SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: 79-02-04	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE *(and Subtitle)*: MODEL MANAGEMENT SYSTEMS: A FRAMEWORK FOR DEVELOPMENT	5. TYPE OF REPORT & PERIOD COVERED: Technical Report, Jan. 78-Dec. 79
6. PERFORMING ORG. REPORT NUMBER: 79-02-04	
7. AUTHOR(s): Joyce J. Elam	8. CONTRACT OR GRANT NUMBER(s): N00014-75-C-0440
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Department of Decision Sciences, The University of Pennsylvania, Philadelphia, PA 19104	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: Task NR049-360
11. CONTROLLING OFFICE NAME AND ADDRESS: Office of Naval Research, The Department of the Navy, 800 N. Quincy St., Arlington, VA 22217	12. REPORT DATE: February 1979
	13. NUMBER OF PAGES: 24
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*: —	15. SECURITY CLASS. *(of this report)*: Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

See next page.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

Wharton

Department of Decision Sciences

University of
Pennsylvania
Philadelphia PA 19104

MODEL MANAGEMENT SYSTEMS: A FRAMEWORK FOR DEVELOPMENT

Joyce J. Elam

79-02-04

Department of Decision Sciences
The Wharton School
University of Pennsylvania
Philadelphia, PA 19104

Draft # 1

February 12, 1979

This research was supported in part by
ONR N00014-75-C-0440

ABSTRACT

This paper presents an architecture for a generalized model management system that facilitates the integration of management science models into a decision support system. The objective of the system is to support the decision-maker both in specifying a problem and in effecting a solution. This is accomplished by providing him/her with a means for interacting with a complex structured database to specify the structure of some problem; and to solve the model defined for the problem using appropriate information -- either from the database or some other source -- and efficient solution procedures.

MODEL MANAGEMENT SYSTEMS: A FRAMEWORK FOR DEVELOPMENT

Joyce J. Elam

1.0 INTRODUCTION

This paper presents a general framework for the development of a model management system (MMS) that facilitates the use of mathematical models and techniques by a decision-maker in an interactive problem solving environment. The objective of the system is to support the decision-maker both in specifying a problem and in effecting a solution. This is accomplished by providing him/her with a means for interacting with a complex structured database to construct a model of some problem, to find, if available, a previously developed model for the problem, and to solve the model defined for the problem using appropriate information-- either from the database or some other source-- and efficient solution procedures.

The philsophy which underlies the design of this system is (1) models, like data, are an organizational resource and can be described, executed, and manipulated by some generalized software system and (2) a general framework can be designed for managing a variety of model types (optimization, sub-optimization (heuristics), statistical,

simulation, descriptive, etc.). By viewing models in an analogously way as data, much of the recent research in database management systems can applied to this research.

2.0 NEED FOR RESEARCH

The problems facing today's decision-maker - both in the private and public sectors - are highly complex, are continually changing--both as a result of a dynamic environment or a changed perception of the problem-- and require immediate solutions that have far-reaching implications. In addition, the data that impacts on solutions to these problems is voluminous, dynamic, and may originate from many different sources. The complexity of these problems has necessitated the use of mathematical models and efficient data management capabilities to organize the voluminous amounts of data into useful information.

Data management capabilities have been greatly enhanced with the development of database management systems. Much research has focused on efficient ways to create, access, and maintain the large collection of data of an organization. A major advantage of database management systems is that the physical storage details are transparent to the users or procedures that require access to the database.

Optimization algorithms and mathematical methods such as linear programming, network analysis, simple and multiple regression, exponential smoothing, Monte Carlo simulation, etc. have been the subject of intensive research during the last few years. As a result, the cost of solving various models has dropped significantly. It is now computationally feasible for organizations to use these techniques in supporting their decision-making.

Although considerable research has been directed toward database technology and management science independently, little research has been directed toward integrating the two technologies. This is evidenced by currently available modeling systems such as TROLL [NAT01] and TSP [HAL01] which have good analytical capabilities but poor data management capabilities and SYSTEM 2000 [MRI01], SEED [INT01], and IMS [IBM01] which provide good data management capabilities but poor analytical capabilities. Database management systems have not been designed with the goal of supporting analytical techniques. Correspondingly, analytical systems have assumed that the data to be analyzed preexists in some standard form and have not addressed the problems associated with structuring the data in the proper format.

The lack of integration has resulted in the following:

1. Current database management systems are used in applications aimed at operational control or management control in organizations. The major

concerns of such systems are with the raw data of an organization. In general, current systems do not have the capabilities to support higher-level decision making.

2. Mathematical techniques are not being efficiently used. An extremely important aspect for the implementation, use, and acceptance of a mathematical model is its informational requirements and accesibility. Too often it is difficult if not impossible to extract the data needed for an analysis from the organizational data base. Even if the data is obtained, it is up to the user to reformat the data to conform to the data requirements of the analytical software system to be used.

The major impact of management science has been on structured problems where the decision-maker can be provided with detailed recommendation for handling these problems [KEE01]. Expanding the use of management science techniques for supporting the decision-making required by less structured problems requires a more extensive involvement of the decision-maker. The MMS will facilitate this involvement.

3.0 MODEL MANAGEMENT SYSTEMS: WHAT CURRENTLY EXISTS

There have been attempts over the past several years to automate the building and processing of computer-based information systems. This section will briefly review this research and summarize those aspects that relate to the development of a generalized MMS.

The ISDOS project developed by Teichroew and others [TEI01] at the University of Michigan provides computer-aided techniques for defining, recording, and analyzing functional requirements for information processing systems. The ultimate object of the ISDOS project is to produce executable software for a particular computing environment directly from a set of functional requirements. ISDOS contains two major components: PSL (Problem Statement Language) -- a formal language for expressing in a structured form the major components and relationships that exist in a system that is to be computerized and PSA (Problem Statement Analyzer) -- a software system that analyzes the requirements specified in the PSL and attempts to construct software modules and databases in accordance with these requirements and some stated performance criteria. The concept of a PSL has direct implications for the model definition language used in the MMS in that the MMS should have a language for specifying the general characteristics of a model independent of the specific programming language that is eventually used to represent

the problem in a structured form suitable for analysis.

Interactive planning systems have been developed for representing and manipulating models and data that relate to financial planning problems. These systems include BPL [INT02], FORESIGHT [FOR01], PLATO [IBM02], IPSY [INT03] and others [SOF01]. In general, these systems have the following capabilities: (1) a particular modeling language for specifying the structure of the problem faced by the planner (2) report-generation routines for allowing reports to be tailor-made by the user and (3) various analysis capabilities that can be defined by the modeling language or can be defined at execution time [HUR01]. These systems facilitate user interaction in the planning activity but provide no general model management functions. Few systems have the capability to interface with the operational data of the organization of interest even though the data used by the planning system tends to be a specialized processed version of the operational data of the organization.

Other software systems falling under the general category of decision support systems provide some model management functions. The GPLAN system developed by Whinston and other at Purdue [NUN01] for supporting a decision-maker in water quality management incorporates (1) a library of application models and (2) an English-like mapping language for automatically interfacing data with models. The GADS system [MAN01] developed by IBM to aid the

city of San Jose police department in the assignment of police beats facilitates person-machine interaction both in constructing a solution to the problem and in presenting various solutions to the decision-maker. However, it does not permit easy access to data either by the decision-maker or the solution process that is constructed. The system recently deveoped by Donovan and others [DON01] at MIT for nationwide energy planning information system provides for generalized interfaces between software systems but no facilities for model definition. No system incorporates a full range of model managememt functions. This is not unexpected since each system is specialized for a particular application and for a particular class of users.

The most recent relevant work in model management is the interactive modeling system developed by Katz and Miller [KAT01] to aid decision-makers in accessing the relative benefits and costs of alternative mitigation and recovery policies for natural hazards. This system is composed of three subsystems: the model subsystem, the analysis subsystem, and the library subsystem. The model subsystem applies a set of processing subroutines to an input file of victim records in order to calculate some additional attribute value(s) for each record. The output of this stage is a file of expanded victim records. The analysis subsystem performs aggregation, summarization, statistical analysis, etc. on a file of victim records created by the modeling subsystem. The library subsystem manages the

libraries of subroutines available to users in the model and analysis subsystems. Each of these systems is invoked by a separate person-computer interface. By means of these interfaces, user interaction with the system is facilitated. The main limitations of this system can be summarized as follows:

1. A variety of model types cannot be defined
2. Although this is one of the few systems that contain a library function, it is rather limited in scope.
3. The system does not interface directly with some generalized data base.

This system does, however, contain some unique features for facilitating interactions with users, for integrating new processing subroutines into the subsystems, and for linking data values with the processing subroutines. Many of the ideas set forth in this paper for a generalized model management system are drawn from this work.

The DAISY system [HUR02, MOR01] developed at the University of Pennsylvania strives to aid the decision-maker in his/her decision process by providing easy access to programs, data, and models that are available and relate to some general decision problem. Although some research has been directed toward interfacing models with data [MIT01],

no general model management capabilities are available. The MMS specified in this paper can be integrated within the DAISY framework, drawing on the other existing components of DAISY when necessary. In a like manner, a general MMS could be incorporated into any of the specialized systems listed above.

4.0 FRAMEWORK FOR DESIGN

4.1 Introduction

A computer-based information system can be viewed as consisting of four major software components [JOH01]:

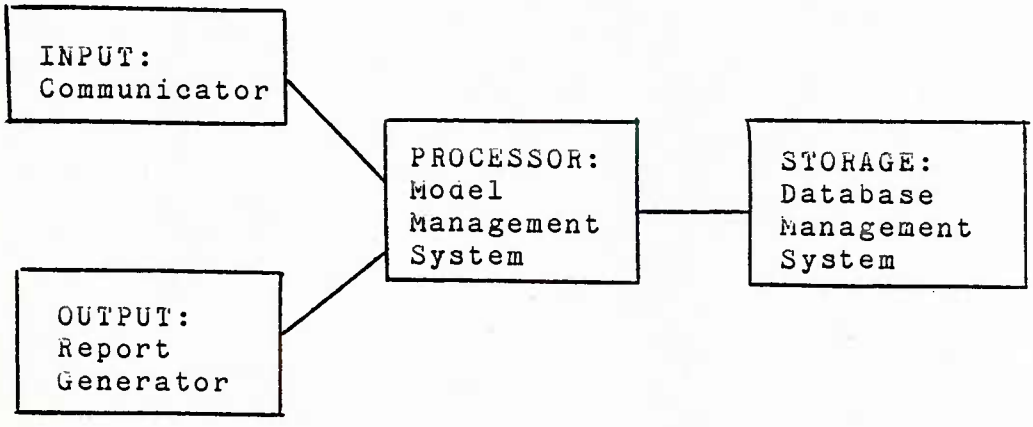


Figure 1

The communicator is responsible for facilitating interchange between a user and the information system; the Report Generator, for providing system-generated information in user-specified form ; the Database Management System, for retrieving and storing information; the Model Management System (MMS), for all other processing in the system. Each process under control of the MMS will be referred to as a

model.

The MMS thus forms the kernel of computer-based information system. The remainder of this paper focuses on its design and its interaction with the other components.

4.2 Objectives

Effective model management encompasses all phases of model activity -- construction, testing, execution, validation and maintenance -- and facilitates the use of models that are simple, robust, easy to control, adaptive, complete on important issues, and easy to communicate with [LITT01]. The following general objectives for the MMS are formulated within this context.

1. The MMS should provide both formal and informal descriptions of a model and the assumptions that went into constructing it.

2. Model description should be separated from model execution.

3. The execution of a model (the joining of a model process with its information requirements) should be automatic.

4. The MMS should provide information on what types of models exist and for what purposes.

5. The MMS should be an active system in that it (1) alerts user when model assumptions are violated and (2) provides alerts concerning model validity.

6. The MMS should allow a model to be changed easily.

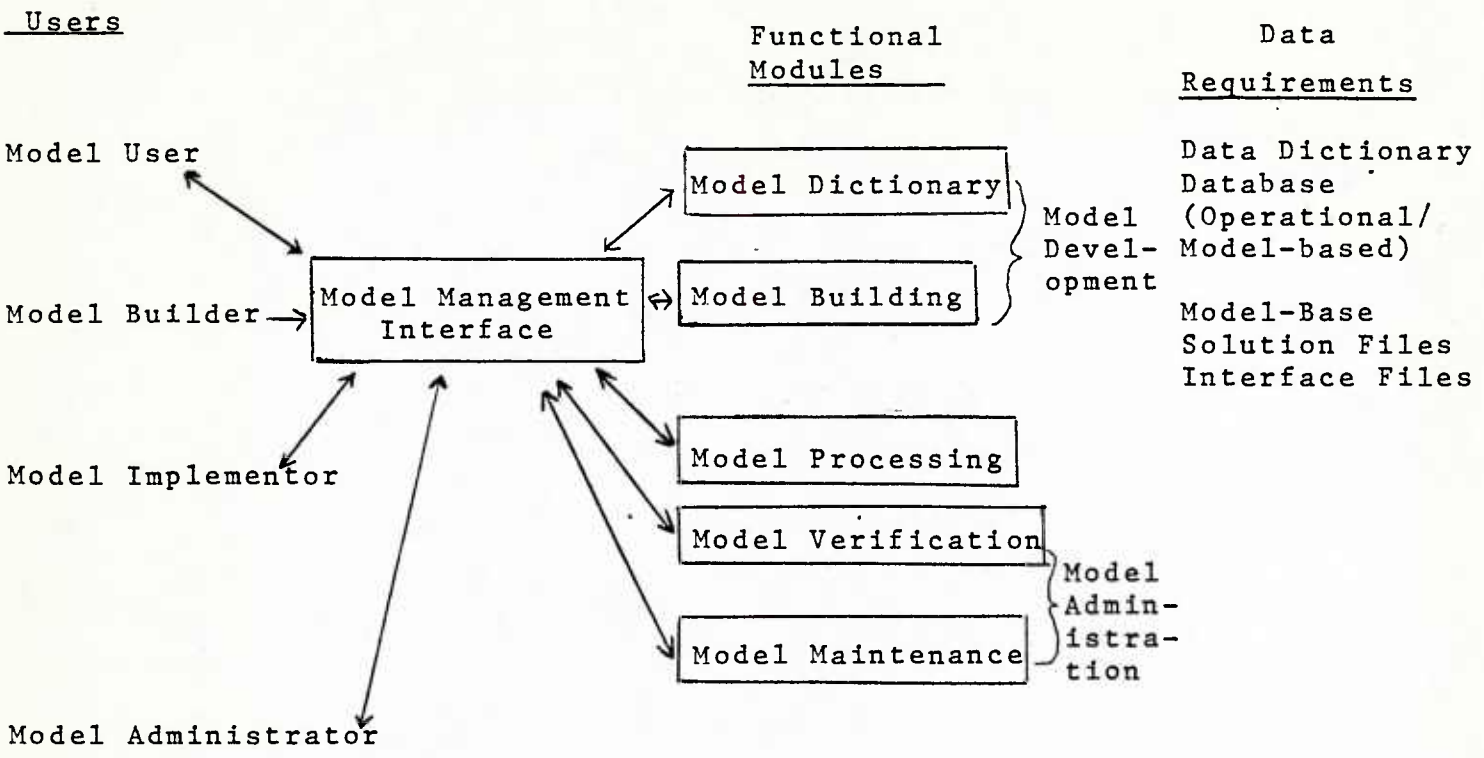
7. The MMS should provide information on model useage, cost, security.

4.3 Architecture

An overview of the architecture proposed for the MMS is depicted in Figure 2. The three components of this architecture -- users, functional modules, and data -- are detailed below.

4.3.1 Users -

The users who interface with the MMS can be divided into four classes: the model user, the model builder, the model implementor and the model administrator. These users are involved with different phases of the modeling activiy and interact with the MMS at different levels. A similar classification of users was proposed by Katz and Miller [KAT01].



Model Management System Architecture

Figure 2

The model user interacts with the MMS at a high level to find and execute previously developed models. The model user through the Communicator issues commands to initiate model execution and responds to any prompts for data. The MMS automatically performs model execution and presents the results as specified by the user. The model user is typcially a non-programmer who requires minimal knowledge of the system.

The model builder interacts with the MMS to construct models. The model builder is supplied with commands for data collection, data analysis, and model assembly. The model builder is a more sophisticated user than a model user and requires a more thorough knowledge of the system. This user does not, however, need to be a computer programmer.

The model implementor interacts with the MMS to provide the interfaces between model definition, data requirements, and model processing programs that are necessary to support automatic model execution and validation. The model implementor may also provide the computer programs necessary for model processing (These may also be supplied from other sources.). The model implementor interacts at a low level with the MMS and requires computer programming and analytical skills.

The model administrator has overall responsibility for the MMS. He/she ensures that the MMS objectives are met in the most efficient manner. The model administrator is

involved with model debugging, testing, validation, documentation, accounting and access.

4.3.2 Functional Modules -

The MMS is divided into four major components: Model Development, Model Processing, Model Administration, Model Interface. The functions contained in each component are detailed below.

4.3.2.1 Model Development -

The basic functions of the model development component are to (1) support interactive model building and (2) to provide information about previously defined models to users and to the other components of the MMS.

The process of model building involves data collection, data analysis, and model assembly. We assume that the data to be collected is contained in the operational database of the organization or in a special model-related database. The model building subsystem must contain a query language for easily accessing this data. These languages are available in many database managment systems [INT01,MRI01] and can easily be interfaced with the model building subsystem. The major focus of the model building subsystem will be on model assembly. [Data analysis can be viewed in the same manner as any model which is built, executed, and

analyzed through the MMS and data collection is supported by the Database Management System.]

A basic concept underlying the design of the MMS is that the description of a model is separated from its actual physical realization. The decision-maker who builds a model of some decision process (i.e., the model builder) should have a language for expressing this model in informal, English-like terms. A general design for a model description language (MDL) is shown in Figure 3. Using this language, a model is characterized by a set of entities, a set of relationships between these entities, and a set of assumptions upon which these relationships are based. The entities and relationships are described by a set of attributes that are defined as being either controllable or uncontrollable. At any instance of time, particular data values can be associated with each attribute. The data values associated with controllable attributes are (1) user supplied (2) data-base supplied or (3) model-supplied (i.e., the result of some previous model process). The data values associated with uncontrollable attributes are determined by applying a model processing program to the model

```
MODEL NAME:

ENTITY NAME:
        ATTRIBUTE NAME:
             TYPE:<CONTROLLABLE,UNCONTROLLABLE>
             SOURCE:<Database, User, Model>
        SELECTION CRITERIA:


RELATIONSHIP NAME:
        ENTITIES INVOLVED:
        ATTRIBUTE NAME:
             TYPE:<CONTROLLABLE, UNCONTROLLABLE>
             SOURCE:<Database, User, Model>
        SELECTION CRITERIA:


MODEL TYPE:

MODEL SOLUTION:
MODEL INTERFACE:
VERSION:
DATE:
KEYWORDS:
MODEL ASSUMPTIONS:NAME:<Assumption Name>:predicate
```

Figure 3

definition. These data values are stored in the data base, reported to the user, and/or supplied as input to another model. The remaining information provided by the MDL supports other model management functions to be discussed subsequently. The model building subsystem contains software to process the model definition as expressed in the MDL. Using an available data dictionary and the organizational database (operational and model-based), this software creates a model-base. The model-base is the primary means for logically linking data to the model definition.

Another major function included within model development is the model dictionary. The model dictionary subsytem supports users in determining what types of models exist and for what purposes. It also provides information on the types of model processing programs (solution procedures) that are available and their informational requirements. The model dictionary subsystem interacts with two basic types of data: the model-base and the interface files. The interface files contain information on solution procedures and will be discussed more thoroughly in model processing. Using this data, the model dictionary subsystem can provide the following information:

1. The input and output specifications for each set of model processing programs (solution procedures) managed by the MMS.

2. The useage of specified data items in model descriptions.

3. Model assumptions.

4. Models that relate to specific applications.

5. Solution procedures that are available for various model types.

6. Model descriptions.

4.3.2.2 Model Processing -

As explained in Model Development, data values can be associated with the attributes that define some model. Model processing involves the physical linking of data values to these attributes. The linking of data values to controllable attributes is referred to as creating a model instance. The linking of a model instance to a solution procedure is referred to as model execution. The linking of data values to uncontrollable attributes is referred to as model solution. Model processing will perform this linking in much the same manner as proposed in [KAT01]. The system will do what linking it can automatically and will leave the rest to the user.

Model processing interacts with four basic types of data: the database, the model-base, interface files and solution procedure files. To create a model instance, the model execution subsystem uses the database, model-base, and user-supplied data. If the source of any controllable attribute is a model process, a submodel instance is created and processed in order to obtain the appropriate data values. The linking of a model instance to a solution procedure is accomplished through the use of interface files. The interface files contain information on input and output requirements of a solution procedure. Many different models that are defined may use the same solution procedure. By associating a general interface file with each solution

procedure, the automatic linkage of data to model processing programs is facilitated. It is expected, however, that some interaction between the user and the system will be necessary to perform linkages when using a general interface file. In addition to general interface files, specialized interface files can be constructed for a particular model and solution procedure that would minimize the need for user interaction in executing the model. This would be desirable for frequently executed models. Once a model has been processed (i.e., physically linked to a solution procedure) it can be solved either in batch or interactive mode.

Once a model has been solved, the results can be presented to a user, stored in the database, or used in another process. The Model Processing System again interacts with the model-base, data-base, interface files, and user profile files to output the results in the appropriate way. The user-profile files contain information on the format of output desired by each individual user. The Model Processing component also will interact with the Report Generator to present the results of model solution in a specified form.

4.3.3 Model Administration -

Model Administration is concerned with model validation and model maintenance. The model validation subsystem is responsible for monitoring the model assumptions and informing a user when the assumptions are violated during model processing.The model assumptions can be specified in a similar manner as integrity constraints in database management systems [DAT01]. The model validation system should also provide the capabilities for replicatively validating a model - matching the model-generated data with data already available in the database - and for predictively validating a model - matching the model-generated data with data to be stored in the database at some future point.

The model maintenance subsystem provides software to modify model descriptions, monitor access to models, provide back-up, and create and maintain user profile files, interface files, and solution files.

4.3.4 Model Interface -

The MMS is a command-driven system. The Model Interface analyzes commands for various model-management functions supplied by the user through the Communicator and invokes the appropriate subsystem for performing that command. The basic commands to be supported correspond to the modeling activities of development, processing, and administration. A sample list of commands is shown in

Admini-stration	BACK	Provide Back-up
	NEW	Add new file
	MODIFY	modify existing file
	USEAGE	Provide information on useage of models
	DELETE	Delete file

Figure 4

5.0 CONCLUSIONS

The purpose of this paper has been to outline the architecture for a generalized model management system that facilitates the integration of management science models into a decision support system. The MMS provides support for the modeling activity through Model Development, Model Processing and Model Administration. Through model development, a decision-maker can formulate a model of some decision process and relate this model to other operating models within the system. Model processing provides a high-level mechanism for interfacing data, models and solution procedures so that the user is relieved of low-level data management functions. Model administration allows the collection of models to be treated as an organizational resource and managed accordingly. The MMS provides a general framework for incorporating into a decision process the use of any type of formal model.

Bibliography

DAT01 Date, C. J., An Introduction to Database Systems, Addison-Wesley (1972).

DON01 Donovan, John, "Database System Approach to Management Decision Support", TODS, 1,4, pp. 344-369 (1976).

FOR01 Foresight Systems, Inc. 276 Madison Avenue, New York, NY 10016

HAL01 Hall, R. E. , TSP Manual, Harvard Tech. Rep. No. 12, Harvard Institute of Economic Research, Cambridge, MA (April 1975).

HUR01 Hurst, E. G., "Interactive Planning Systems -- Their Characteristics, Use and Future", Working Paper 74-11-04, Department of Decision Sciences, University of Pennsylvania (1974).

HUR02 Hurst, E. G., Morgan, H. and Ness D., "DAISY: A Decision-Aiding Information System", Working Paper 75-01-05, Department of Decision Sciences, University of Pennsylvania (1975).

IBM01 IBM Information Management System: General Information Manual, Form No. GH20-1260, White Plains, NY (1975).

IBM02 IBM, PLATO:Software Planning Tools, White Plains, NY.

INT01 International Database Systems, SEED Reference Manual, Philadelphi, PA (1976).

INT02 International Time Sharing, Jonathan Industries, Chanka, Minn. 53318

INT03 Interactive Planning Systems, Inc. 18 W. 44th Street, New York, New York 10036

JOH01 Johnson, Tom Private Communication (1976).

KAT01 Katz, N. and Miller, L., "An Interactive Modeling System," Working Paper 77-09-02, Department of Decision Sciences, University of Pennsylvania (1977).

KEE01 Keen, P. and Scott Morton, M., Decision Support Systems, Addison-Wesley (1978).

LIT01 Little, J., "Models and Managers: The Concept of a Decision Calculus", Management Science, Vol. 16,

Figure 4.

Related MMS Function	Command Function	
Building	FIND	Provide Access to Database
	BUILD	Using MDL specify model structure
	CHANGE	Modify structure
	ADD	Expand structure
	SUBTRACT	Delete all (or part) of structure
Dictionary	SHOW	Display model descriptions, names, etc.
	FIND	Find models relating to specific applications
	HOW	Display how a particular model is solved
Processing	CREATE	Create a model instance
	PROCESS/USING	Process a model using a specified interface file
	RUN/USING	Solve a model using a specified solution procedure
	STORE	Store results of model
Output	LIST/USING	Create a user report using a specified user profile and interface file
Verification	MONITOR	Check Assumptions
	VALIDATE/BY	Specify validation procedure