classes of models (e.g. financial models, accounting models, mathematical programming models) are implemented in a similar manner.

design objective of the system. Two functional classes of MMS have been identified: (1) decision processing MMS and (2) model processing MMS.

A decision processing MMS serves the primary function of implementing the modeling component of a generalized DSS. Small systems that do not possess a complex modeling or decision structure can be implemented using a menu-driven command language and a subroutine library of models. A frame/abstraction model representation can be used to organize the model/decision relationships. Semantic inheritance networks can be used to improve the efficiency and scope of systems with a broad, complex decision and/or model domain.

A model processing MMS is used primarily for centralized management of organizational models. These systems function to insure the integrity, consistency, currency and security of the model base in a manner similar to a centralized database management system. Again, small systems that do not possess a comlex modeling or decision structure can be implemented using a menu-driven command language and a subroutine library of models. A relational representation can be used to improve the efficiency and scope of systems with a complex model domain.

An example of the use of the MMS design framework for the design of a decision processing MMS is presented. A semantic inheritance network and frame/abstraction representation was chosen to implement the system design.

The ability to provide organizatons with flexible and responsive modeling capabilities is becoming a reality. MMS frameworks, based on different model representation schemes, have been proposed as a means to support the modeling needs of an organization. The ability of system designers to tailor the MMS design to the modeling needs of the organization will significantly enhance the implementation of flexible and responsive MMS within organizations.

Bibliography

Alshawi, H., "A Clustering Technique for Semantic Network Processing", *Conference Proceedings: European Conference on Artificial Intelligence*, 1982.

Angell, T. and Randell, T.M., "Generalized Data Management Systems", *IEEE Computer News*, 2(2):5-12, Nov., 1969.

Applegate, L.M., Konsynski, B.R. and Nunamaker, J.F., "Design of a Decision Processing Model Management System," *Technical Working Paper*, Department of Management Information Systems University of Arizona, 1985.

Bisschop and Meerhaus, "Toward Successful Modeling Applicatons in a Strategic Planning Environment", *Large Scale Linear Programming Vol. 2*, (ed.) Dantzig, Dempster and Kallio, Luxembourg: IIASA, 1981.

Blanning, R., "A Relational Framework for Model Management in Decision Support Systems", *DSS-82 Transactions*, 1982.

Blanning, R., "The Existence and Uniqueness of Joins in Relational Model Banks", *Owen Graduate School of Management*, Vanderbilt University, 1982.

Blanning, R., "Issues in the Design of Relational Model Management Systems", *National Computer Conference*, 1983.

Bonczek, R.H., Holsapple, C.W., Whinston, A.B., "A Generalized Decision Support System Using Predicate Calculus and Network Data Base Management", *Operations Research*, 29(2):263-281, 1981.

Brachman, R.J., "A Structural Paradigm for Representing Knowledge", Report No. 3605, Bolt, Beranek and Newman, Inc., 1978.

CODASYL Systems Committee, *CODASYL Data Base Task Group Report*, NY: ACM, 1971.

Codd, E.F. "A Relational Model of Data for Large Shared Data Banks", *CACM*, 13(6):377-387, 1970.

Elam, J.J., Henderson, J.C., Miller, L.W., "Model Management Systems: An Approach to Decision Support in Complex Organizations", *Proceedings of the First Conference on Information Systems*, 1980.

Ernst, G. and Newell, A. *GPS: A Case Study in Generality and Problem Solving*, NY: Academic Press, 1969.

Hendrix, G.G., "Expanding the Utility of Semantic Networks Through Partitioning", *4th International Joint Conference on Artificial Intelligence*, 1975.

Kein, R.T. and Philippakis, A.S., "Decision Support Systems in Practice: Profile on Management and Professional Workstations", *Proceedings of the 5th International DSS-85 Conference*, (ed.) Elam, Cambridge: IADSS, 1985.

Klein, G., Konsynski, B., and Beck, P., "A Linear Representation for Model Management in a DSS," *MIS Technical Report 83-3*, University of Arizona, 1983.

Konsynski, B. and Dolk, D., "Knowledge Abstractions in Model Management", *DSS-82*, 1982.

Konsynski, B., Kotteman, J., Nunamaker, J. and Stott, J., "Plexsys-84: An Integrated Development Environment for Information Systems", *Journal of Management Information Systems*, I(3):64-104, 1984.

Marsten, R., "The Design of the XMP Linear Programming Library", *ACM Transactions on Mathematical Software*, 1981.

Minker, J. "General Data Management Systems—Some Perspectives", *University of Maryland Computer Sciences Center Technical Report*, 1969.

Minsky, M., "A Framework for Representing Knowledge", *The Psychology of Computer Vision*, ed. P.H. Whinston, NY:McGraw Hill, 1975.

Nilsson, N.J., *Principles of Artificial Intelligence* Palo Alto: Tioga Publishing Co., 1980.

Nunamaker, J.F., Swenson, D.E. and Whinston, A.B., "Specifications for the Development of a Generalized Data Base Planning System", *AFIPS Conference Proceedings*, June, 1983.

Quillian, M.R., "Semantic Memory", *Semantic Information Processing*, ed. M. Minsky, Cambridge: MIT Press, 1968.

Rice, J.R. and Rosen, S., "NAPSS: A Numerical Analysis Problem Solving System", *Proceedings of the ACM National Conference* 1966.

SPSS Inc., *SPSSx*, NY:McGraw Hill, 1983.

Ulvila, J.W. and Brown, R.V., "Decision Analysis Comes of Age", *Harvard Business Review*, Sept-Oct., 1982.

Sprague, R.H. and Carlson, E.D., *Building Effective Decision Support Systems*, NJ: Prentice Hall, 1982.

Will, H.J., "Model Management Systems", *Information Systems and Organization Structure* (id.) Grochla and Syperski, Berlin: Walter deGruyter, 1975.