NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, tests publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

Canada

AN EXPERT SYSTEM FOR SIMULATION MODELLING

by

Jimmy Chi-Ming Tam

A thesis
presented to the University of Ottawa
in fulfillment of the
thesis requirement for the degree of
Master of Computer Science

in

Department of Computer Science
University of Ottawa
Ottawa, Ontario, Canada.
September, 1988

ISBN 0-315-46862-9

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

University of Ottawa requires the signature of all persons using or photocopying this thesis. Please sign below, and give address and date.

ABSTRACT

A Modelling Advisor and certifier for the simulation specification language Gest has been developed on a Sun Workstation. The system called Magest (Modelling Advisor for **Gest** programs) uses knowledge of the Gest modelling formálism and the incremental knowledge obtained from the modeller, to guide him to correctly specify models. Modelling functions are supported to specify several types of single models such as continuous models, continuous multi-models, discrete models, discrete multi-models, memoryless models, memoryless multi-models, macro models, as well as interconnected models such as coupled models and multi-rate models. Acceptable models are certified by the Magest system.

Details are provided for the system functions in Magest as well as modelling assistance it provides, and the semantic rules and facts it uses to certify models. To assure a modular, flexible, and robust architecture, the Magest system is conceived as a finite state machine.

ACKNOWLEDGEMENTS

It is a pleasure to acknowledge my gratitude to my research supervisor, Dr. Tuncer I. Ören of the Department of Computer Science. Dr. Ören suggested the system, and his encouragement and friendly guidance in the ways of research made the work a rare educational experience.

I would also like to express my appreciation to others who helped:

To the National Sciences and Engineering Research Council of Canada which provided financial support in the form of research grants;

To all my colleagues and friends for their help and support during the course of this research; and

To my wife Monica for unflagging patience and support.

TABLE OF CONTENTS

ABSTRACT			iv
ACKNOWLEDGEMENTS			v
Chapter	1.	INTRODUCTION	1
Chapter	2.	GEST PROGRAM AND MODEL STRUCTURE	7
Chapter	3.	SYSTEM FUNCTIONS IN MAGEST	20
Chapter	4.	MODELLING ASSISTANCE PROVIDED BY MAGEST	72
Chapter	5.	SEMANTIC FACTS AND RULES USED IN MAGEST	91
Chapter	6.	ARCHITECTURE OF THE MAGEST SYSTEM	100
Chapter	7.	IMPLEMENTATION ISSUES	113
Chapter	8.	CONCLUSIONS AND FURTHER RESEARCH	126
Appendix	A.	FILES OF THE GEST ENVIRONMENT	130
Appendix	B.	PROCEDURES AND FUNCTIONS IN MAGEST	133
Appendix	C.	MESSAGES IN THE MAGEST SYSTEM	151
Appendix	D.	SYSTEM TEMPLATES	167
References			177

DETAILED TABLE OF CONTENTS

ABSTRACT iv

ACKNOWLEDGEMENTS v

Chapter 1. INTRODUCTION 1
1.1 Aim and Structure of the Thesis 1
1.2 Built-in Quality Assurance 4
1.3 Architecture of the Gest Environment 5

Chapter 2. GEST PROGRAM AND MODEL STRUCTURE 7
2.1 Introduction 7
2.2 Gest Program Structure 8
2.3 Model Representation in Gest 10
2.3.1 Static Structure 12
2.3.2 Dynamic Structure 14
2.3.3 Coupled Model 16
2.3.4 Multi-rate Model 18

Chapter 3. SYSTEM FUNCTIONS IN MAGEST 20
3.1 Introduction to System Functions in Magest 20
3.1.1 Entering the Gest Environment 23
3.1.2 Gest Executive 24
3.2 Major Functions of Magest 26
3.2.1 Single Model Environment 26
3.2.2 Coupled Model Environment 29
3.2.3 Multi-Rate Model Environment 32
3.3 Magest I/O 34
3.3.1 Introduction to the Magest I/O 34
3.3.2 Save 35
3.3.3 Help 36
3.3.4 Return 39
3.3.5 Print 40
3.3.6 Clear 42
3.3.7 Load 43

3.4	Prepare Coupling	45
	3.4.1 Introduction to Prepare Coupling	45
	3.4.2 External Coupling	46
	3.4.3 Internal Coupling	49
3.5	Detailed Functions in Magest	53
	3.5.1 Certification	54
	3.5.2 ShellTool	57
	3.5.3 Reference File	58
	3.5.4 Symbol Table	60
	3.5.5 Reserved Words	61
	3.5.6 Options	62
	3.5.7 Walking menu in the Text Editor	64
	3.5.8 Insert Type of Models	71
Chapter 4.	MODELLING ASSISTANCE PROVIDED BY MAGEST	72
4.1	Single Model Specification	73
4.2	Coupled Model Specification	85
4.3	Multi-rate Model Specification	87
Chapter 5.	SEMANTIC FACTS AND RULES USED IN MAGEST	91
5.1	Introduction to Semantic Facts and Rules	91
5.2	Facts and Rules in General	93
5.3	Facts and Rules Relating to the Static Structure of Models	94
5.4	Facts and Rules Relating to the Dynamic Structure of Models	96
5.5	Facts and Rules Relating to Coupling	99
Chapter 6.	ARCHITECTURE OF THE MAGEST SYSTEM	100
Chapter 7.	IMPLEMENTATION ISSUES	113
7.1	Introduction to the Implementation of Magest	113
7.2	Knowledge Representation in Magest	117
7.3	Window Structure and Management	120
7.4	Limitations of Magest	124
Chapter 8.	CONCLUSIONS AND FURTHER RESEARCH	126

Appendix A. FILES OF THE GEST ENVIRONMENT 130
A.1 Object File of the Gest Environment 130
A.2 Implementation Files 130
A.3 Documentation Tools 131
A.4 Template Files 131
A.5 Help Files 132

Appendix B. PROCEDURES AND FUNCTIONS IN MAGEST 133

Appendix C. MESSAGES IN THE MAGEST SYSTEM 151
C.1 Messages for pop up windows 152
C.2 System error messages 153
C.3 Syntactic and Semantic error messages 155

Appendix D. SYSTEM TEMPLATES 167
D.1 Template for continuous Models 168
D.2 Template for continuous multi-models 169
D.3 Template for discrete models 170
D.4 Template for discrete multi-models 171
D.5 Template for memoryless models 172
D.6 Template for memoryless multi-models 173
D.7 Template for macro models 174
D.8 Template for coupled models 175
D.9 Template for multi-rate models 176

References 177

LIST OF FIGURES

Figure 1.3.1	Elements of the Gest Environment	6
Figure 2.2.1	A Gest program specification	9
Figure 2.3.1	A Gest component model specification	11
Figure 2.3.2	A coupled model specification	17
Figure 2.3.3	A multi-rate model specification	19
Figure 3.1.1	Elements of Magest System	21
Figure 3.1.2	Type gest to enter the Gest Environment	23
Figure 3.1.3	Gest Executive - Main Menu	25
Figure 3.2.1	A pop up window appears on the center of the screen, for single model environment entry	26
Figure 3.2.2	Environment for single model	28
Figure 3.2.3	A pop up window appears on the center of the screen for coupled model environment entry	30
Figure 3.2.4	Environment for coupled model	31
Figure 3.2.5	A pop up window appears on the center of the screen for multi-rate model environment entry	32
Figure 3.2.6	Environment for multi-rate model	33
Figure 3.3.1	The Magest I/O	34
Figure 3.3.2	Submenu of Help function	37
Figure 3.3.3	Help pop up window with pop up menu	38
Figure 3.3.4	Submenu of Return function	39
Figure 3.3.5	Print file menu	41
Figure 3.3.6	Confirm save window for clear function	42
Figure 3.3.7	A pop up window to get information of load file	44
Figure 3.3.8	Message to suggest the text editor window clear before loading another file	44
Figure 3.4.1	Confirmation windows to delete old blocks for External Coupling function	46
Figure 3.4.2	Unfinish coupled Model A	47
Figure 3.4.3	Continuous model "B" and discrete model "C"	47
Figure 3.4.4	A "MacroModels" block and a template of "ExternalCouplings" for the coupled model "A"	48
Figure 3.4.5	Confirmation windows to delete the old block for the Internal Coupling function	49
Figure 3.4.6	Unfinish coupled Model A	50
Figure 3.4.7	A template of "InternalCouplings" for the coupled model "A"	51

Figure 3.4.8	Internal couplings for the coupled model "A"	52
Figure 3.5.1	Certification functions of the environment for single models	55
Figure 3.5.2	Only Static Structure is certified	55
Figure 3.5.3	Certification function of the environment for coupled models	55
Figure 3.5.4	Certification function of the environment for multi-rate models	55
Figure 3.5.5	The ShellTool subwindow	57
Figure 3.5.6	Once the Reference File button is activated, a pop up window appears	59
Figure 3.5.7	Reference file subwindow	59
Figure 3.5.8	Symbol table subwindow	60
Figure 3.5.9	Submenu of reserved words function	61
Figure 3.5.10	Reserved words window	61
Figure 3.5.11	Options in Magest	63
Figure 3.5.12	Walking menu for the single model text editor window	64
Figure 3.5.13	Walking menu for the coupled model and multi-rate model text editor window	64
Figure 3.5.14	Submenu of Find	66
Figure 3.5.15	Submenu of Insert line	66
Figure 3.5.16	Submenu of Static Structure for continuous models, continuous multi-models, discrete models and discrete multi-models	66
Figure 3.5.17	Submenu of Static Structure for memoryless models, memoryless multi-models and macro models	67
Figure 3.5.18	Submenu of Dynamic Structure for continuous models	67
Figure 3.5.19	Submenu of Dynamic Structure for continuous multi-models	67
Figure 3.5.20	Submenu of Dynamic Structure for discrete models	68
Figure 3.5.21	Submenu of Dynamic Structure for discrete multi-models	68
Figure 3.5.22	Submenu of Dynamic Structure for memoryless models and macro models	68
Figure 3.5.23	Submenu of Dynamic Structure for memoryless multi-models	69
Figure 3.5.24	Submenu of Dynamic Structure for macro models	69
Figure 3.5.25	Submenu of Selection block	69
Figure 3.5.26	Submenu of Repetition block	70

Figure 3.5.27	Insert Type of Models functions	71
Figure 4.1.1	Rough version of continuous model "Motor"	73
Figure 4.1.2	From Unix to Suntool	74
Figure 4.1.3	Type gest to enter the Gest Environment	75
Figure 4.1.4	Gest Environment	76
Figure 4.1.5	A pop up window appears on the center of the screen, for single model environment entry	77
Figure 4.1.6	Environment for single model	78
Figure 4.1.7	Deleting a block in the text editor window	79
Figure 4.1.8	Typing in the text editor window	80
Figure 4.1.9	Error occurs in environment of single model	81
Figure 4.1.10	Semantic error occurred in continuous model	83
Figure 4.1.11	Enhanced version of continuous model "Motor"	84
Figure 4.2.1	Environment of a coupled models	86
Figure 4.3.1	Multi-rate model "ElectronicOscillator"	88
Figure 4.3.2	ContinuousModel "Fast"	89
Figure 4.3.3	ContinuousModel "Slow"	90
Figure 6.1	Gest Environment: Overall System Architecture	101
Figure 6.2	Magest Environment for Single Model: System Architecture	102
Figure 6.3	Magest Environment for Coupled Model: System Architecture	103
Figure 6.4	Magest Environment for Multi-Rate Model: System Architecture	104
Figure 6.5	Magest Text Editor with the System Architecture	105
Figure 6.6	Find and Insert line within the System Architecture	106
Figure 6.7	Static Structure within the System Architecture	107
Figure 6.8	Dynamic Structure within the System Architecture	108
Figure 6.9	Selection block and Repetition block within the System Architecture	109
Figure 6.10	Magest Support Windows within the System Architecture	110
Figure 6.11	File Information within the System Architecture	111
Figure 6.12	Magest I/O within the System Architecture	112
Figure 7.1.1	Anatomy of the Magest System	115
Figure 7.2.1	Algorithm to implement semantic rule MD.14 of chapter 5	119
Figure 7.3.1	Main window structure in Magest	121
Figure 7.3.2	Structure of pop up windows	123

LIST OF TABLES

Table 2.1	Static structure declarations	13
Table 2.2	Dynamic structure declarations	15
Table 3.1	Description of Gest Executive - Main Menu	24
Table 3.2	The contains of pop up frame menu in the help subwindow	36
Table 3.3	Description of print file subwindow	40
Table 3.4	Options in Magest	62
Table 3.5	Functions in walking menu	65
Table 5.1	Division codes of semantic facts and rules	91
Table 7.1	Description of an item in symbol table	118
Table 7.2	Subframe of Magest frame	120
Table 7.3	Constant limit of Magest	125
Table A.1	Object file of the Gest Environment	130
Table A.2	Implementation files of the Magest system	130
Table A.3	Documentation tools of the Magest system	131
Table A.4	Template files of the Magest system	131
Table A.5	Help files of the Magest system	132
Table B.1	Magest source files and details	133
Table D.1	The name and filename of templates	167

Chapter 1

INTRODUCTION

1.1 Aim and Structure of the Thesis

"An expert system is a computing system capable of representing and reasoning about some knowledge-rich domain, such as internal medicine or geology, with a view to solving problems and giving advise" (Jackson 1985 p.1). Magest (Modelling Advisor and certifier for GEST programs) is an expert system to construct Gest models. Magest provides guidance to specify models, and certifies models if they are correctly specified. An early phototype was reported in the literature (Ören and Aytac 1985, Aytac and Ören 1986).

Gest (GEneral System Theory implementor) is the first modelling and simulation language which provides modular model and simulation specification facilities (Ören 1984). There are basically two categories of models in Gest: single models and interconnected models. Single models can be continuous models, continuous multi-models, discrete models, discrete multi-models, memoryless models, memoryless multi-models and macro models. Interconnected models are coupled models and multi-rate models. Representation of models are to be discussed in chapter two.

Magest uses two types of knowledge in assisting the modeller. The first type of knowledge is based on the methodology of the Gest language and consists of syntactic and semantic rules and facts. About sixty of the semantic rules and facts that Magest uses are discussed in Chapter five.

The second type of knowledge that Magest uses is the incremental knowledge obtained from the user. Any knowledge is scrutinized by Magest before being accepted.

This thesis consists of three parts. The first part consists of chapters one and two. It provides an introduction to built-in quality assurance, the architecture of the Gest environment and an introduction to the Gest model structure.

The second part consists of chapters three and four. Chapters three provide detailed explanations of system functions. In chapter four, we provide a complete tutorial of how a model would be constructed with assistance provided by Magest.

The last part of the thesis consists of chapters five to eight. It presents the internal details of Magest system. Chapter five discusses the knowledge used in Magest, i.e., the semantic facts and rules. Chapter six explains the architecture of the Gest environment and the Magest system. A series of finite state machines are used to represent the architecture of the system. Chapter seven discusses the design phase of the Magest system and the limitations of the system. Finally, chapter eight discusses the conclusions and further studies that would enhance the existing implementation.

The contribution of this thesis consists of the design of the system architecture of the Gest environment, and the design, implementation and documentation for the modelling part of the Magest system. These include the design and organization of the architecture, data structures,

windows, syntax analyzers, semantic analyzers and program documentations. In addition, some semantic facts and rules are defined and all of them are implemented.

The environment to specify the parameters and experimentations are being developed as a separate study by another graduate student. A translator to translate Gest programs into equivalent C programs is at its last stage of implementation (Ye and Ören 1988).

A presentation of the abilities of this current implementation has been done by Ören and Tam (1988).

1.2 Built-in Quality Assurance

Quality assurance in modelling and simulation has been well elaborated on in Balci (1987), Banks et al. (1987), Ören (1981, 1987a), and Ören et al. (1985). Built in quality assurance is the use of knowledge-based techniques to guide the user in the specification phase of activities or elements for the purpose of some types of errors such as morphological errors and semantic errors (Ören 1987b).

In order to realize built-in quality assurance, the system must have a knowledge-base and have a detailed data structure to store the incremental knowledge. The principal difference between a knowledge-base and a traditional data structure is that the description and relationship information has much more semantic contents (Symonds, 1986). Another component of a knowledge-base includes the semantic rules that represent computational relationships among facts.

The current knowledge-base of the Magest system is based on about sixty semantic facts and rules. Details of semantic facts and rules are described in chapter 5. These semantic facts and rules are used to guide the modeller in the specification of Gest models.

Knowledge about a model, provided by the modeller is scrutinized by Magest according to the semantic facts and rules and becomes part of its incremental knowledge. Any additional model knowledge provided by the user is tested by Magest with respect to the semantic facts and rules as well as the already accepted incremental knowledge. This process provides built-in quality assurance ability in the specification of models in Gest.

1.3 Architecture of the Gest Environment

The Gest environment is a software package to specify and execute Gest programs. It is implemented in C language, under the UNIX 4.2 operating system, on a Sun workstation. As depicted in Figure 1.3.1 (Barrett 1988), the elements of the Gest environment consists of the following main components:

1) The Gest executive
2) The Magest system
3) The Gest translator
4) The run-time library
5) Simulation control

The Gest executive is the top level of the Gest environment. It provides the user/system interface.

The Magest system provides assistance in the construction of Gest programs, it also certifies Gest programs. The translator translates the Gest program into an interactive procedural program expressed in the C language. This program is then compiled by the the C compiler to allow for simulation execution.

The system discussed in this thesis consists of the Gest executive and the Magest model specification environment. More details of the system functions are described in chapter three.

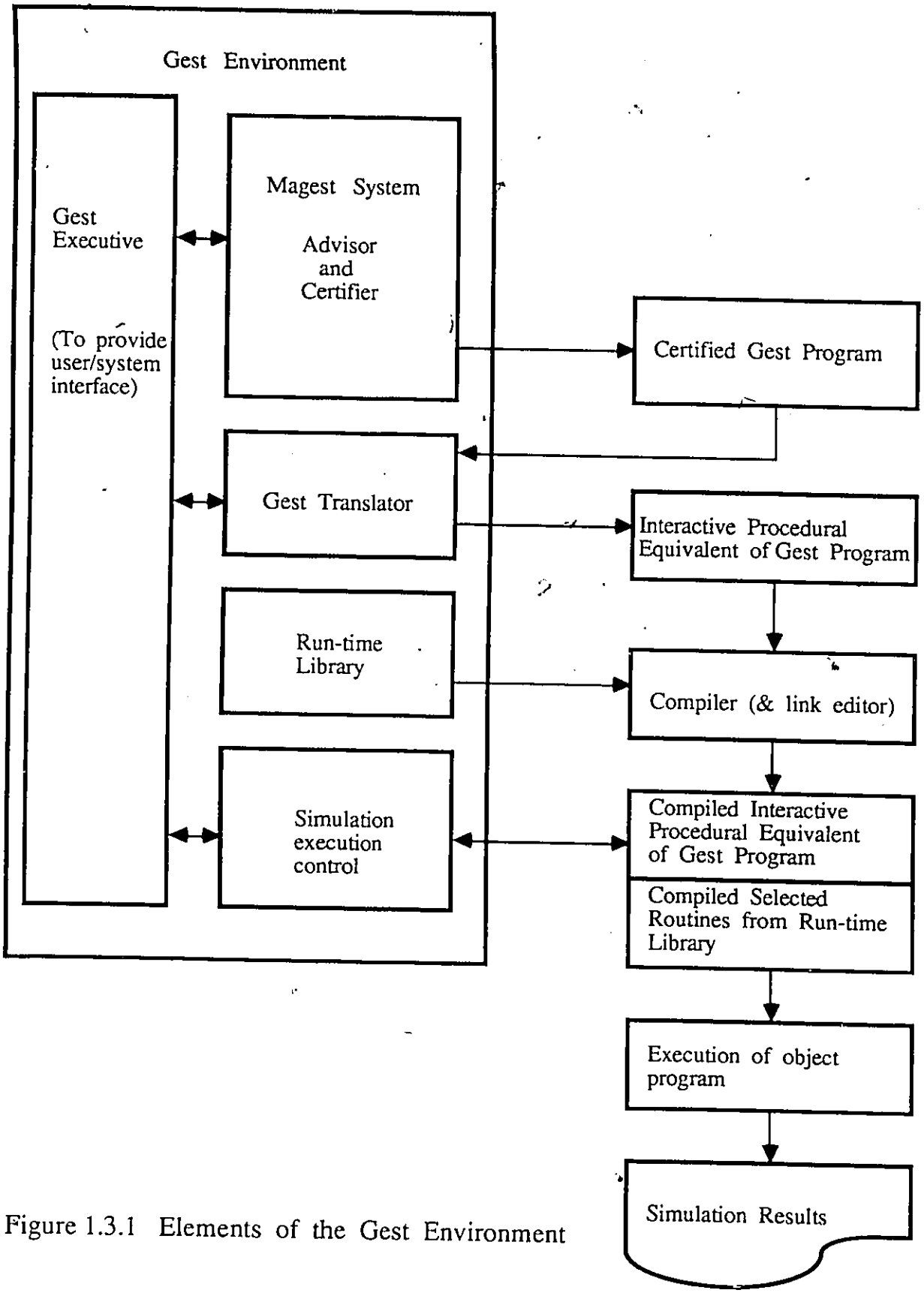

Figure 1.3.1 Elements of the Gest Environment

Chapter 2

GEST PROGRAM AND MODEL STRUCTURE

2.1 Introduction

GEST (GEneral System Theory implementor) is a high-level model and simulation experimentation specification language based on general system theoretic concepts (Ören 1984). A model is a description of a system. The modelling world view of Gest is based on the axiomatic system theory of Wymore (Wymore 1967). The first version of Gest was published in 1971 (Ören 1971). It is the first model-based simulation language which departed radically from other simulation languages. In Gest, specifications of the model, parameter, experiment, run specification and output are totally separated.

2.2 Gest Program Structure

A Gest program consists of five distinct parts:

1) The mathematical model
2) The parameter set (Optional)
3) The experiment specification
4) The run specification
5) The output module

The mathematical model may be any one of a number of different simulation models and consists of the simulation specification. The parameter set which could be optional from a Gest program allows us to specify the run time values for input variables. The experiment specification provides information about experiment conditions which have to be applied to a model. The run specification provides information for controlling a Gest program. Finally, the output module is the specification of the output program to be used to display the result of the simulation study. Figure 2.2.1 is an example of a Gest program specification.

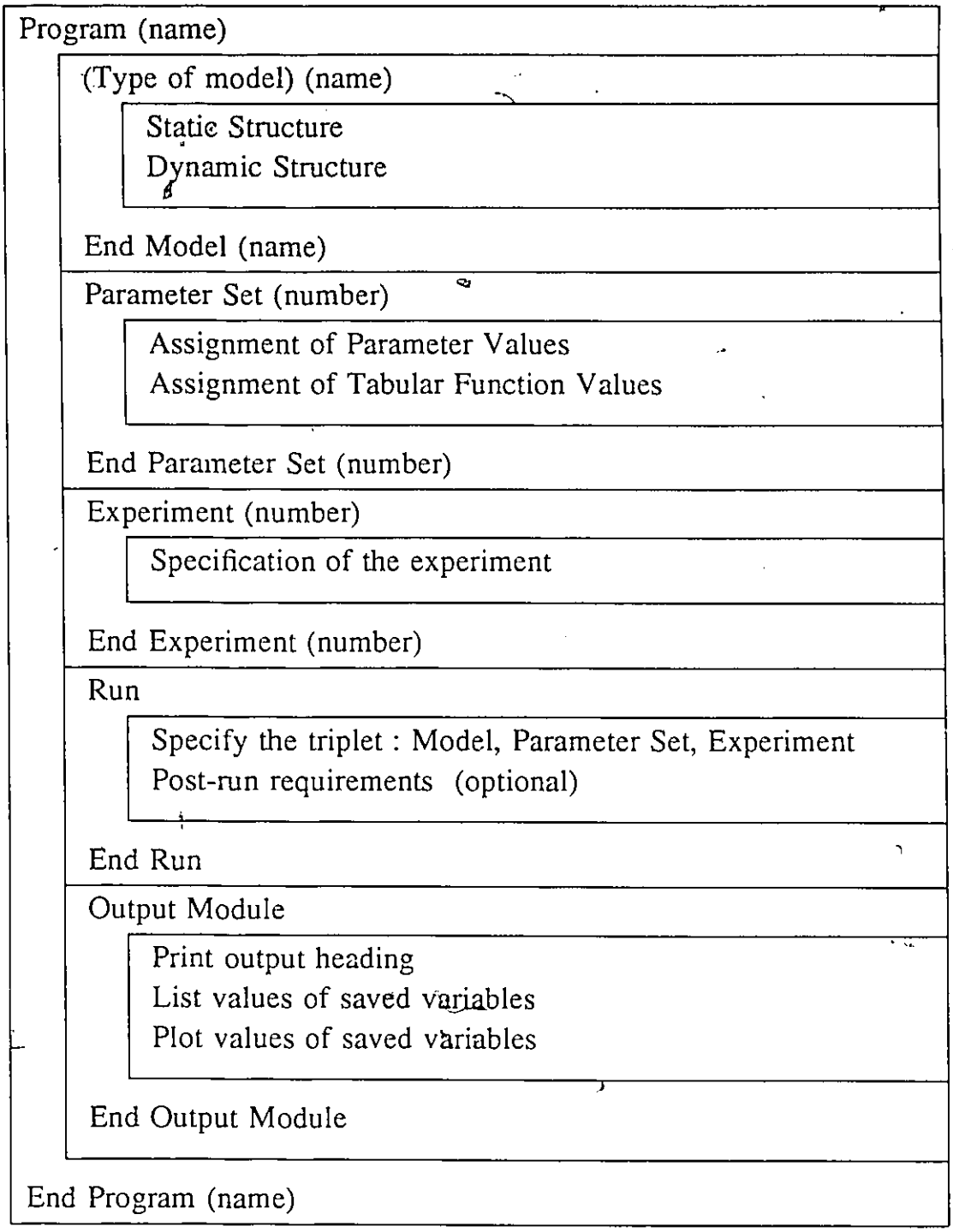

Figure 2.2.1 A Gest program specification.

2.3 Model Representation in Gest

There are basically two categories of models in Gest: single models and interconnected models. Single models can be continuous models, continuous multi-models, discrete models, discrete multi-models, memoryless models, memoryless multi-models or macro models. Interconnected models are coupled models or multi-rate models.

As shown in Figure 2.3.1, the Gest specification of a component model requires the elaboration of two components:

1) Static Structure

2) Dynamic Structure

The details of both static and dynamic structures depend on the type of model. All variables needed in a component model are declared in the static structure section. The dynamic structure contains information needed for behavior generation.

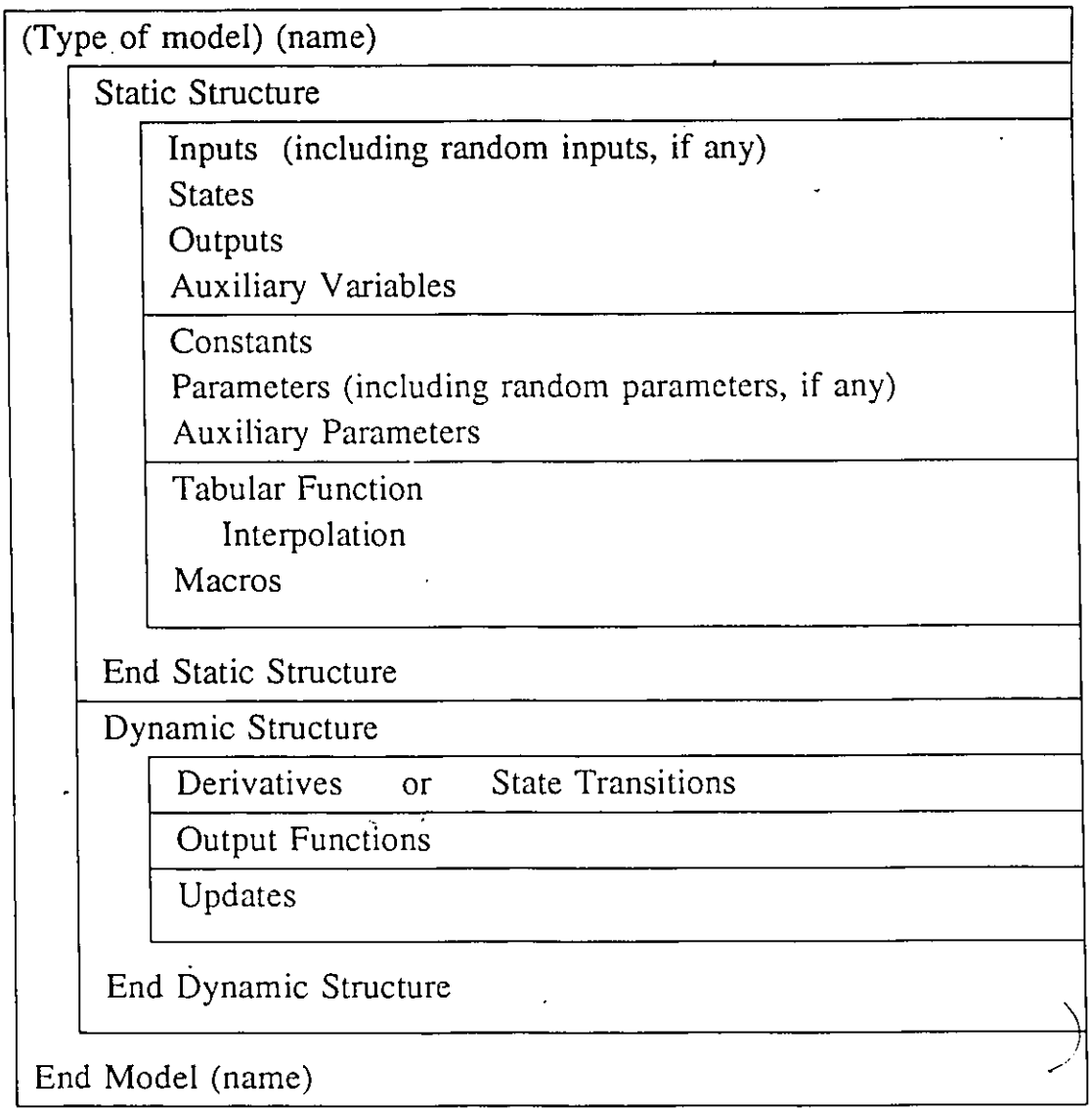

Figure 2.3.1 A Gest component model specification.

2.3.1 Static Structure

The static structure of each component model is used to declare all the descriptive variables such as inputs, states, outputs and auxiliary variables. Input variables are optional. By definition, an autonomous model does not need input to operate. In some models, state variables and auxiliary variables may be considered to be the output variables of the model. In memoryless models, state variables do not exist. In this case, current output is computed based on the values of the current inputs.

In addition, the static structure of a model requires other declarations, such as type and range of values of the descriptive variables of the model as well as constants, parameters, auxiliary parameters, tabular functions, interpolated variables, and macros used.

The Table 2.1 below summarizes the declarations present in the static structure of different type of models.

Type of models	Declarations			
Continuous model	[I]	S	O	[Others]
Continuous multi-model	[I]	S	O	[Others]
Discrete model	[I]	S	O	[Others]
Discrete multi-model	[I]	S	O	[Others]
Memoryless model	I		O	[Others]
Memoryless multi-model	I		O	[Others]
Macro model	I		O	[Others]

I - Input declarations
S - State declarations
O - Output declarations
Others - Auxiliary Variable declarations
Constant declarations
Parameter declarations
Auxiliary Parameter declarations
Tabular Function declarations
Interpolated variable declarations
Macro declarations

Note: Letter within [ ] represents optional declaration

Table 2.1 Static structure declarations.

2.3.2 Dynamic Structure

The dynamic structure contains information needed for behavior generation.

The major elements of dynamic structure includes:

1) A derivative block which contains the specification of derivatives of the state variables along with any computations necessary for auxiliary variables. (The derivative block exists only for continuous models and for continuous multi-models.)

2) A state transition block which contains the specification of the next values of the state variables along with any computations necessary for auxiliary variables. (The state transition block exists only for discrete models and for discrete multi-models.)

3) An output block contains the transformations of the state and/or auxiliary variables into output variables.

4) An optional update block. This block provides a model-oriented specification of any discontinuities that may exist in the model.

5) An optional model selection block. This block may only appear in continuous multi-models, discrete multi-models and memoryless multi-models. This block specifies which dynamic structure is to be selected when there are more than one dynamic structure in multi-models.

Table 2.2 summarizes the blocks present in the dynamic structure of

different types of models.

Type of models	Declarations				
Continuous model	D		[O]	[U]	
Continuous multi-model	D		[O]	[U]	M
Discrete model		S	[O]	[U]	
Discrete multi-model		S	[O]	[U]	M
Memoryless model			O	[U]	
Memoryless multi-model			O	[U]	M
Macro model			O		

D - Derivative block
S - State Transition block
O - Output Function block
U - Update block
M - Model Selection block

Note: Letter within [ ] represents optional declaration.

Table 2.2 Dynamic structure declarations.

2.3.3 Coupled Model

A coupled model consists of more than one component model. As shown in Figure 2.3.2, the coupled model specification consists of:

1) The external variables
2) The component models
3) The macro models
4) The external coupling, and
5) The internal coupling

The **external variable** section contains the declaration of inputs and outputs for the coupled model. For every input or output variable, the range of the acceptable values may also be specified. The list of **component models** declares the type and the name of the component models to be interconnected.

The **macro models** block is a list provided by Magest for documentation purposes. It lists all macro models used in the component models. The **external coupling** block consists of external inputs and external outputs. The external inputs section of the external couplings block specifies the connection of each external input of a the coupled model to the input of a corresponding component model. The external output section specifies the connection of output variables of a coupled model to the outputs of the component model.

The **internal coupling** block specifies the input/output interfaces of the component models.

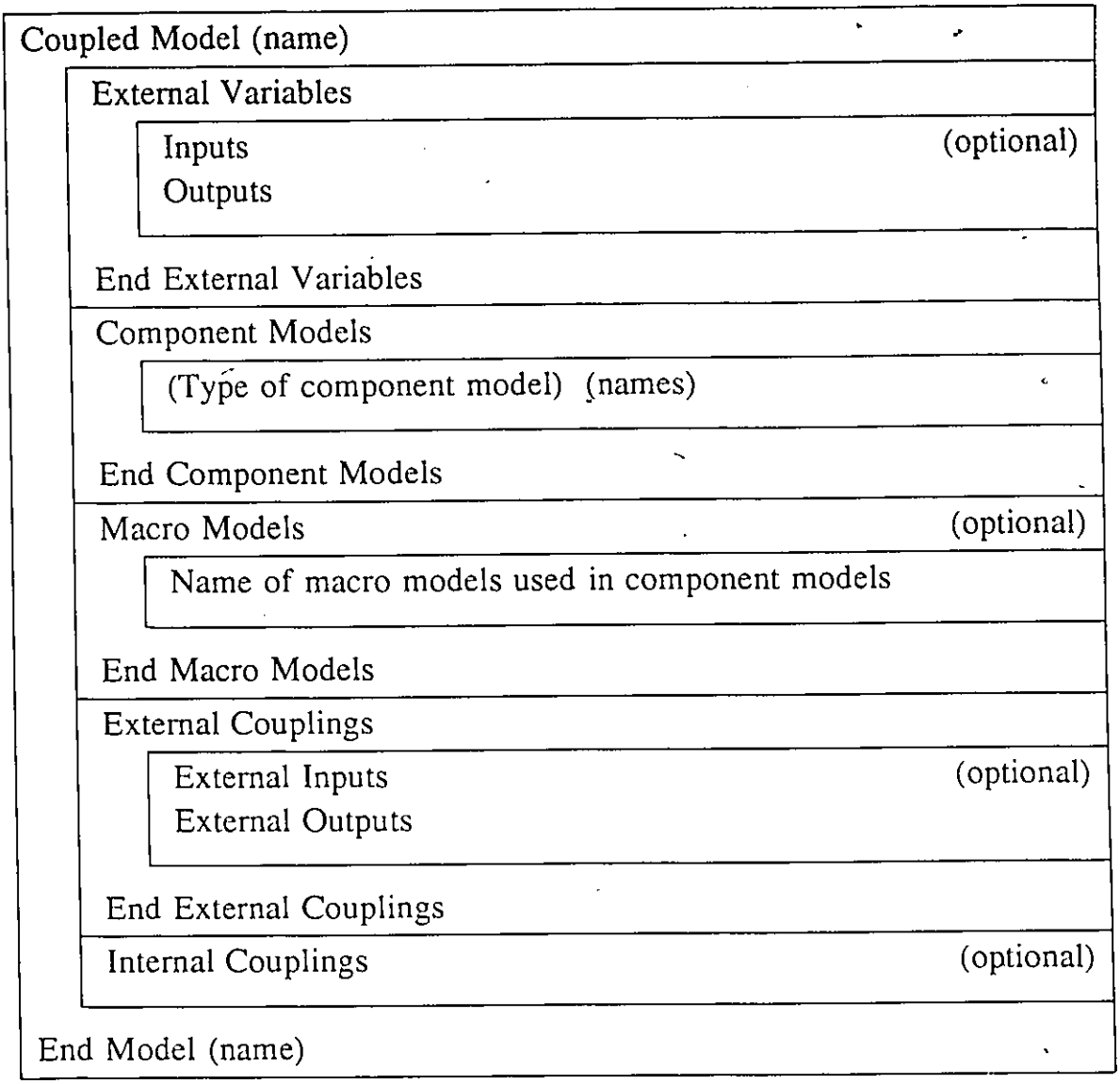

Figure 2.3.2 A coupled model specification.

2.3.4 Multi-rate Model

The specification of multi-rate models is similar to that of coupled models. Figure 2.3.3 shows a multi-rate model specification.

The declaration of component models is the only difference between coupled models and multi-rate models. In multi-rate models, we only have two component models as follows: a fast continuous model and a slow continuous models. Both models are continuous models.

The multi-rate model is a special case of the coupled model, it is provided for the convenience of the user.

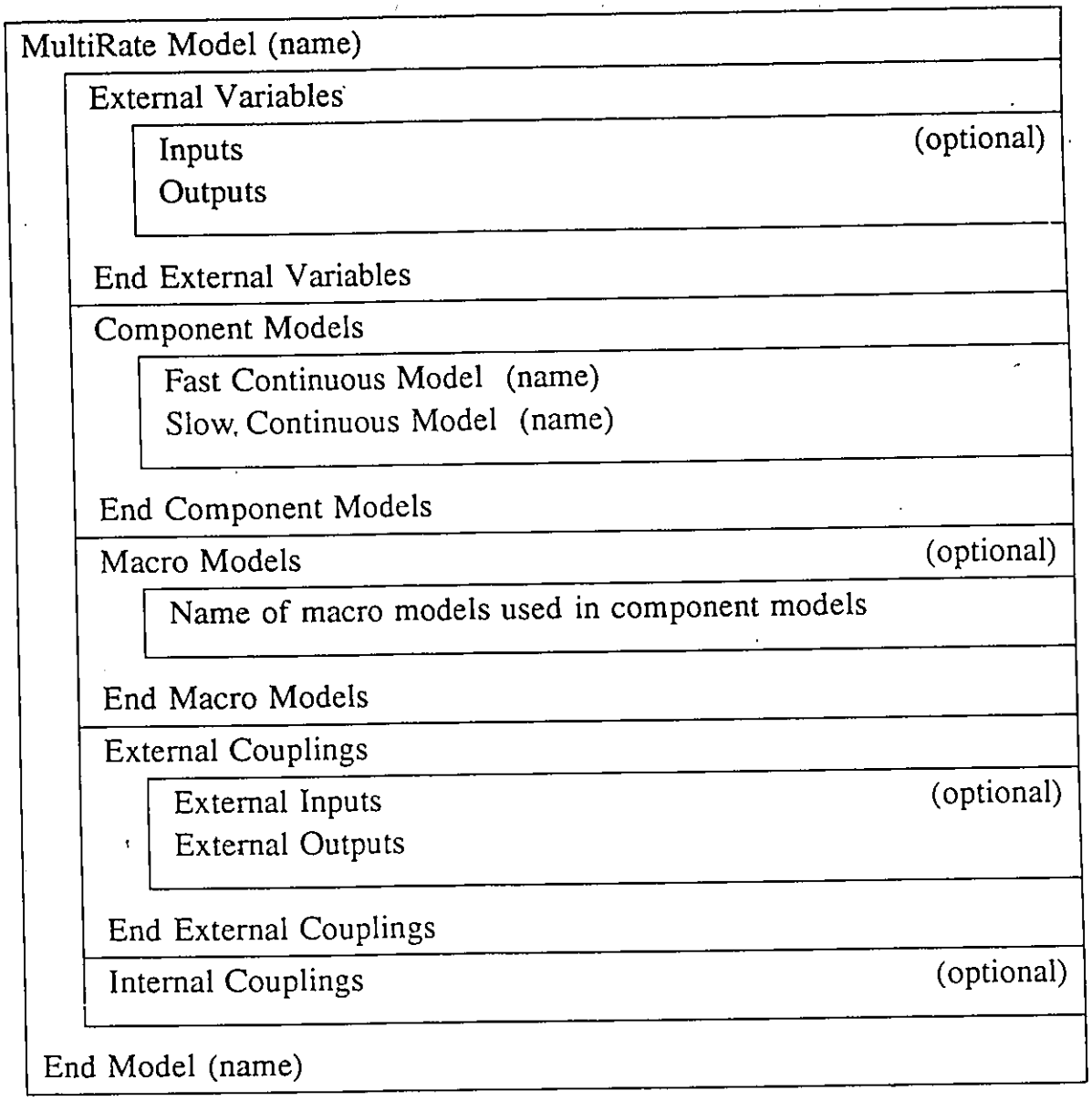

Figure 2.3.3 A multi-rate model specification.

Chapter 3

SYSTEM FUNCTIONS IN MAGEST

3.1 Introduction to System Function in Magest

As depicted in Figure 3.1.1, Magest provides functions for the specification of:

1) Single models
2) Coupled models
3) Multi-rate models
4) Parameters
5) Experimentations
6) Run controls
7) Output modules, and
8) Complete Gest programs

The specification of **single models** is a basic function of Magest. It also provides an environment for building seven component model types: continuous models, continuous multi-models, discrete models, discrete multi-models, memoryless models, memoryless multi-models and macro models (Ören 1987b). In addition, Magest has the ability to certify the component models, i.e., to assure that the component models are specified according to the Gest formalism.

The specifications of **coupled** and **multi-rate models** are also part of the model specification abilities of Magest. To be able to specify a

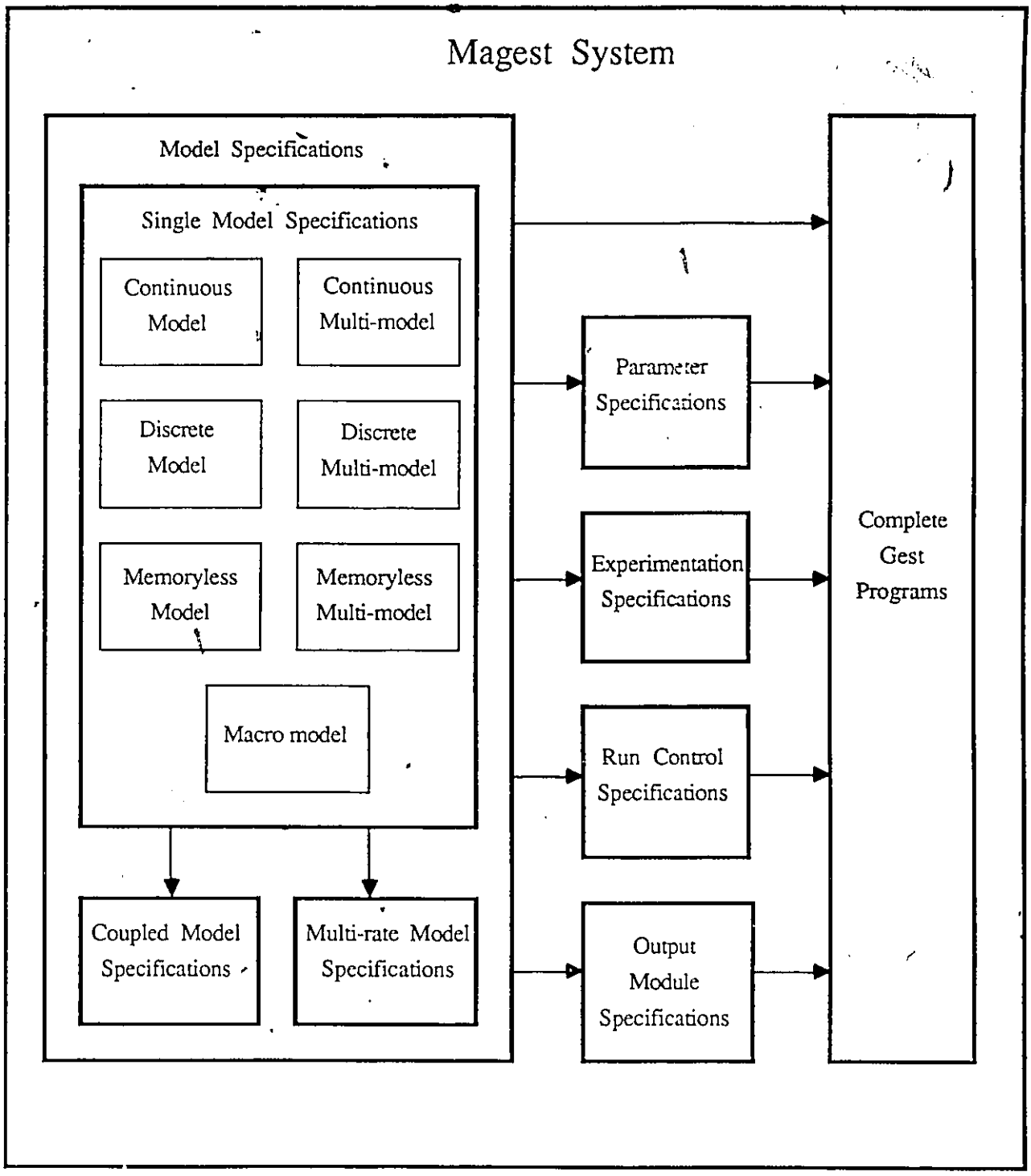

Figure 3.1.1 Elements of the Magest System

coupled or a multi-rate model, the modeller has to use Magest to express all the relevant component models. Then, Magest can be used to complete the relevant additional coupling specification. The modeller could then switch to the Gest program mode, to combines all of the specifications into a single Gest program. However, the computer aided parameter and experimentation specification abilities are beyond the scope of this thesis.

3.1.1 Entering the Gest Environment

To execute the Magest system, the minimum hardware requirement is a SUN 3/50 workstation, with 4 Mbytes of memory. The workstation should be connected to its local area network.

The operating system which is required is UNIX 4.2, with the SunView windows environment (Sun Microsystem 1986a). The modeller can enter the window environment by typing **suntool.** In one of the windows, the modeller types **gest** to enter the Gest environment (see Figure 3.1.2).

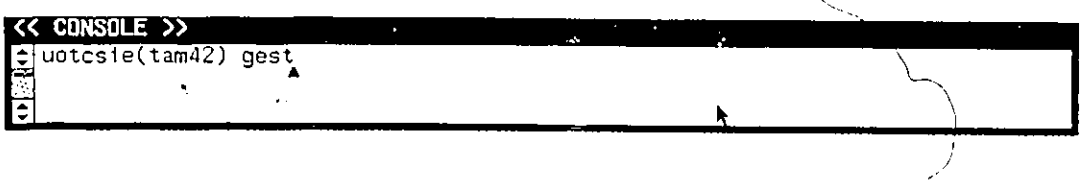

Figure 3.1.2 Type **gest** to enter the Gest Environment.

3.1.2 Gest Executive

As shown in Figure 1.1, the Gest executive is the entry level to the Gest environment. It provides all high level functions for a simulation environment. Figure 3.1.3 shows the screen once we enter the Gest environment. The right hand side contains the main menu of the Gest environment.

Brief descriptions of each of the items on the main menu of the Gest environment are given in Table 3.1.

Button	Description
Single Model	To enter single model specification environment.
Coupled Model	To enter coupled model specification environment.
Multi-Rate Model	To enter multi-rate model specification environment.
Parameter	To enter parameter specification environment.
Experiment	To enter experiment specification environment.
Run Control	To enter run control specification environment.
Output Module	To enter output module specification environment.
Gest Program	To enter gest program specification environment.
Translate	To translate a Gest program into a C program, and compile it into an executable file.
Simulation	To perform simulation study.
Help	To provide help facilities.
ShellTool	To provide a Unix emulator subwindow.
Quit	To leave Gest environment and return to the Unix environment.

Table 3.1 Description of Gest Executive - Main Menu.

As shown in Figure 3.1.3, the Gest environment was and is being designed and implemented by Gest Research Group, University of Ottawa. Each member in the group is responsible for a different area of research. This thesis provides a complete design and implementation of the Gest executive and half of the Magest system. This half of Magest system includes the modelling environment for single models, coupled models and multi-rate models. The architecture of the system is depicted in chapter 6. This architecture has provisions for additions to the system.

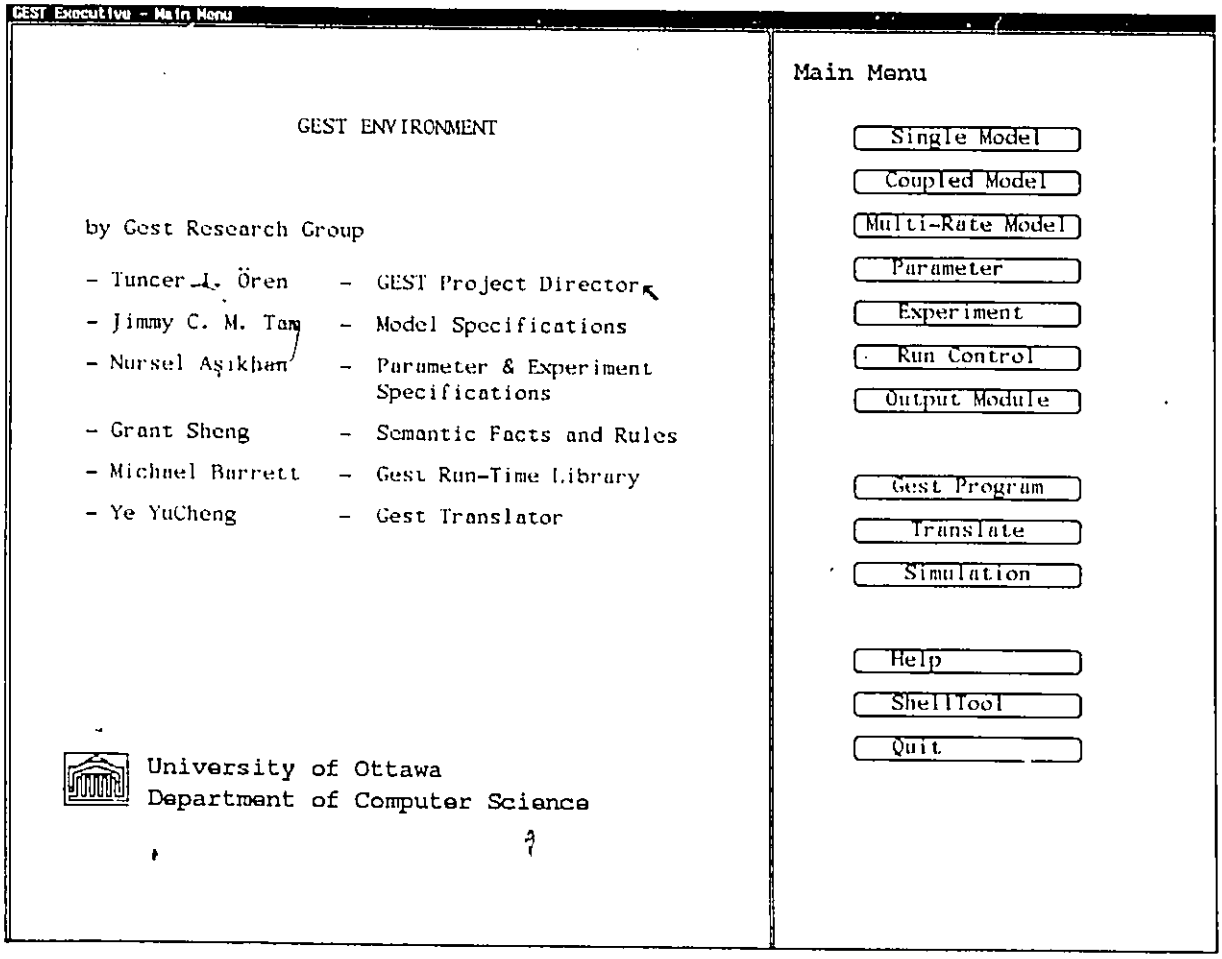

Figure 3.1.3 Gest Executive - Main Menu.

3.2 Major Functions of Magest

3.2.1 Single Model Environment

The single model environment provides computer-aided facilities to specify and edit a Gest component model. To enter the single model mode, press the **Single Model** button on the main menu, a pop-up window will appear. As seen in Figure 3.2.1, you should enter and select the following information:

1) The name of the model.

2) The directory should specify where the model is kept. By default, the modeller's current directory is given.

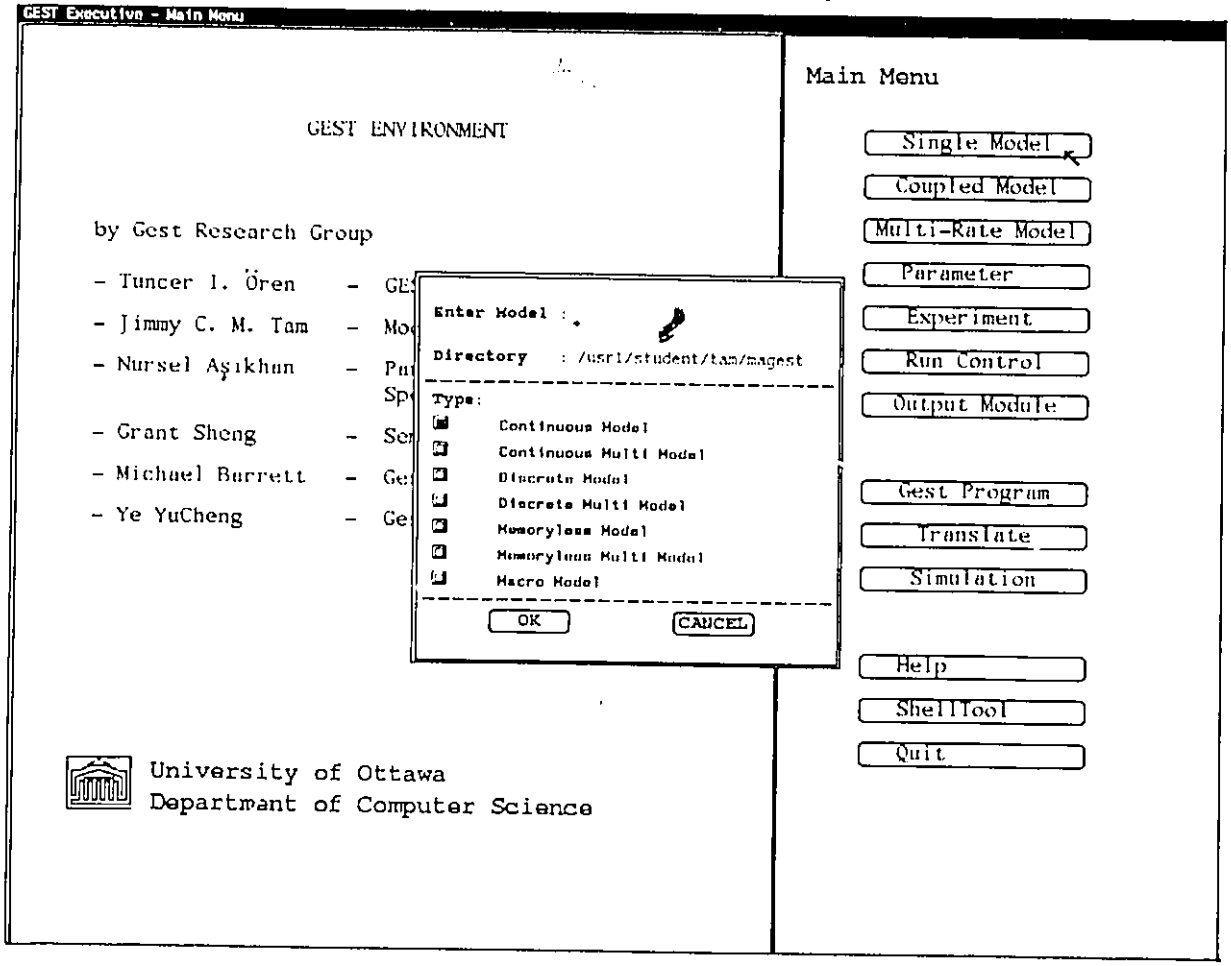

Figure 3.2.1 A pop up window appears on the center of the screen, for single model environment entry.

If the modeller would like to work on an existing model, the directory should specify where the model is kept.

3) Type of model. Select one of the choices to indicate the type of the model. By default, it is a continuous model.

After entering and selecting all of the information above, press the **OK** button.

At this moment, Magest prepares the environment for single model mode. It checks whether or not the given model exists or not. If it does not exist, Magest would provides a dynamic template which corresponds to the type of the model indicated. If the model already exists, then Magest will load the model into the text editor window.

As shown in Figure 3.2.2, the single model mode environment consists of the following components:

1) Text editor window. (On the top left hand side)

2) Message window. (On the bottom left hand side)

3) A panel. (On the right hand side)

The text editor window is used to enter the specification of a component model. Magest provides a template to be filled in and performs syntactic and semantic checks each time the modeller enters information and presses the return key. If the modeller starts modelling from scratch, after pressing the return key, the cursor is positioned at the beginning of the data entry part of the following line. If the modeller wants to edit an existing model, after pressing the return key, the cursor is positioned to the end of the following line. In addition, Magest provides a

walking menu in the text editor window to assist the modeller to specify or edit models. More details are given in section 3.5.7.

The purpose of the message window is to communicate with the modeller. A list of over 250 messages are listed in appendix C. These messages include messages for pop up windows, system error messages, syntactic and semantic error messages.

The panel contains various types of functions and status indications (Sun Microsystem 1986b). Details of panel functions are given in sections 3.3 - 3.5.

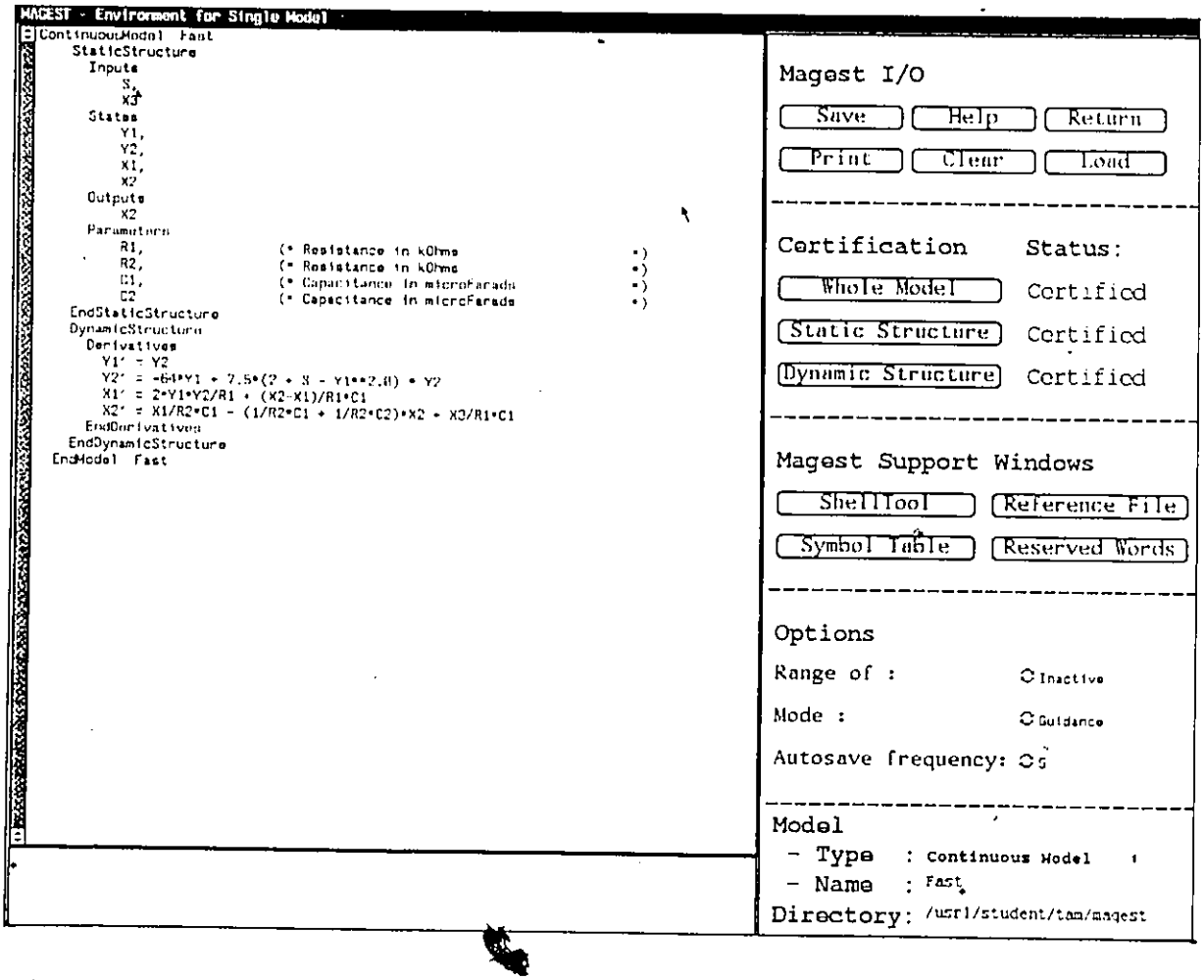

Figure 3.2.2 Environment for single model.

3.2.2 Coupled Model Environment

Coupled model environment provides computer-aided facilities to specify and edit a Gest coupled model. In Gest, a coupled model consists of a set of component models having input/output relationships among themselves and with their environment. In order to specify a coupled model, the modeller must have all the component models already specified and certified.

The advantage of this approach is to make sure that all of the component models within the coupled model are complete. Magest provides templates for "External Coupling" and "Internal Coupling".

To enter the coupled model environment, press the **Coupled Model** button on the main menu. A pop-up window shows up. As seen in Figure 3.2.3, the modeller should enter the following information:

1) The name of the model.
2) The directory specifies the location of the model. By default, the modeller's current directory is given. If the modeller would like to work on an existing model, then the directory is the one where the model has already been kept.

After entering the above information, press the **OK** button.

At this moment, Magest prepares the environment for the coupled model. As in the single model environment, it checks whether or not the given filename exists in the directory. If it does not exist, Magest provides a dynamic template for coupled models. If the file already exists, then Magest loads the file and displays it on the screen. A picture of a

coupled model mode environment is shown on Figure 3.2.4.

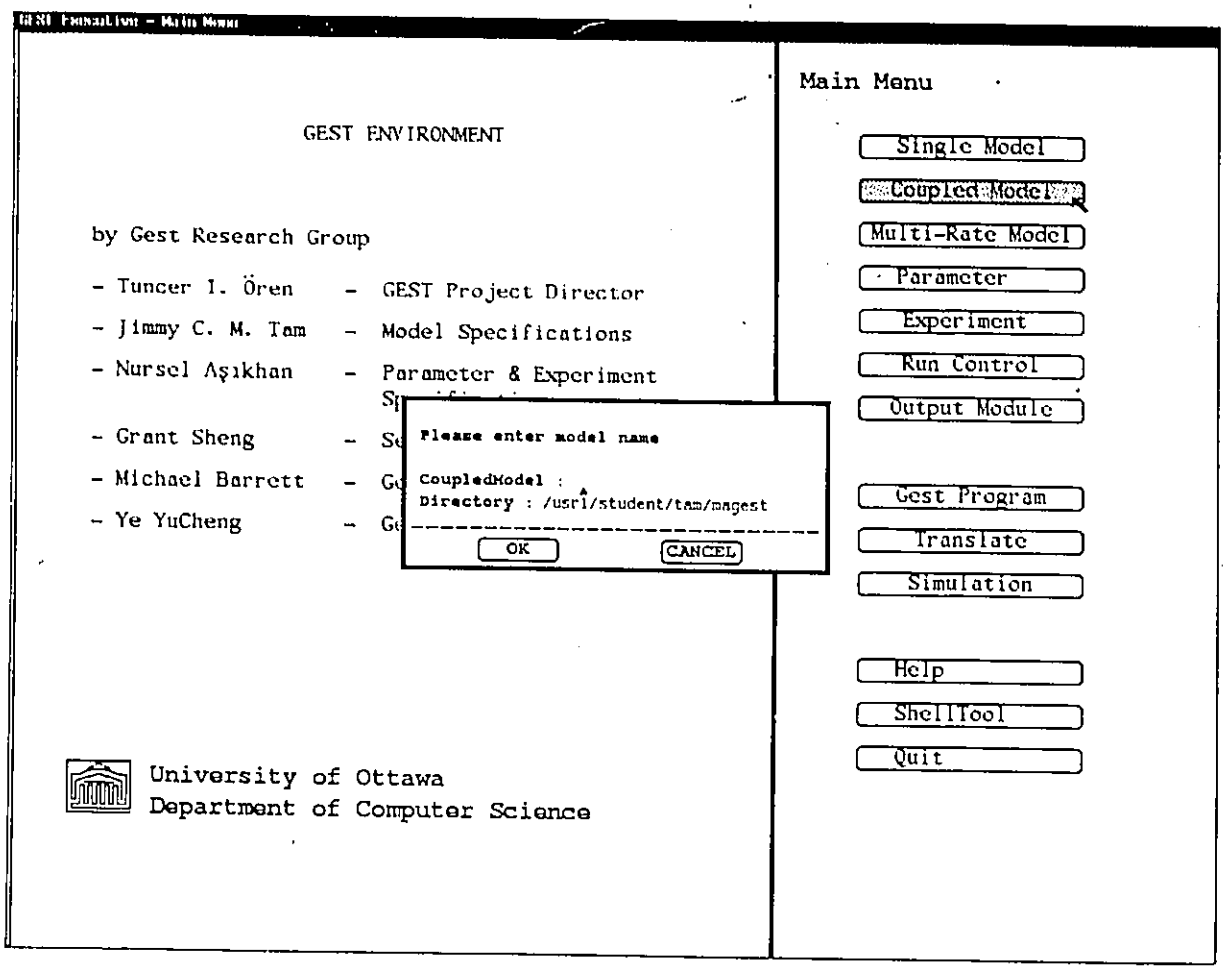

Figure 3.2.3 A pop up window appears on the center of the screen for coupled model environment entry.

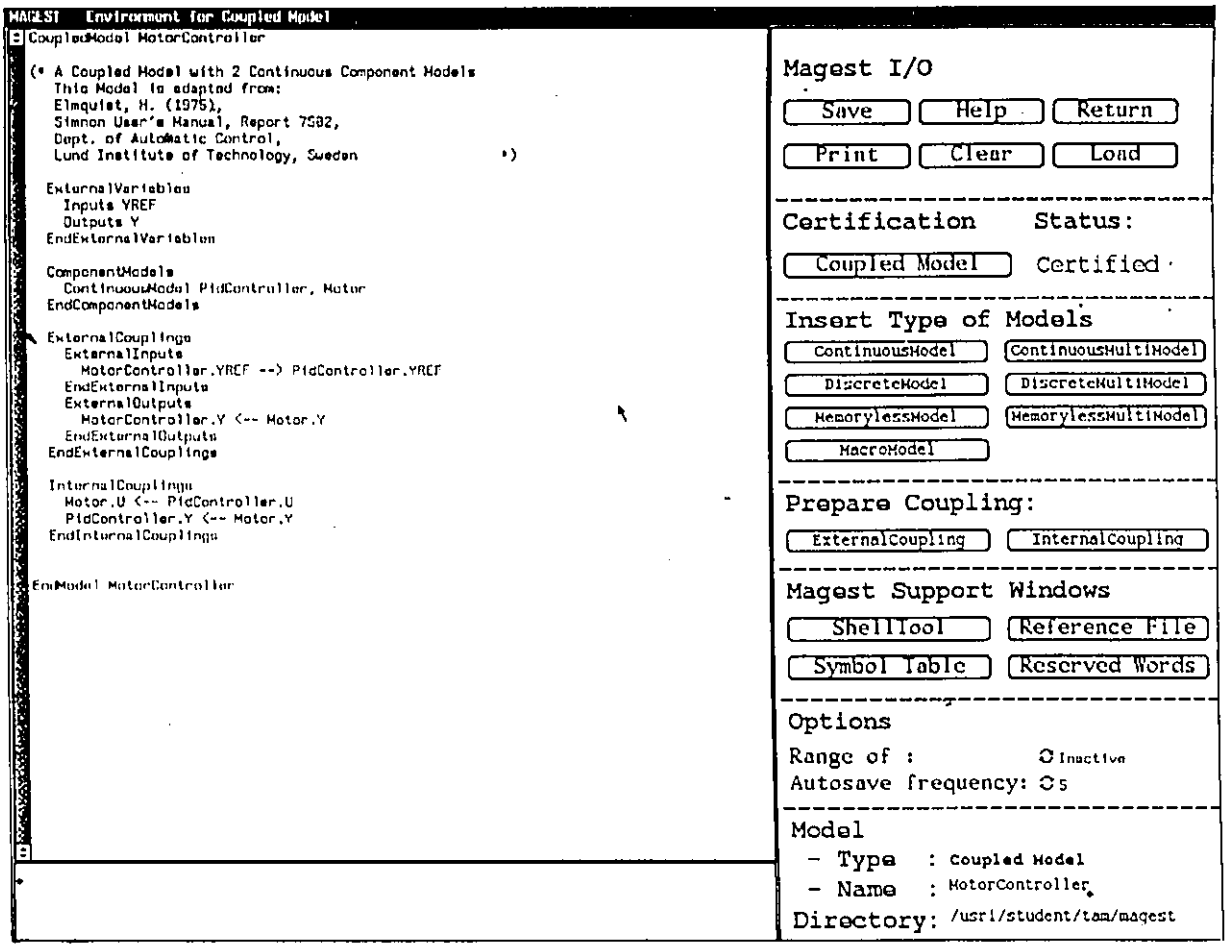

Figure 3.2.4 Environment for coupled model.

3.2.3 Multi-Rate Model Environment

The multi-rate model environment provides computer-aided facilities to specify and edit a Gest multi-rate model. The operations for multi-rate models are similar to the operations for coupled models. Figures 3.2.5 and 3.2.6 depict entering a multi-rate model environment and a multi-rate model environment, respectively.

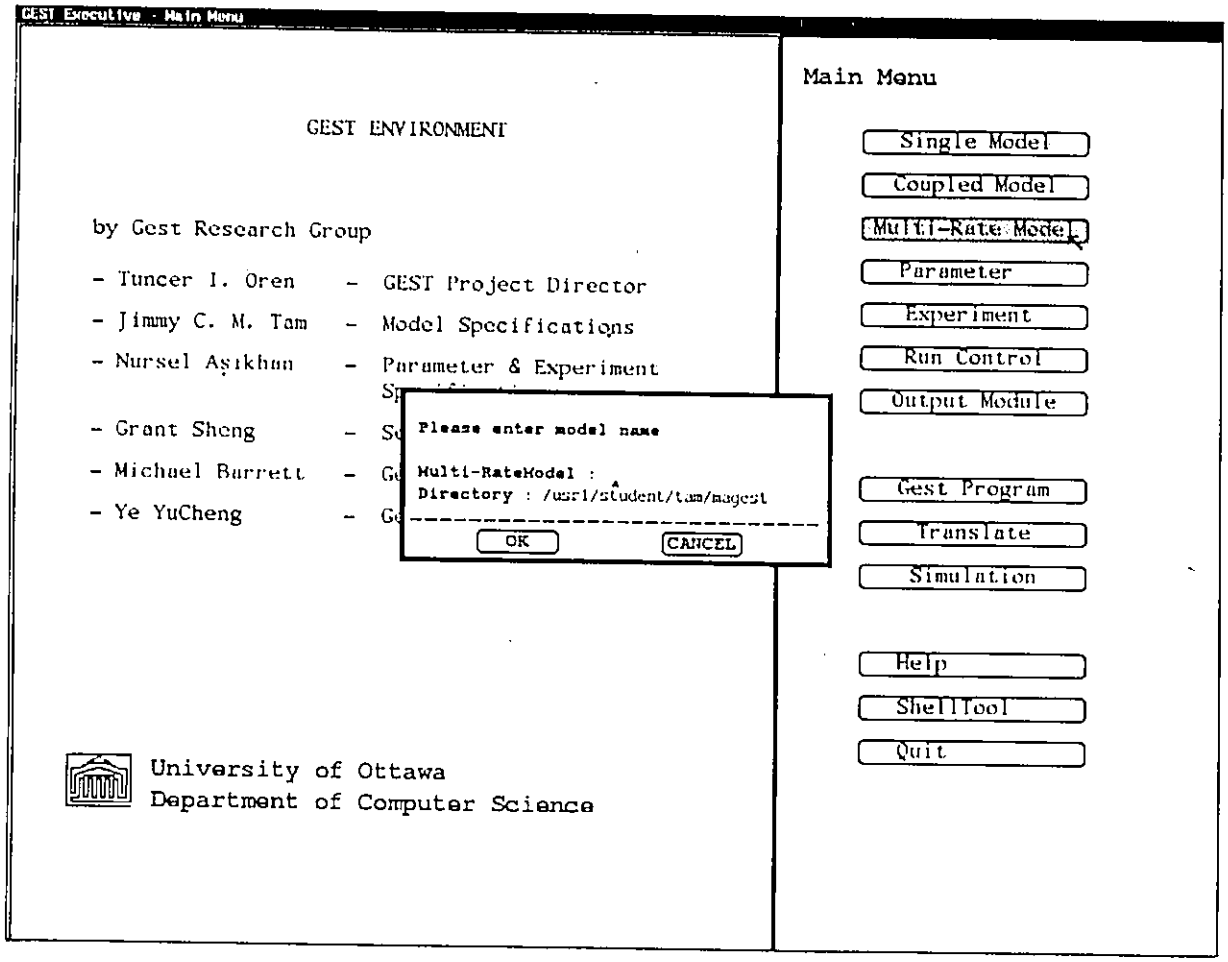

Figure 3.2.5 A pop up window appears on the center of the screen for multi-rate model environment entry.

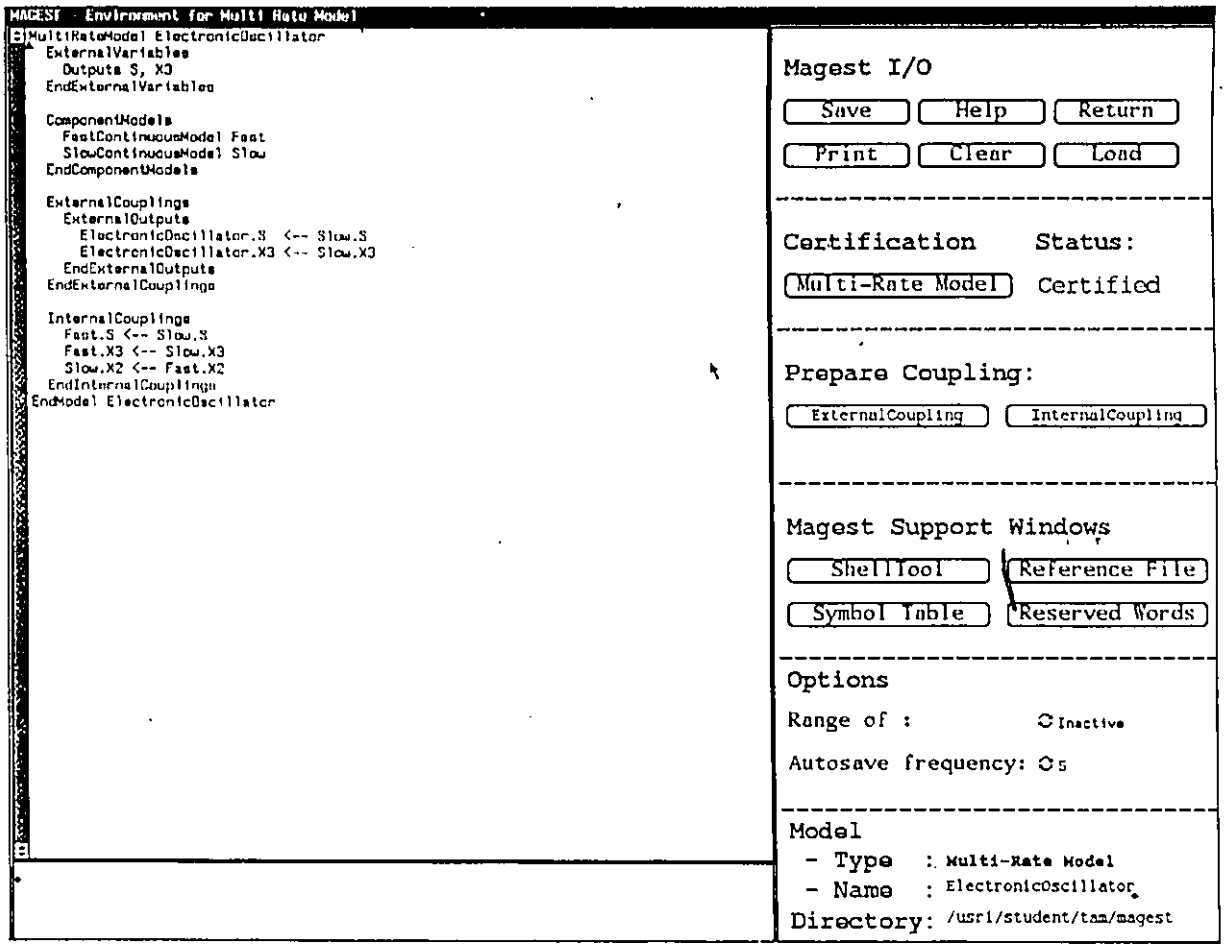

Figure 3.2.6 Environment for multi-rate model.

3.3 Magest I/O

3.3.1 Introduction to the Magest I/O

Magest I/O (Magest Input/Output) provides basic system functions for the modeller. These system functions consist of communications between the modeller and the operating system. They are the most commonly used functions and are put in one group and displayed in the same location as it is the case for single models (Figure 3.2.2), coupled models (Figure 3.2.4), and in multi-rate models (Figure 3.2.6).

As seen in Figure 3.3.1, the functions provided as Magest I/O functions consist of: save, help, return, print, clear and load.

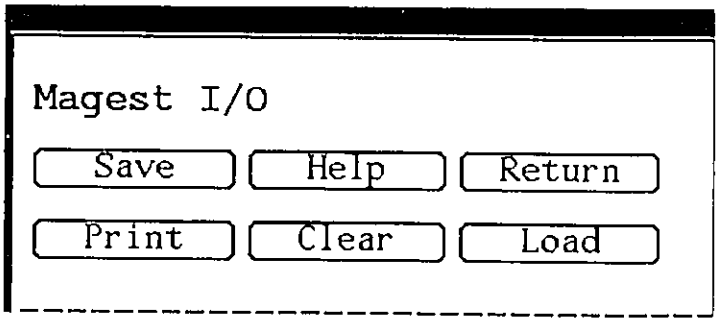

Figure 3.3.1 The Magest I/O

3.3.2 Save

The **Save** function provides the ability to save a copy of file on the text editor window to the disk drive. The filename is on the right bottom corner of the screen, which is the last item of the panel.

Note that the system might perform the save automatically if the number appearing on the autosave option is more than 0. We suggest that the modeller set the autosave option to 5 to ensure an autosave after each five modifications to the text editor window. This will avoid of losing the file if a system crash occurs.

3.3.3 Help

The **Help** function provides the ability to display appropriate sections of the Gest tutorial manual in the help window. Once the **Help** button is activated, a submenu appears. As shown in Figure 3.3.2, this submenu contains the topic of the available reference manuals. After the modeller has made a selection, a subwindow will appear with the appropriate documentation.

As shown in Figure 3.3.3, the pop up menu of the help window contains the following functions:

Function	Description
Done	Reading is finished, close the window.
Move	To move the window.
Resize	To change the size of window.
Redisplay	Redisplay the window.
Print	To print the appropriate section of the Gest tutorial manual (see section 3.3.5 for a description).

Table 3.2 The contents of the pop up frame menu in the help subwindow.

The Help documentation is not fully implemented in the current version of the Magest system. However, the architecture of the system has provision for it. The names of the help files to be provided is given in appendix A.5.

Due to the sun workstation's limitation on windows, the help subwindow is shared with the reference file subwindow. Therefore, when

this subwindow appears, the reference file subwindow will disappear. Similarly, when the reference file subwindow appears, the help subwindow will disappear.

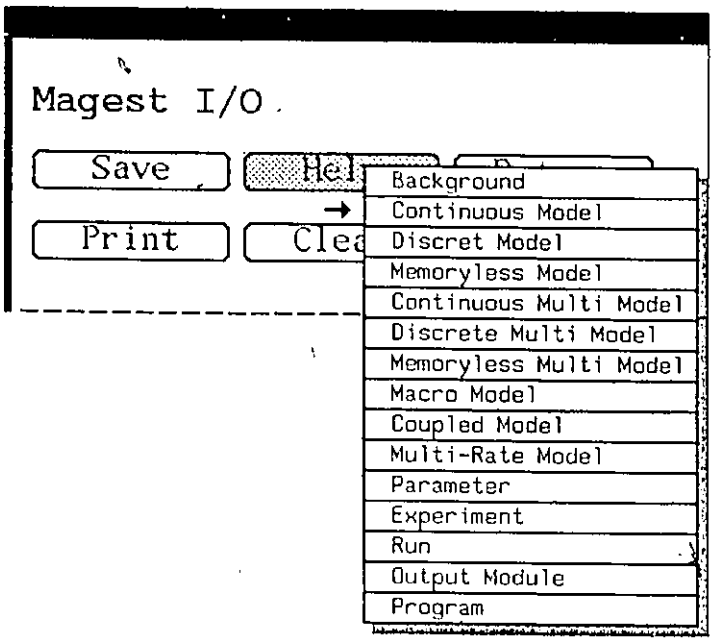

Figure 3.3.2 Submenu of Help function.

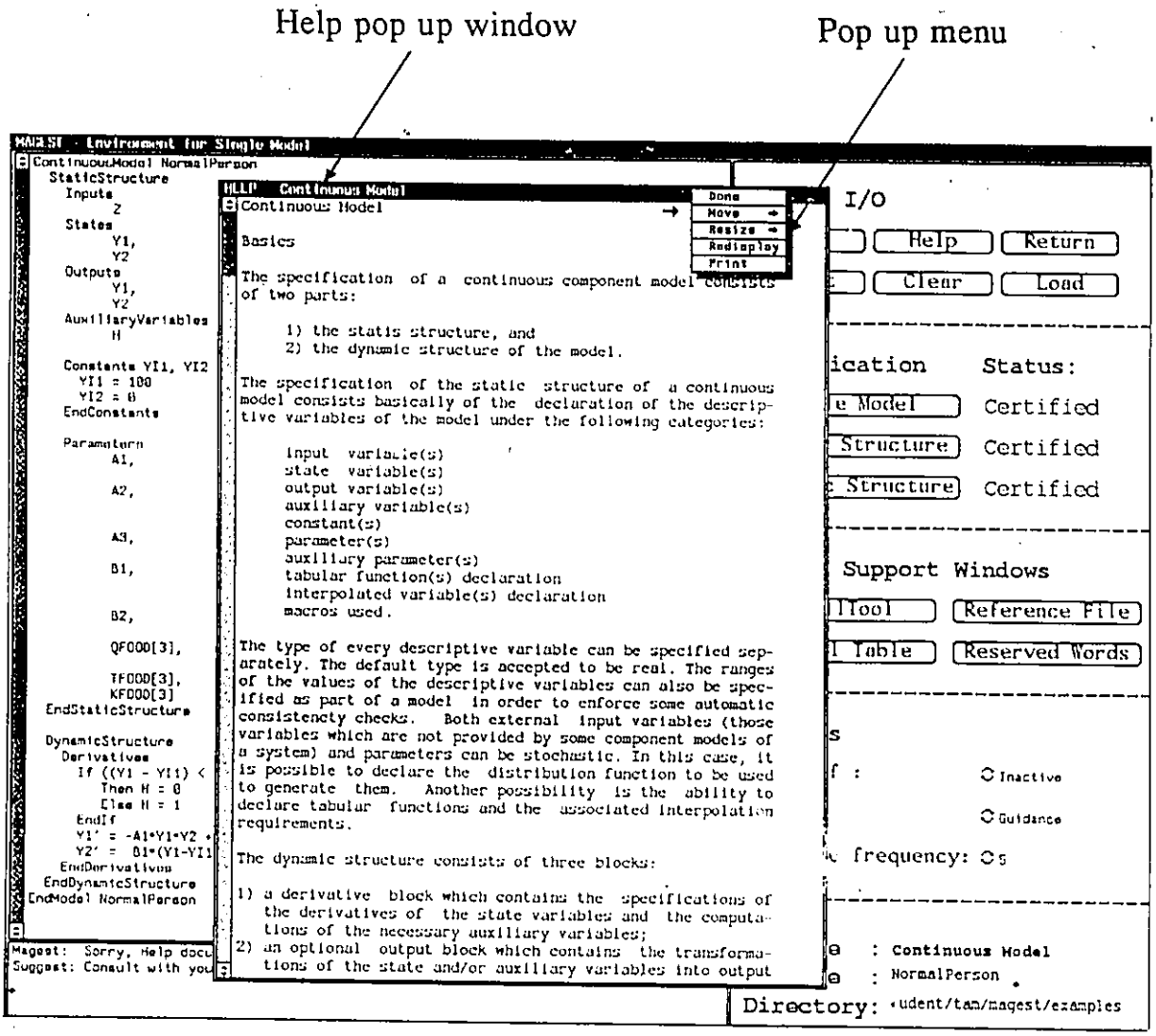

Figure 3.3.3 Help pop up window with pop up menu.

3.3.4 Return

The **Return** function allows the modeller to leave the present environment, either to return to the Main Menu, or to return to the Unix operating system. Once the **Return** button is activated, a submenu appears. As shown in Figure 3.3.4, this submenu contains the return choices.

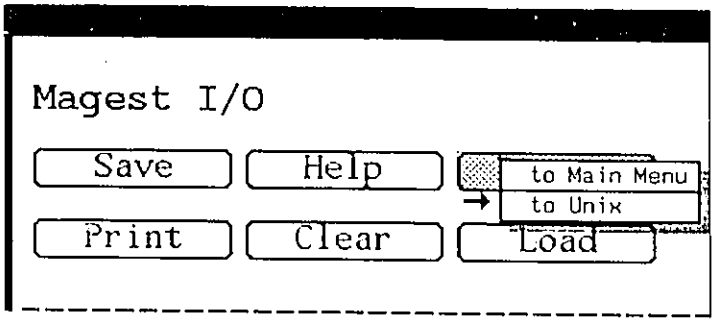

Figure 3.3.4 Submenu of Return function.

3.3.5 Print

The **Print** function provides the ability to print files. Once the **Print** button is activated, a pop up window appears (see Figure 3.3.5). This pop up window contains the following items:

Item	Description
Filename	The name of the file to be printed.
Directory	The directory where the file is located.
Printer Name	- The name of printer where the file has to be printed. - To use laser printer enter: lw - Default is the normal printer.
Page mode	- If the switch is on, page heading information is included. - If the switch is off, page heading information is not included. - Page heading information includes the name of the file, date and page numbering. - Default is on.
Copy	- Number of copies to be printed. - Default is 1.
Column	- Number of columns per output line. - Is ignored for laser printer. - For normal printer (at the University of Ottawa): 80 - indicate 10 pitch (default value). 132 - indicate 12 pitch.

Table 3.3 Description of print file subwindow.

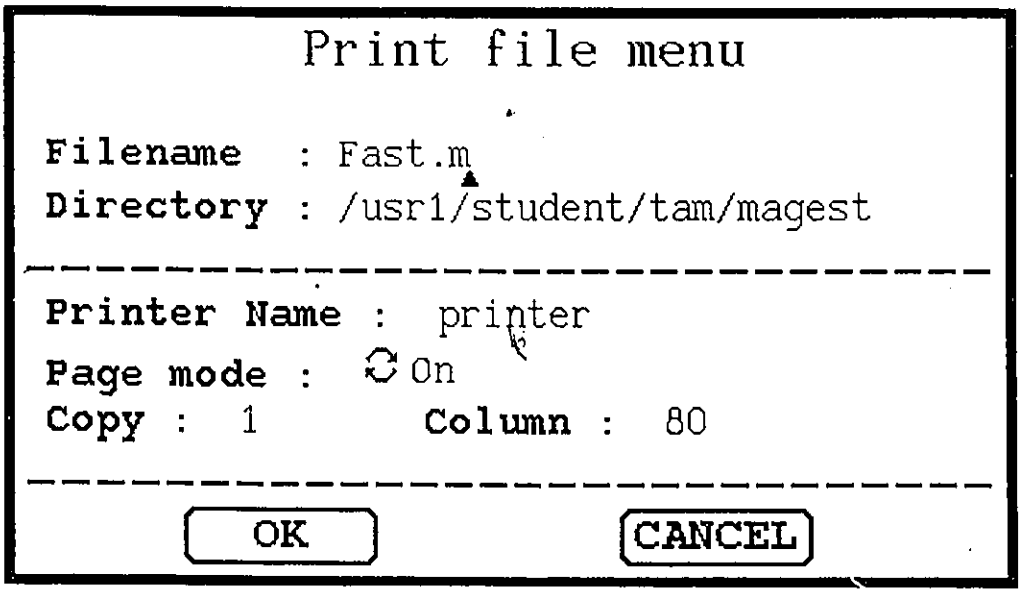

Figure 3.3.5 Print file menu.

3.3.6 Clear

The **Clear** function provides the ability to clear the text editor window. Once the **Clear** button is activated, a pop up window message appears (see Figure 3.3.6). If the modeller presses **Yes,** the system will save the file and clear the text editor window. If the modeller presses **No,** the system will clear the text editor window, without saving the file. If the modeller presses **Cancel,** the system will not clear the text editor window.

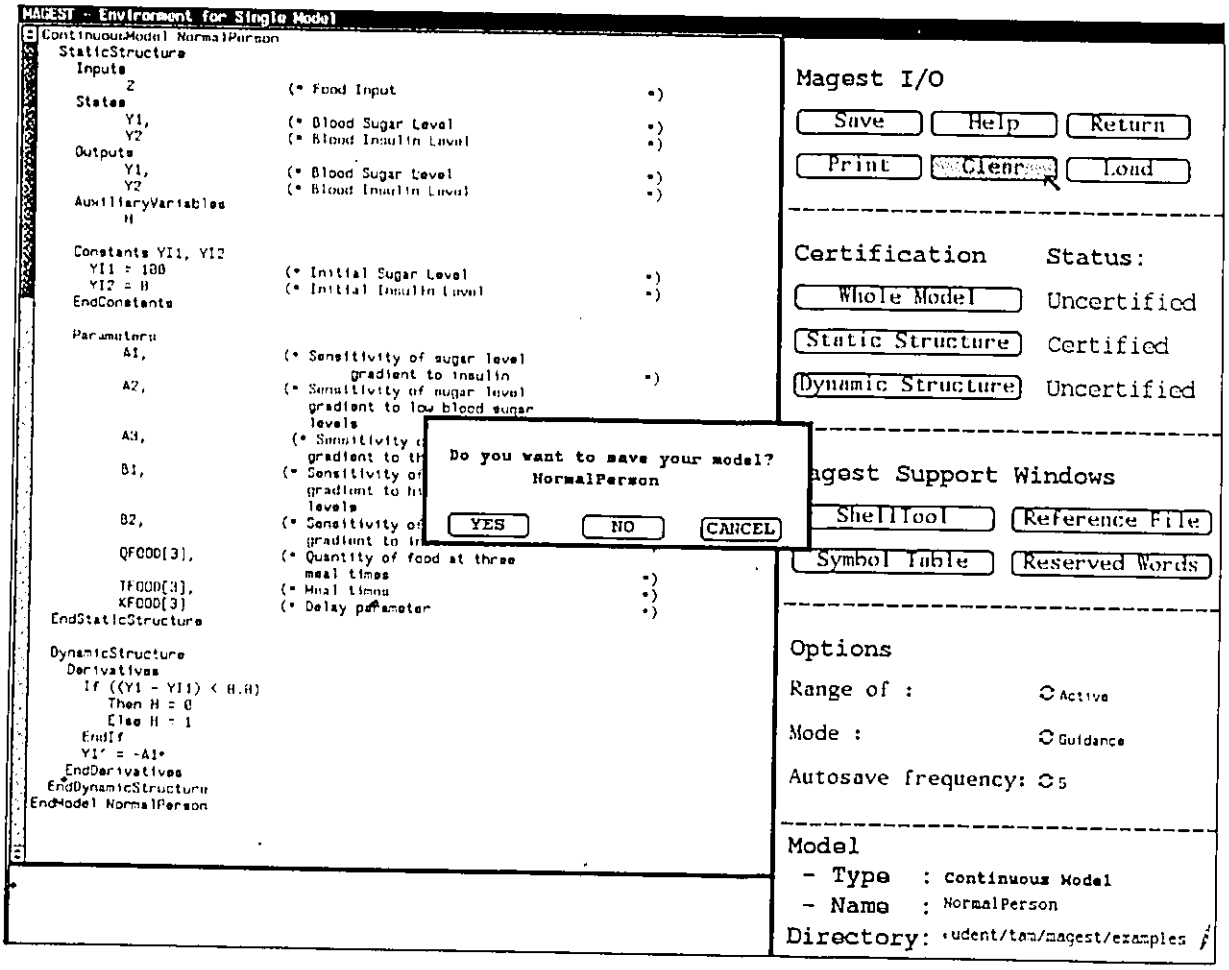

Figure 3.3.6 Confirm save window for **Clear** function.

3.3.7 Load

The **Load** function provides the ability to load a file into the text editor window. Once the **Load** button is activated, a pop up window appears in the center of the screen (Figure 3.3.7). This window asks the modeller to enter the filename and the directory name. By default, the present directory is given.

The text editor window must not contain anything in order to start editing another file. If the editor window has not been cleared, the system will advise the modeller to clear the editor window before loading another file (see Figure 3.3.8). The editor window can be cleared by pressing the **Clear** button (see section 3.3.6).

It is important to know that the type of the model you are going to load must be the same as the type of the model you were working on before. The type of model is indicated in the last panel, which is on the right bottom of the screen.

If the modeller wants to load a model which is a different type, then the procedure is to return to the Magest main menu and activate the proper environment.

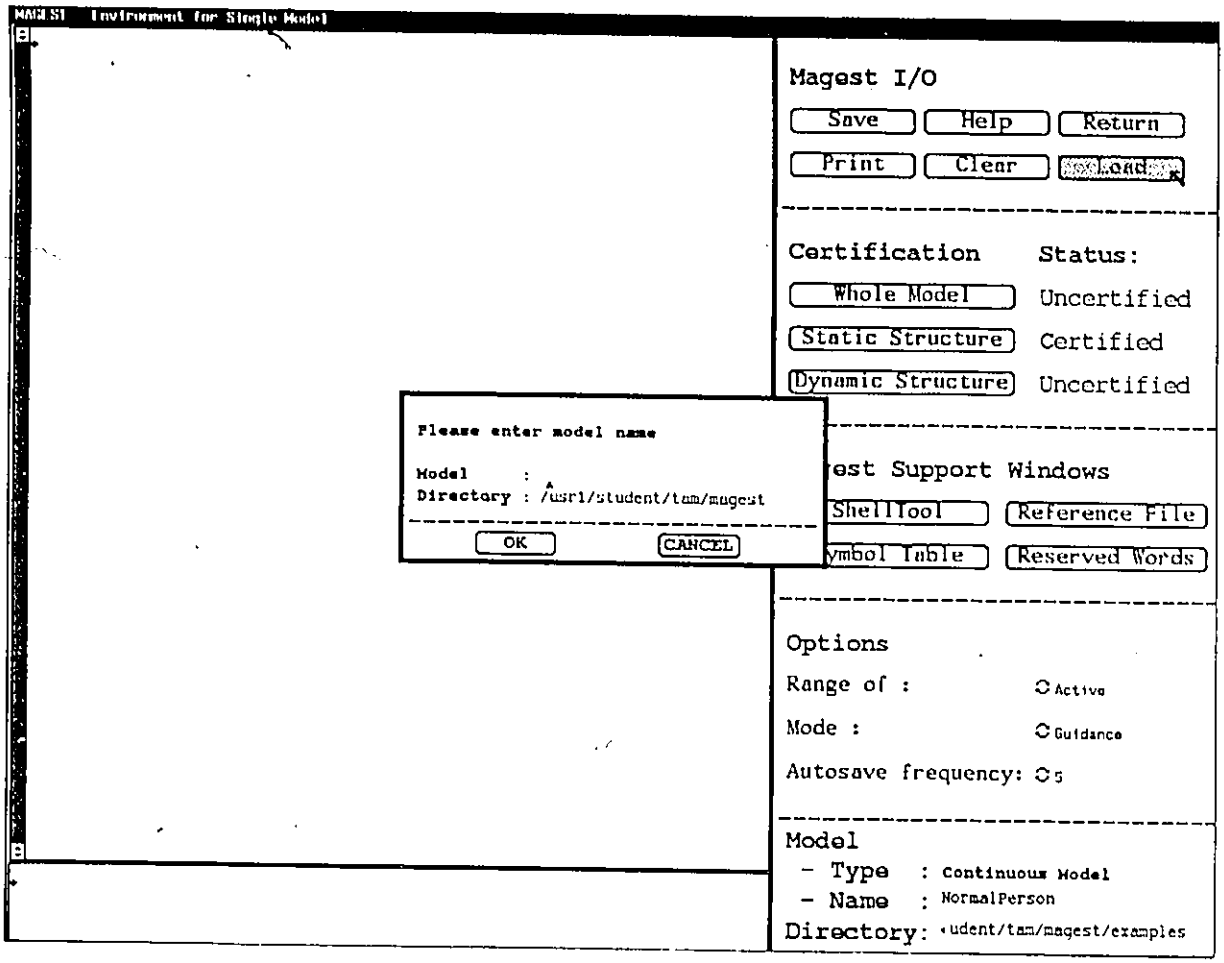

Figure 3.3.7 A pop up window to get information of load file.

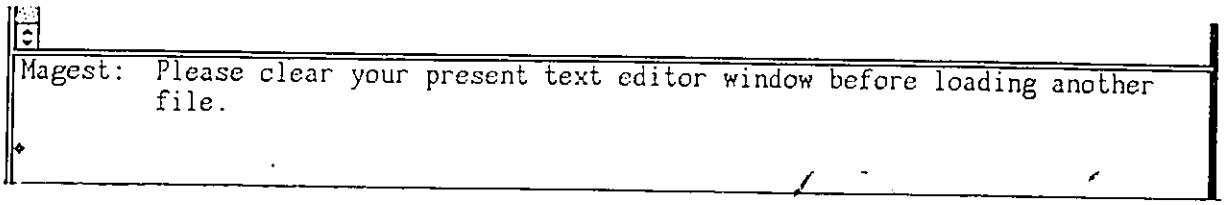

Figure 3.3.8 Message to suggest the text editor window clear before loading another file.

3.4 Prepare Coupling

3.4.1 Introduction to Prepare Coupling

The **Prepare Coupling** function provides facilities for coupled models and multi-rate models, to specify blocks of "MacroModels" as well as templates of "ExternalCouplings" and "InternalCouplings."

The "MacroModels" block is generated by Magest for documentation purposes. It gives a list of all the macro models (if any) used by the components of a coupled or a multi-rate model. The list of the macro models is compiled from the static structures of the component models of a coupled model or a multi-rate model.

The template of the external coupling is based on the knowledge of the external inputs and outputs of the coupled model and the inputs and outputs of the component models. It consists of the specification of an external inputs block where the inputs of the coupled model are assigned to some inputs of relevant component models. The specification of an external outputs block where the outputs of the component models are assigned to some outputs of relevant coupled models.

The template of the internal coupling is based on the inputs of the component models which are not yet assigned any source of input.

3.4.2 External Coupling

The **External Coupling** function provides the specification of a "MacroModels" block and the template of "ExternalCouplings". Once the **External Coupling** button is activated, the knowledge of the coupled model is checked by the system, and the component models of a coupled model are certified by the system. If there are any semantic and/or syntactic errors found, then Magest will inform the modeller to correct the error and the "ExternalCouplings" block will not be provided. If a "MacroModels" block and/or "ExternalCouplings" block already exists, then one of the pop up windows in Figure 3.4.1 is given. After all of that, a "MacroModels" block and/or a template of "ExternalCouplings" is provided.

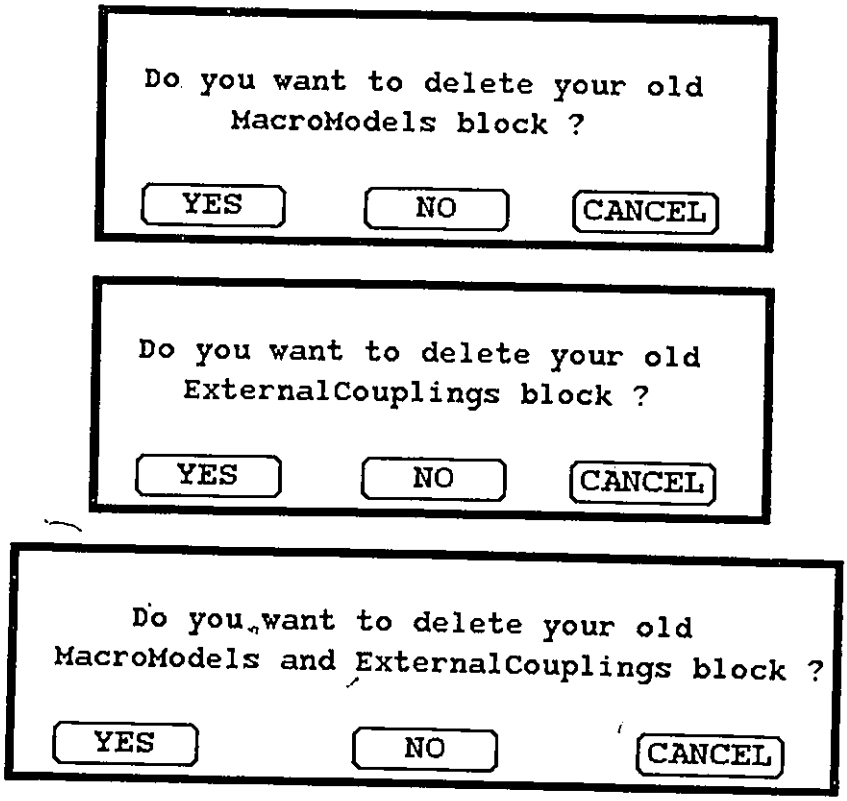

Figure 3.4.1 Confirmation windows to delete old blocks for the **External Coupling** function.

Example:

Consider the coupled model "A" as follows:

```
CoupledModel A
  ExternalVariables
    Inputs   ai1, ai2
    Outputs ao1, ao2
  EndExternalVariables

  ComponentModels
    ContinuousModel B
    DiscreteModel    C
  EndComponentModels
EndModel A
```

Figure 3.4.2 Unfinish coupled Model A.

Continuous model "B" and discrete model "C" are as follow:

```
ContinuousModel B
  StaticStructure
    Inputs bi1, bi2
    States bs
    Outputs bo1, bo2
    Macros m1
  EndStaticStructure
    :
    :
EndModel B
```

```
DiscreteModel C
  StaticStructure
    Inputs ci1, ci2
    States cs
    Outputs co1, co2
    Macros m2, m3
  EndStaticStructure
    :
    :
EndModel C
```

Figure 3.4.3 Continuous model "B" and discrete model "C"

Once the **External Coupling** button is activated, the system checks the specification of the coupled model "A", and certifies component models "B" and "C". After all the necessary checking, the "MacroModels" block and the template of "ExternalCouplings" are given as follows:

```
CoupledModel A
  ExternalVariables
    Inputs  ai1, ai2
    Outputs ao1, ao2
  EndExternalVariables

  ComponentModels
    ContinuousModel B
    DiscreteModel   C
  EndComponentModels

  MacroModels
    m1,
    m2,
    m3
  EndMacroModels

  ExternalCouplings
    ExternalInputs
      A.ai1 -->
      A.ai2 -->
    EndExternalInputs
    ExternalOutputs
      A.ao1 <--
      A.ao2 <--
    EndExternalOutputs
  EndExternalCouplings

EndModel A
```

Figure 3.4.4 A "MacroModels" block and a template of "ExternalCouplings" for the coupled model "A".

The right-hand sides of the symbols "-->" and "<--" are to be filled-in by the user to indicate where the inputs and outputs to model A are connected.

3.4.3 Internal Coupling

The **Internal Coupling** function provides the template of "InternalCouplings". Before working on an internal coupling, the external coupling should be specified and certified by the Magest system. If an "InternalCouplings" block already exists, then a pop up window as shown in Figure 3.4.5 is given. Otherwise, a template of "InternalCouplings" is provided.

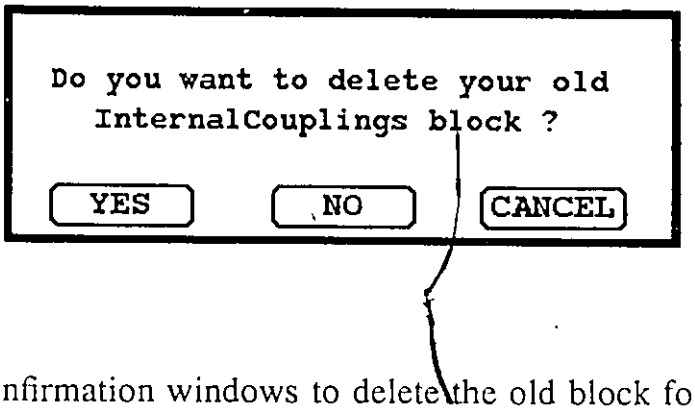

Figure 3.4.5 Confirmation windows to delete the old block for the **Internal Coupling** function.

Example:

Consider the following coupled model "A" where the external coupling is specified and certified:

```
CoupledModel A
  ExternalVariables
    Inputs  ai1, ai2
    Outputs ao1, ao2
  EndExternalVariables

  ComponentModels
    ContinuousModel B
    DiscreteModel   C
  EndComponentModels

  MacroModels
    m1,
    m2,
    m3
  EndMacroModels

  ExternalCouplings
    ExternalInputs
      A.ai1 --> B.bi1
      A.ai2 --> C.ci1
    EndExternalInputs
    ExternalOutputs
      A.ao1 <-- B.bo1
      A.ao2 <-- C.co1
    EndExternalOutputs
  EndExternalCouplings

EndModel A
```

Figure 3.4.6 Unfinish coupled Model A.

Continuous model "B" and discrete model "C" are listed in Figure 3.4.3. Once the **Internal Coupling** button is activated, the system

generates the template of "InternalCouplings" as shown in the last part of Figure 3.4.7.

```
CoupledModel A
  ExternalVariables
    Inputs  ai1, ai2
    Outputs ao1, ao2
  EndExternalVariables

  ComponentModels
    ContinuousModel B
    DiscreteModel   C
  EndComponentModels

  MacroModels
    m1,
    m2,
    m3
  EndMacroModels

  ExternalCouplings
    ExternalInputs
      A.ai1 --> B.bi1
      A.ai2 --> C.ci1
    EndExternalInputs
    ExternalOutputs
      A.ao1 <-- B.bo1
      A.ao2 <-- C.co1
    EndExternalOutputs
  EndExternalCouplings

  InternalCouplings
    B.bi2 <--
    C.ci2 <--
  EndInternalCouplings

EndModel A
```

Figure 3.4.7 A template of "InternalCouplings" for the coupled model "A".

The user has to complete the right-hand sides of the symbol "<--". For example it could be given as in Figure 3.4.8.

```
InternalCouplings
   B.bi2 <-- C.co2
   C.ci2 <-- B.bo2
EndInternalCouplings
```

Figure 3.4.8 InternalCouplings for the coupled model "A".

3.5 Detailed Functions in Magest

Detailed functions in Magest include the following:

1) Certification functions,
2) Magest support windows which include shelltool, reference file, symbol table and reserved words,
3) Options,
4) The text editor walking menu, and
5) Model type insertion functions

3.5.1 Certification

The **Certification** mode is an important function provided by Magest. It provides the ability to perform complete syntactic and semantic checks on a model. Rules for syntactic checks are according to the BNF (Backus-Naur Form) representation of the Gest language. Most of the semantic checks are according to the semantic facts and rules as specified by Ören and Sheng (1988). Over ten more semantic facts and rules have been added to the system. Details of semantic facts and rules are discussed in chapter 5.

As seen in Figure 3.5.1, 3.5.3 and 3.5.4, these are the certification section of the environments for single models, coupled models and multi-rate models respectively.

The certification section for the environments for single models is slightly larger than the other sections of the environment. For a single model, there are three different kinds of certification, i.e., certification of whole model, static structure, and dynamic structure. Certification of static structure requires checks on static structure only. Certification in dynamic structure performs checks on dynamic structure with respects to knowledge already accepted in the static structure. However, certification for the whole model performs complete checks to both structures, as well as some additional semantic checks of the static structure if the dynamic structure has already been certified.

In the environment for single models, certification of dynamic structure will not be performed if the static structure is uncertified. This is

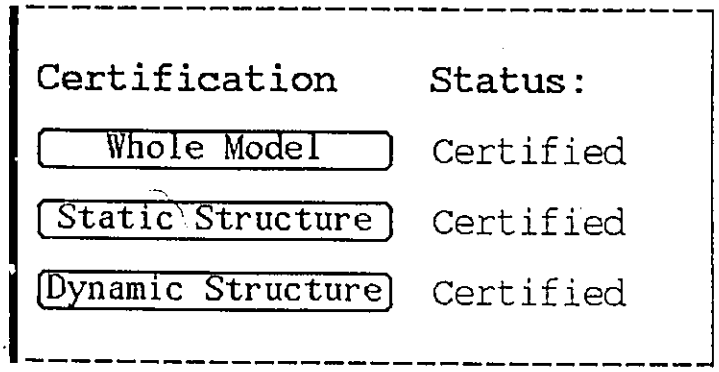

Figure 3.5.1 Certification functions of the environment for single models.

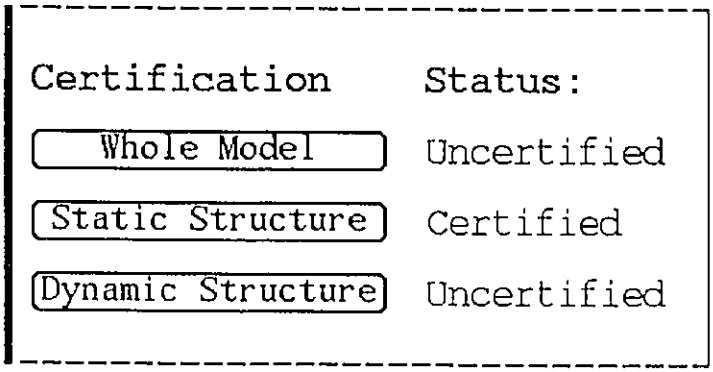

Figure 3.5.2 Only Static Structure is certified.

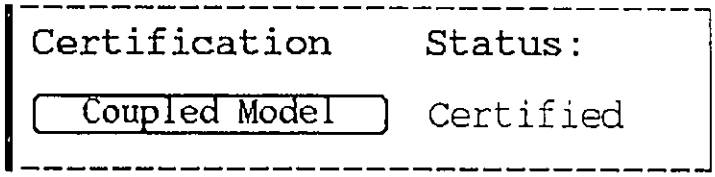

Figure 3.5.3 Certification function of the environment for coupled models.

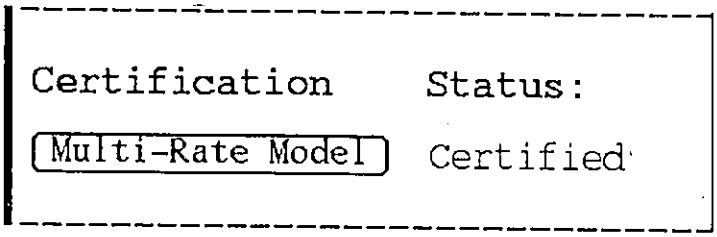

Figure 3.5.4 Certification function of the environment for multi-rate models.

because certification of dynamic structure requires knowledge from the static structure. Moreover, a very simple logical arrangement is applied to the status indication in single model mode only. If one of the structures is not certified, the whole model status is indicated "Uncertified". In addition to that, the whole model status is indicated "Uncertified" automatically. As shown in Figure 3.5.2, the whole model is uncertified because the dynamic structure is uncertified; however the static structure is certified.

The certification sections for the coupled model and multi-rate model environments include the certification of the coupled model and multi-rate model only. However, when assistance is required from Magest to prepare couplings, then all component models of a coupled model (or a multi-rate model) are certified automatically. Once a component model has been certified, Magest will store the knowledge in the model-base. Therefore, Magest will not certify a component model again on the next certification, unless, the modeller has performed some editing to the component model specification. This reduces redundant certification of component models and increases the efficiency of the Magest system.

3.5.2 ShellTool

The **ShellTool** function provides a Unix emulator subwindow. "It runs the shell, or the command interpreter, that you have typed Unix commands to all along" (Sun Microsystems 1986a). The ShellTool function is provided in the Gest executive and all of the Magest environments. Figure 3.5.5 shows the Shelltool subwindow in the environment of a single model.

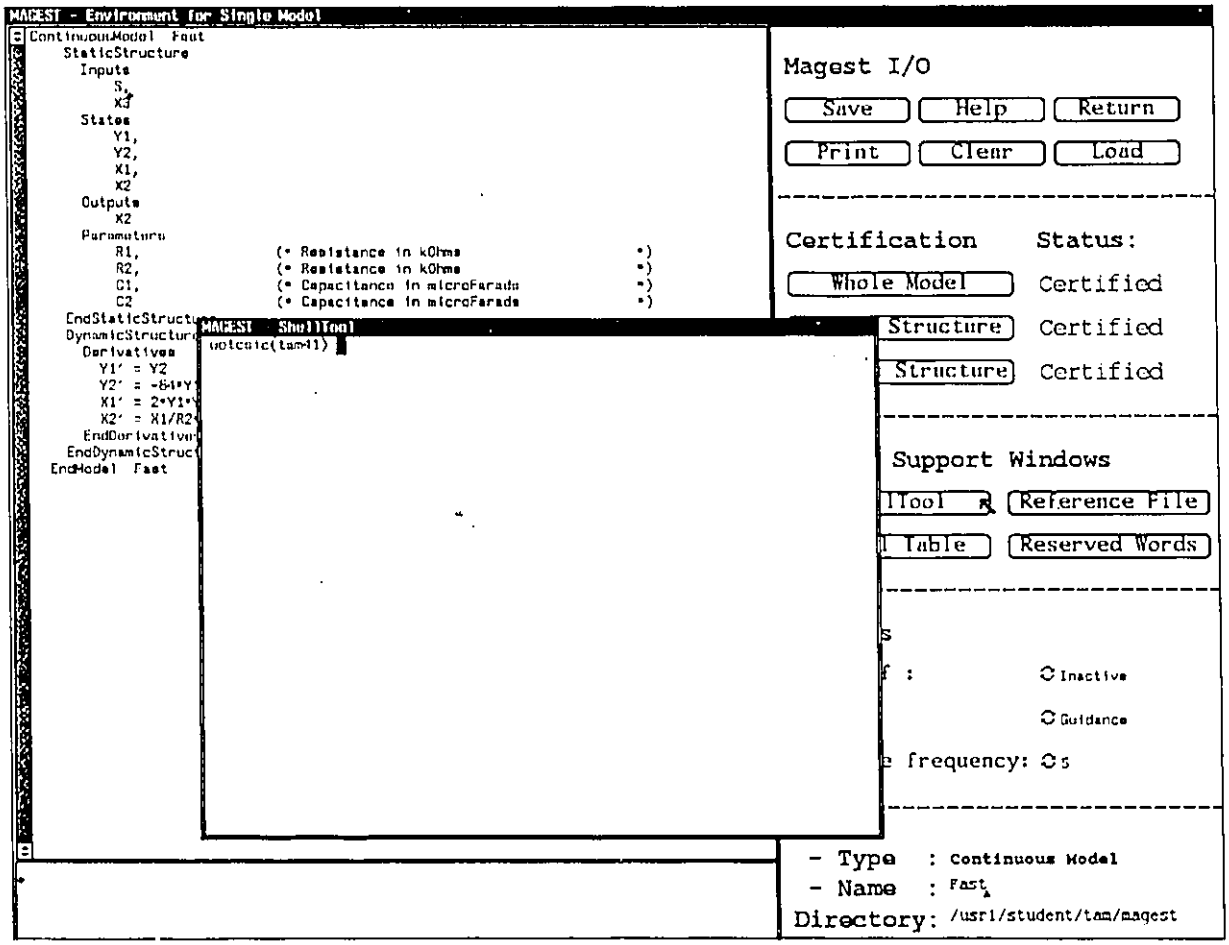

Figure 3.5.5 The ShellTool subwindow.

3.5.3 Reference File

The **Reference File** function provides the ability to view another file. Once the **Reference File** button is activated, a pop up window appears (see Figure 3.5.6). The modeller should enter the following information:

1) The name of the model.

2) The directory corresponding to the file for the model. By default, the modeller's current directory is given.

After entering information above, activate the **OK** button.

At this moment, Magest checks whether or not the given filename exists in the directory. If the file already exists, then Magest opens a text editor subwindow, and displays the file in it (see Figure 3.5.7). This text editor subwindow is read-only. It prevents the modeller from entering incorrect entries by mistake.

Note that due to the sun workstation's limitation on windows, this subwindow is shared with the help subwindow as explained in section 3.3.3. Therefore, when this subwindow appears, the help subwindow disappears. Similarly when the help subwindow appears, the reference file subwindow disappears.

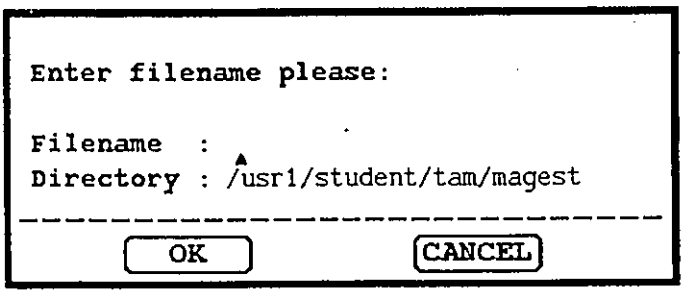

Figure 3.5.6 Once the **Reference File** button is activated, a pop up window appears.

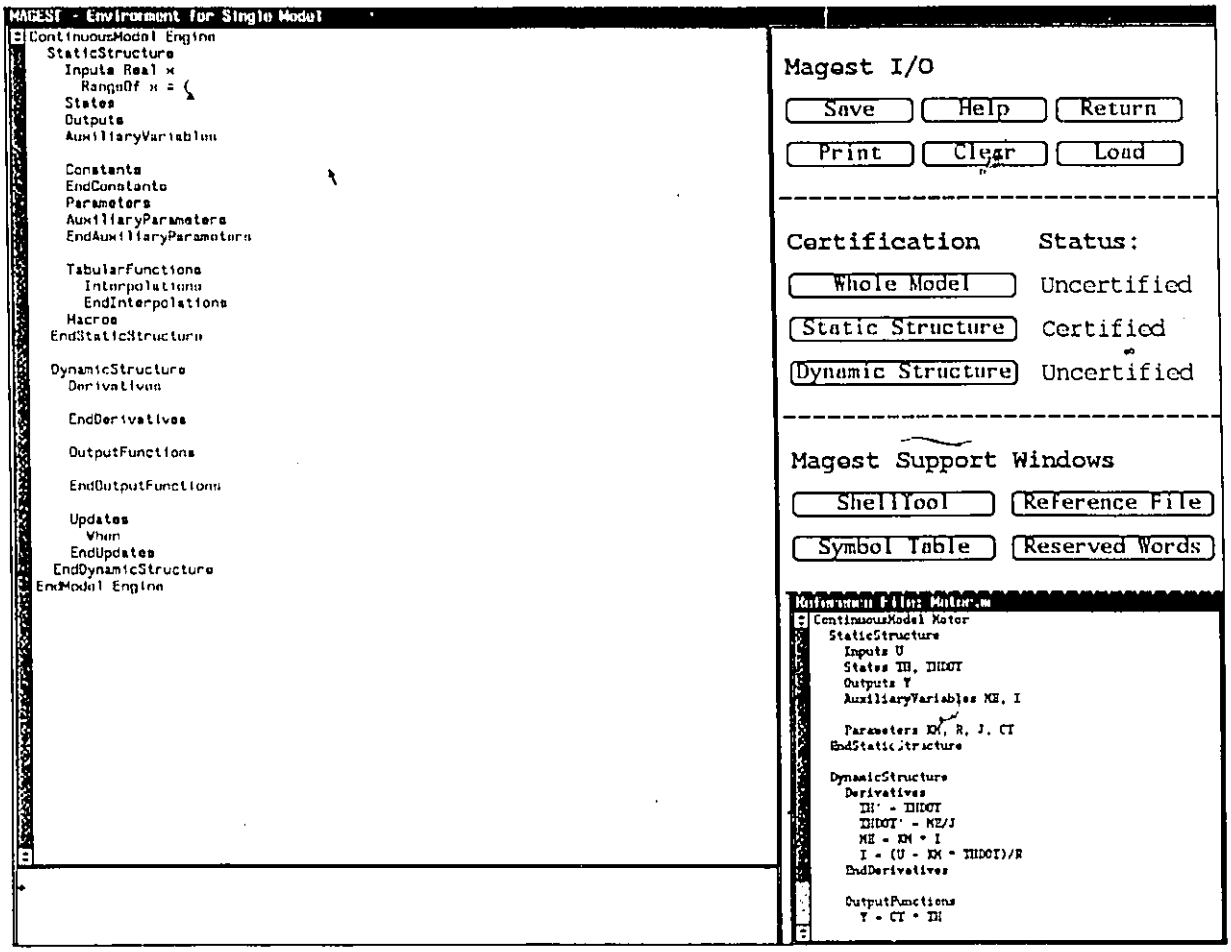

Figure 3.5.7 Reference file subwindow.

3.5.4 Symbol Table

The **Symbol Table** provides the modeller with the symbols used in the present model. Once the **Symbol Table** button is activated, a pop up window appears (see Figure 3.5.8). The symbol table has three sections: name, structure and type. The structure is where the symbol is defined. The type is the type of arithmetic variable classification. After any modification on a model, once the **Symbol Table** button is activated, the updated symbol table is displayed.

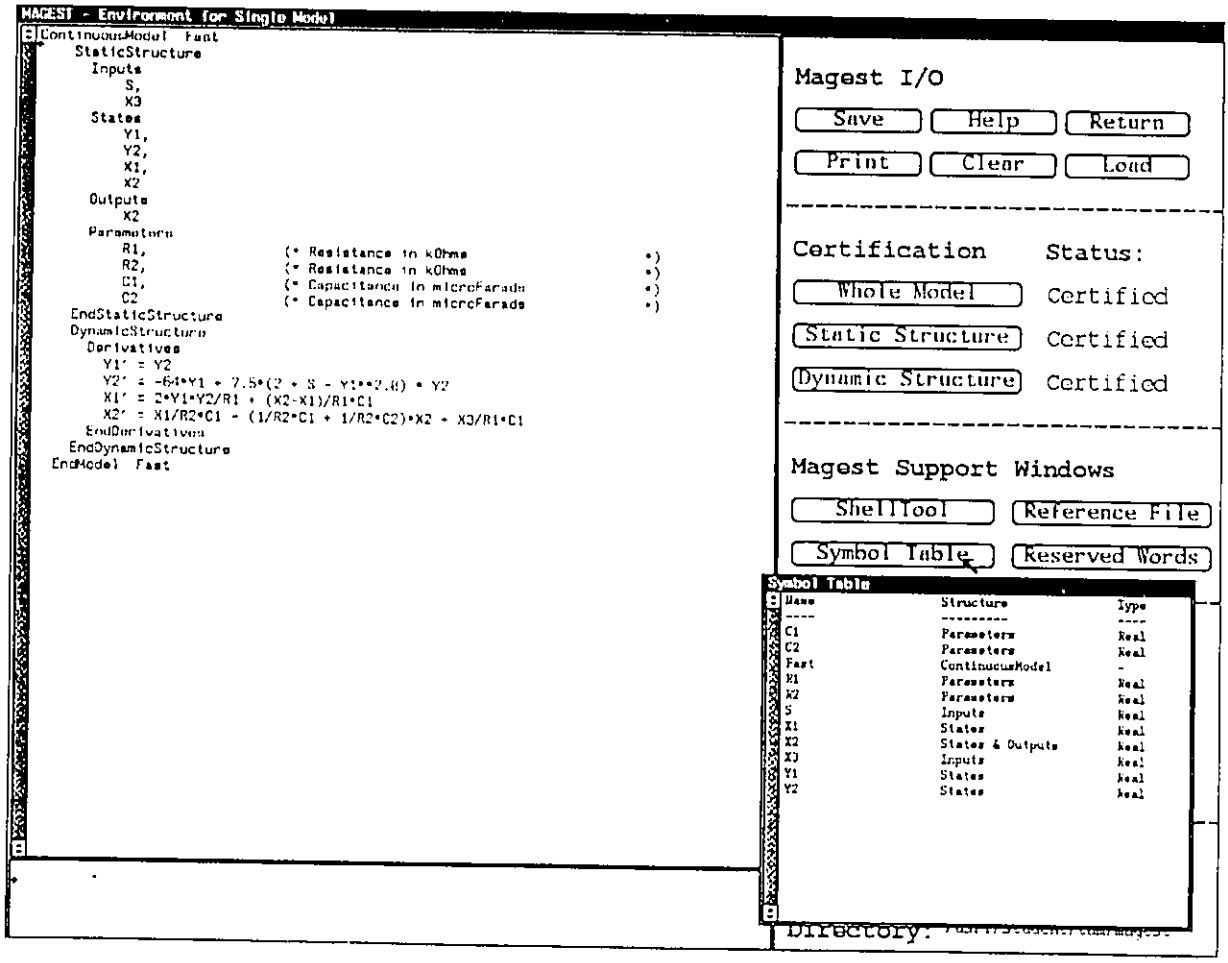

Figure 3.5.8 Symbol table subwindow.

3.5.5 Reserved Words

The **Reserved Words** function provides the ability to display the appropriate section of the Gest reserved words on the reserved word window. Once the **Reserved Words** button is activated, a submenu appears. As shown in Figure 3.5.9, this submenu contains the alphabetic letter from A to Z. It represents the leading letter of reserved words. After the modeller has made a selection, a subwindow will be shown with the appropriate reverved words (see Figure 3.5.10).

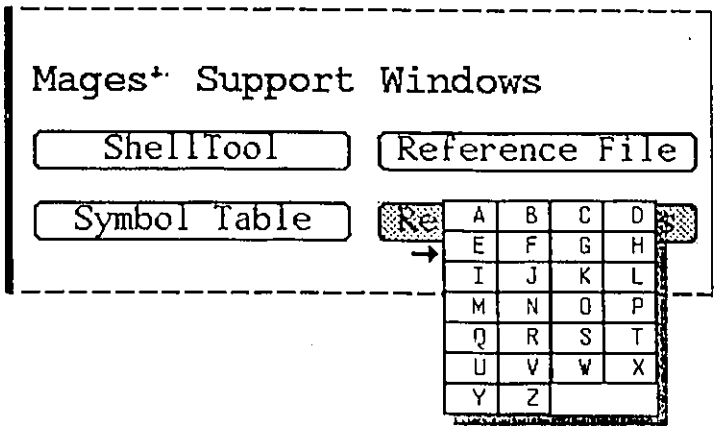

Figure 3.5.9 Submenu of reserved words function.

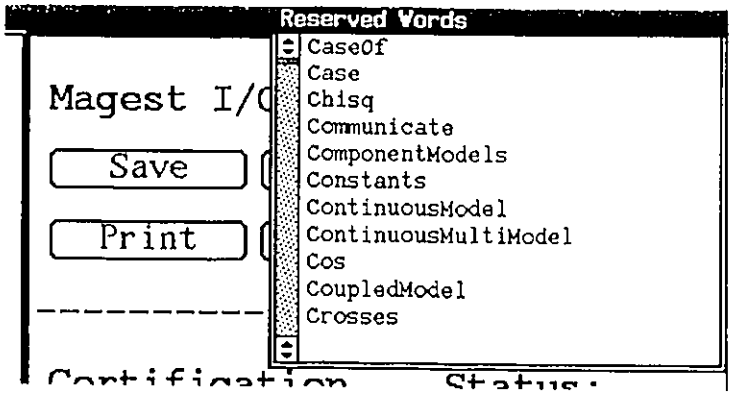

Figure 3.5.10 Reserved words window.

3.5.6 Options

The **Options** section provides the ability to select options provided by Magest. Figure 3.5.11 shows options which could be selected. Table 3.4 below summarizes the options:

Items	Options	Explanations
Range Of	Active	Range Of declaration will be given automatically in Static Structure for all the variables and parameters.
	Inactive	Range Of declaration will not be given automatically in Static Structure.
Mode	Guidance	Magest performs syntactic and semantic checks for each input entry in the text editor window. An input entry is defined by pressing the return key in the text editor window. Checking is performed until the current line in the text editor window.
	Edit	No checks are done on the text shown in the text editor window.
Autosave frequency	5, 10, 15, 20, 0	Number of modifications in the text editor window to automatically save the entries. "0" means no auto-saving has to be done.

Table 3.4 Options in Magest.

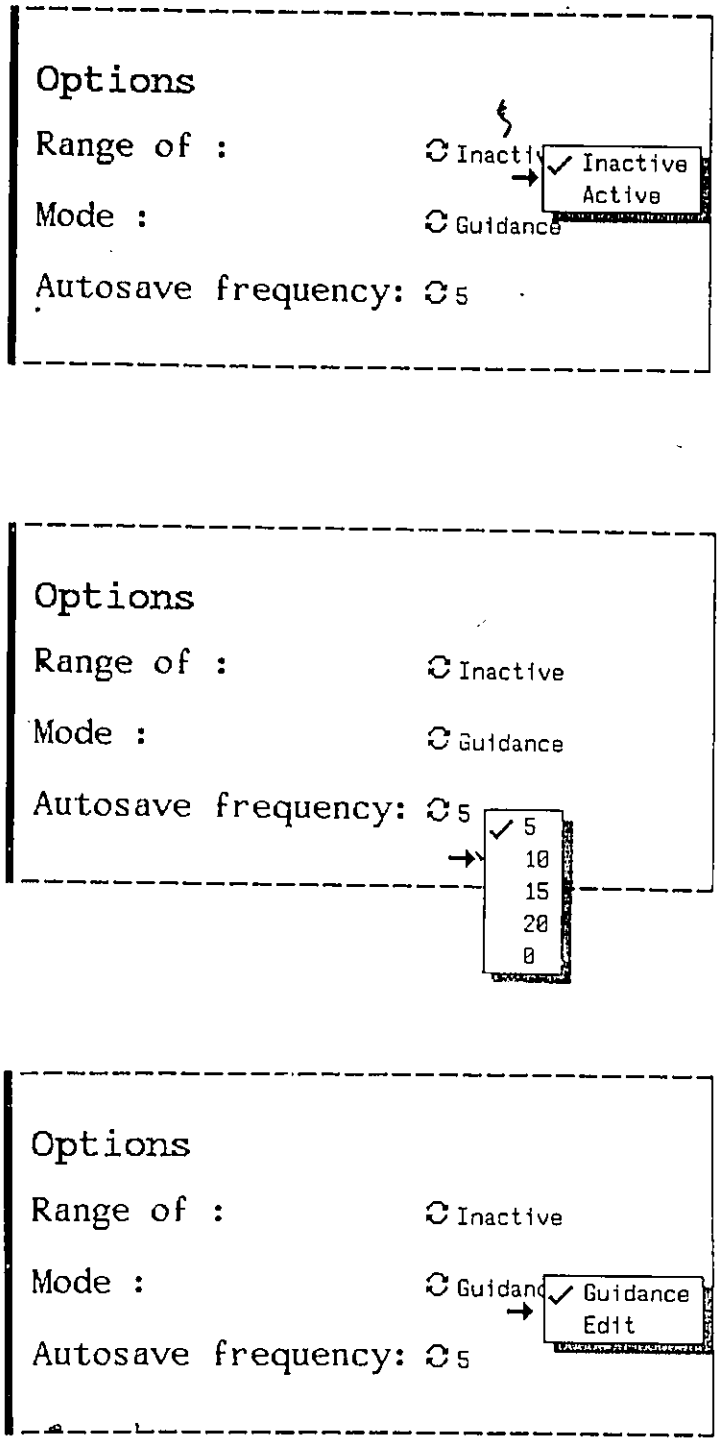

Figure 3.5.11 Options in Magest.

3.5.7 Walking menu in the Text Editor

The walking menu can be obtained anywhere in the text editor window. It provides many convenient functions for the modeller. Since the model structure depends on the type of Gest model, the functions provided in the walking menu are different. Figures 3.5.12 and 3.5.13 are the walking menus for environments for the single models, coupled models and multi-rate models.

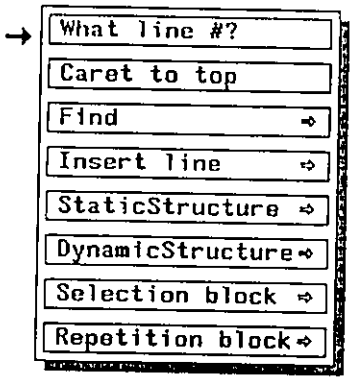

Figure 3.5.12 Walking menu for the single model text editor window.

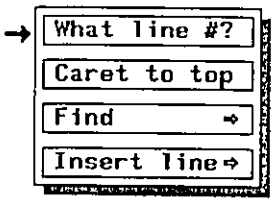

Figure 3.5.13 Walking menu for the coupled model and multi-rate model text editor window.

Table 3.5 below summarizes the functions in the walking menu:

Items	Explanations
What line #?	Report line number of the current caret location.
Caret to top	Send the caret location to the top of the text editor window.
Find	Perform string search functions. Figure 3.5.14 shows the submenu.
Insert line	Insert blank lines below the current line. Figure 3.5.15 shows the submenu.
Static Structure	Figures 3.5.16-3.5.17 shows all submenus of the static structure for different types of models. This function insert an item into the static structure of the model.
Dynamic Structure	Figures 3.5.18-3.5.24 shows all submenus of the dynamic structure for different types of models. This function insert an item into the dynamic structure of the model.
Selection block Repetition block	Figures 3.5.25 and 3.5.26 show the submenus of these functions. These functions insert the specific program structure in the model, starting at the current cursor location.

Table 3.5 Functions in walking menu.

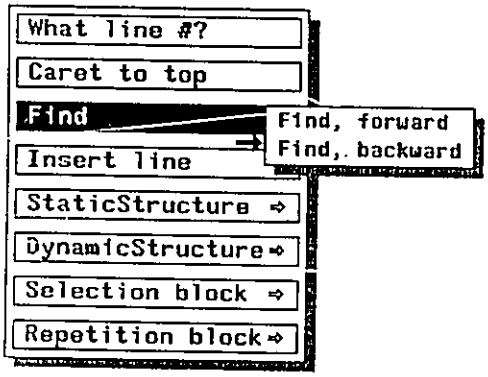

Figure 3.5.14 Submenu of Find.

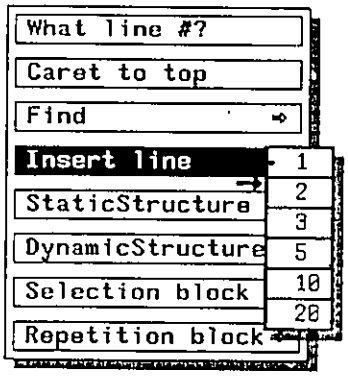

Figure 3.5.15 Submenu of Insert line.

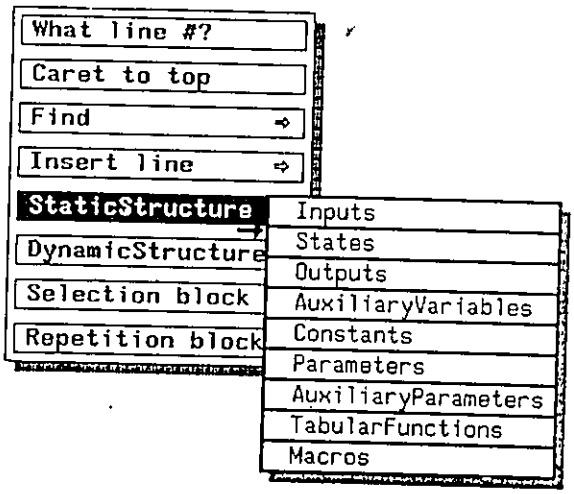

Figure 3.5.16 Submenu of Static Structure for continuous models, continuous multi-models, discrete models and discrete multi-models.

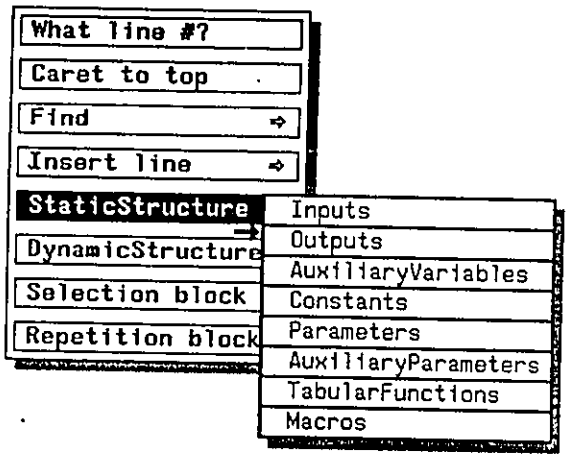

Figure 3.5.17 Submenu of Static Structure for memoryless models, memoryless multi-models and macro models.

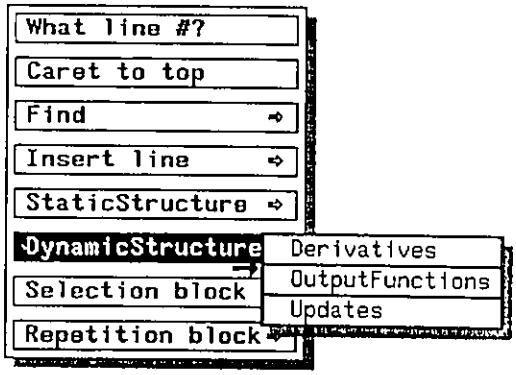

Figure 3.5.18 Submenu of Dynamic Structure for continuous models.

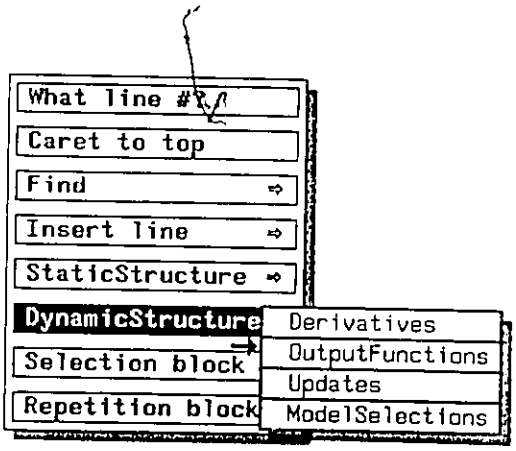

Figure 3.5.19 Submenu of Dynamic Structure for continuous multi-models.

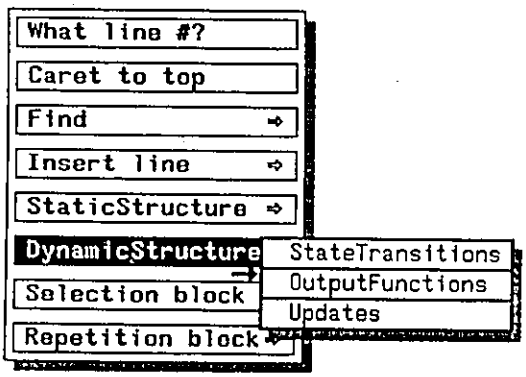

Figure 3.5.20 Submenu of Dynamic Structure for discrete models.

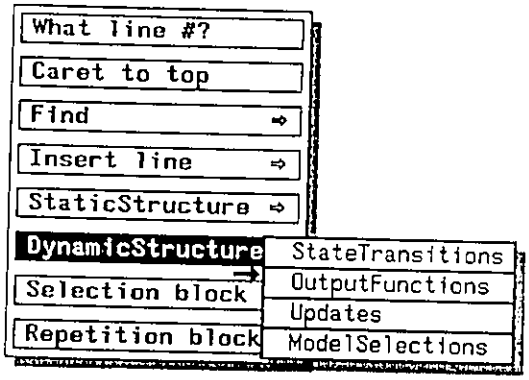

Figure 3.5.21 Submenu of Dynamic Structure for discrete multi-models.

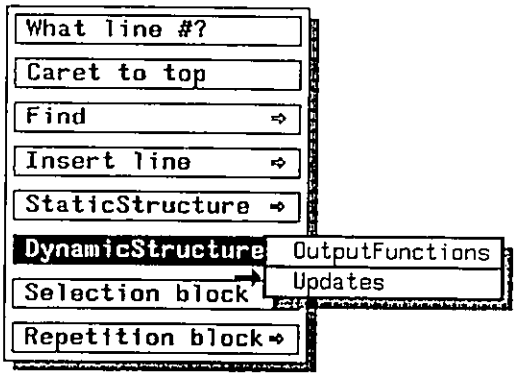

Figure 3.5.22 Submenu of Dynamic Structure for memoryless models.

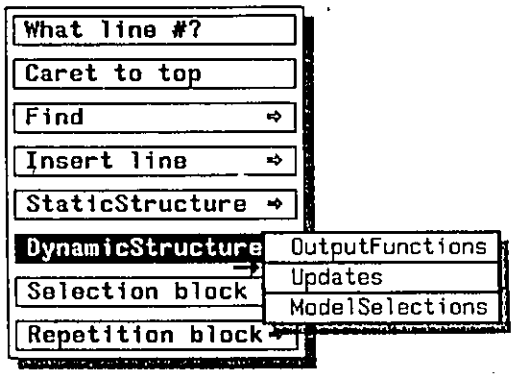

Figure 3.5.23 Submenu of Dynamic Structure for memoryless multi-models.

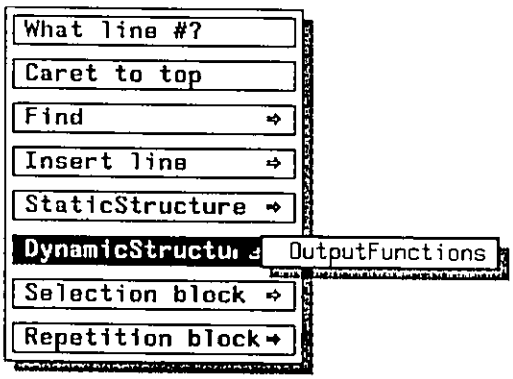

Figure 3.5.24 Submenu of Dynamic Structure for macro models.

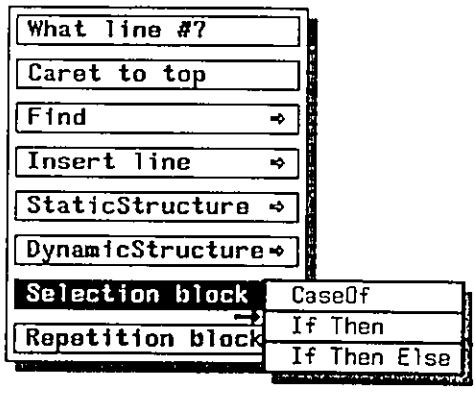

Figure 3.5.25 Submenu of Selection block.

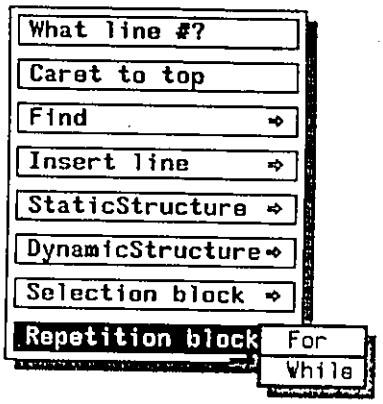

Figure 3.5.26 Submenu of Repetition block.

3.5.8 Insert Type of Models

Insert Type of Models function provides abilities for coupled models and multi-rate models to insert keywords in the text editor window. Once a button is activated, the corresponding keyword will be inserted at the cursor location of the text editor window.

The objective of this function is to provide keywords while the modeller is doing the specification for the "ComponentModels" block. Instead of obliging the user to type the keyword for the type of a model, it is provided by the system.

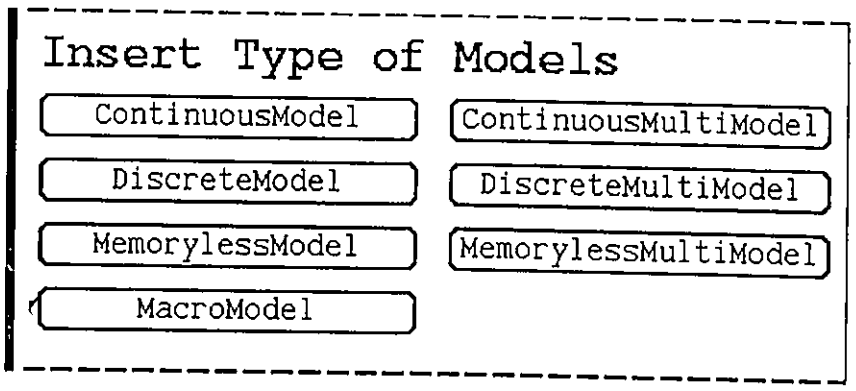

Figure 3.5.27 Insert Type of Models functions.

Chapter 4

MODELLING ASSISTANCE PROVIDED BY MAGEST

System functions in Magest have been described in chapter three. In this chapter, we demonstrate the specification of single models, coupled models and multi-rate models in the Magest environment.

4.1 Single Model Specification

In this section we demonstrate the specification of a single model in Magest. For example, we have the following continuous model:

```
ContinuousModel Motor
  StaticStructure
    Inputs U, V
    States TH, THDOT
    Outputs Y
    AuxiliaryVariables ME, I
    Parameters KM, R, J, CT, TH
  EndStaticStructure

  DynamicStructure
    Derivatives
      TH' = THDOT
      THDOT' = ME/J
      ME = KM * I
      I = (U - KM * THDOT)/R
    EndDerivatives

    OutputFunctions
      Y = CT * TH
    EndOutputFunctions
  EndDynamicStructure
EndModel Motor
```

Figure 4.1.1 Rough version of continuous model "Motor".

This is a rough and uncertified copy of model "Motor". Some example errors to be detected by Magest are shown in **bold face** characters.

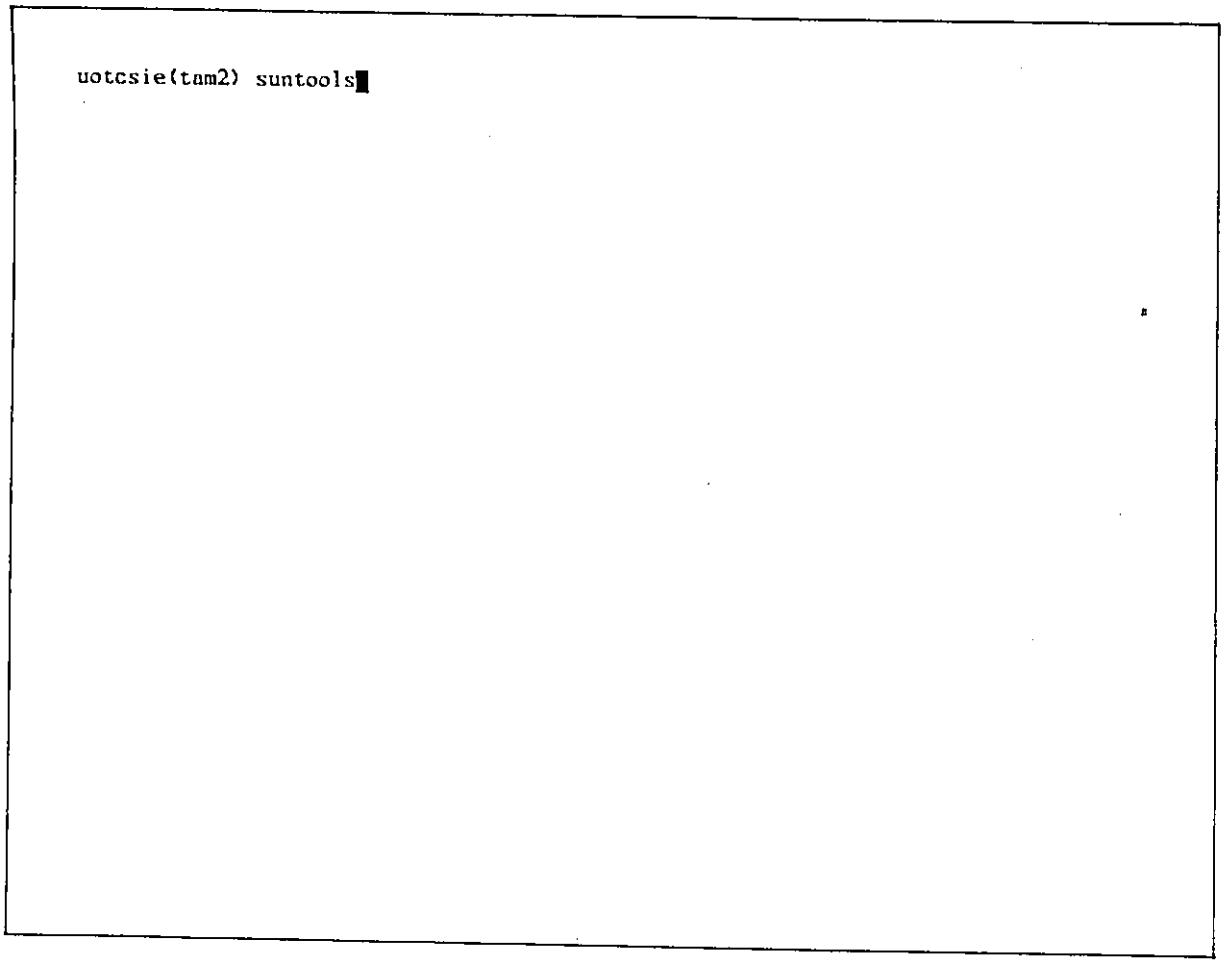

Figure 4.1.2 From Unix to Suntool.

Here, we are starting from the primary environment of Sun workstation. Type **suntool** to enter the SunView window environment. After a few seconds, you see the following screen:

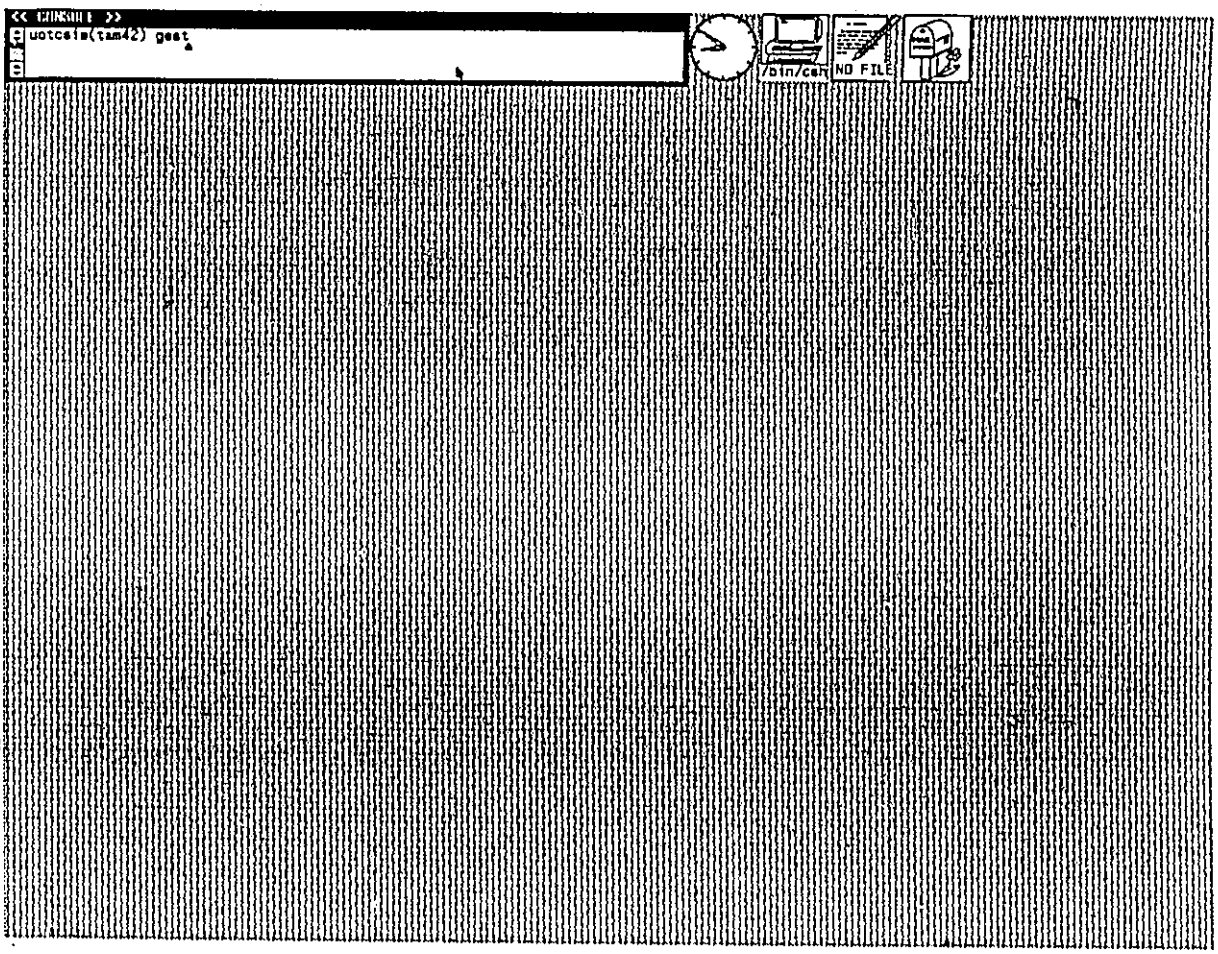

Figure 4.1.3 Type **gest** to enter the Gest Environment.

This is a standard SunView window environment. In one of the Unix emulator windows; e.g., Shelltool or Cmdtool (Sun Microsystems, 1986a). Type **gest** to enter the Gest environment. After a few seconds, the following screen appears:

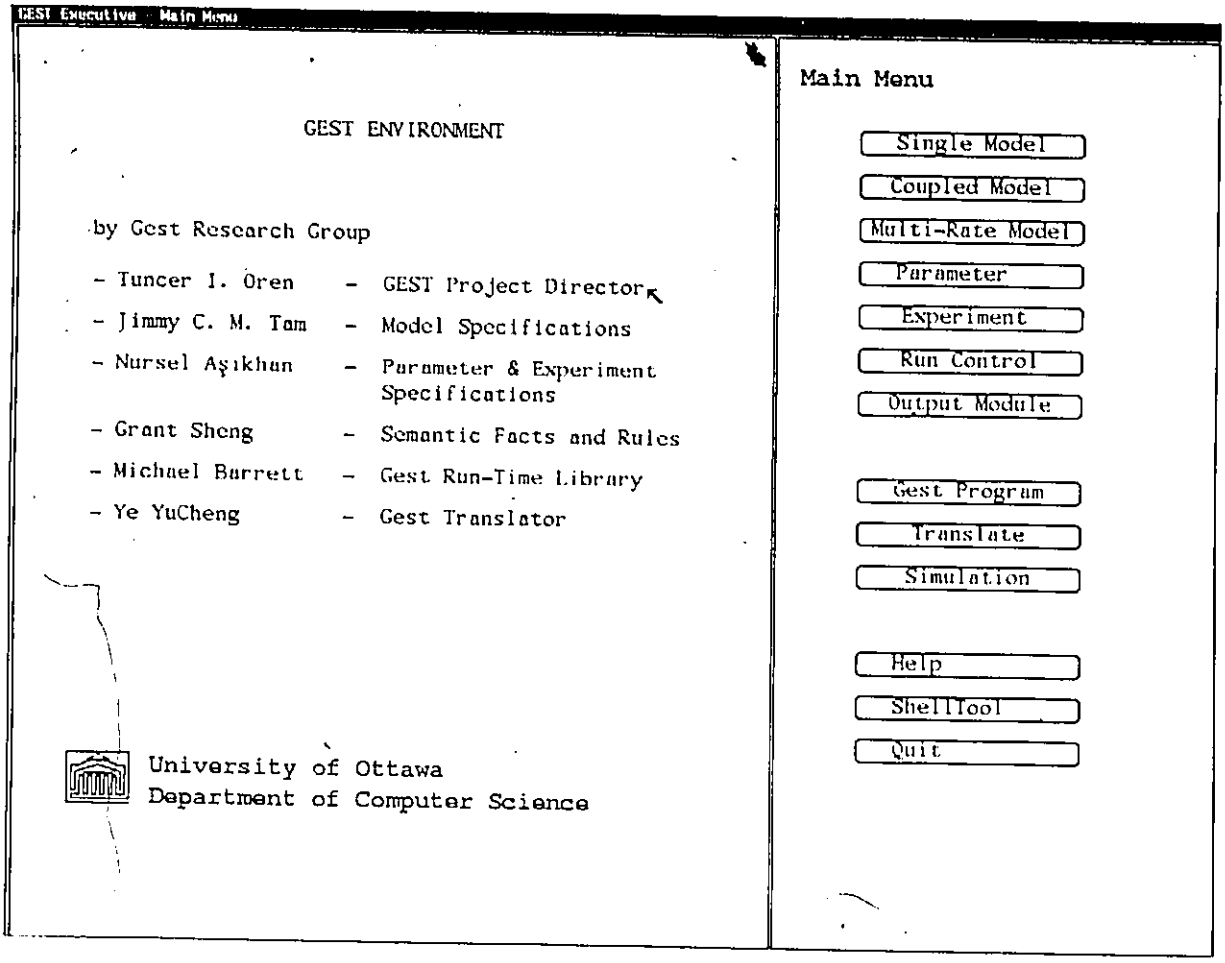

Figure 4.1.4 Gest Environment.

This is a typical main menu of the Gest environment. As depicted in section 3.1.2, this is also called the Gest Executive. Since the continuous model "Motor" is a single model, press the **Single Model** button.

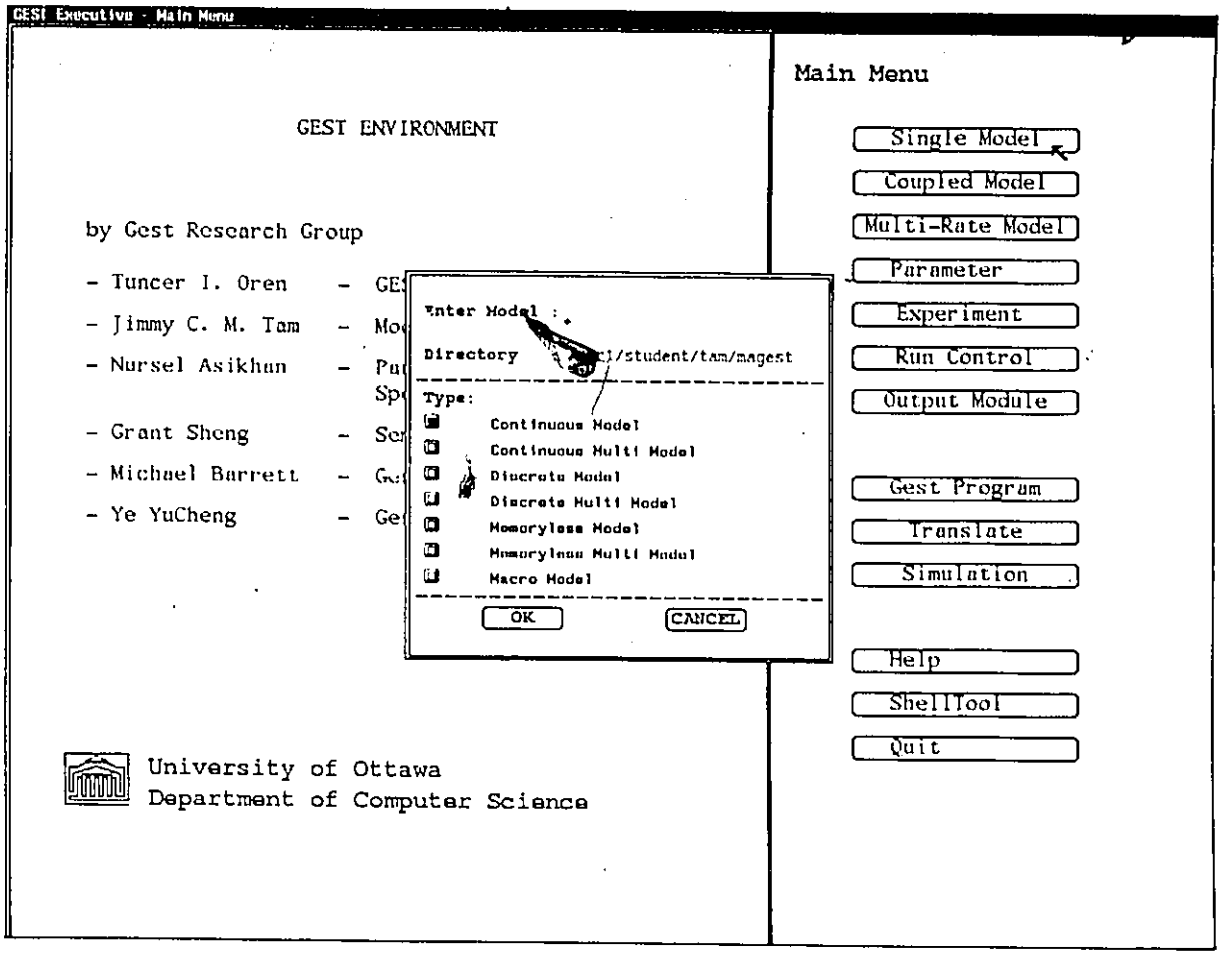

Figure 4.1.5 A pop up window appears on the center of the screen, for single model environment entry.

A pop up window shows up. Enter the model name "Motor" in the model name section and the directory corresponding to the file where the model will be kept. You can change the directory also.

By default, the type of model is a continuous model. Therefore, we don't need to change the selection. After entering and selecting all of the information above, press the **OK** button.

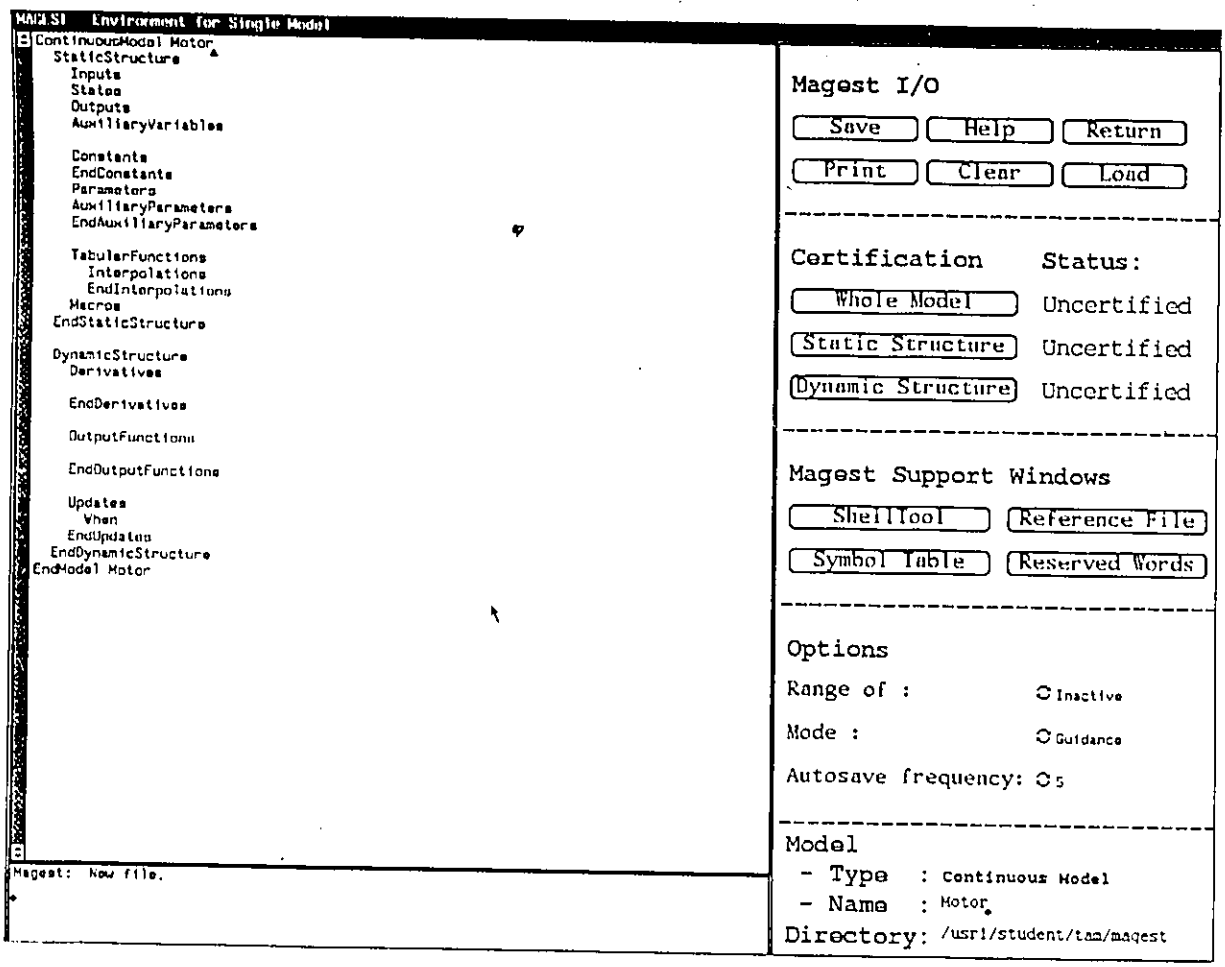

Figure 4.1.6 Environment for single model.

As shown in Figure 4.1.6, a template of continuous model "Motor" is given. The purpose of this template is to provide all of the necessary structure of a continuous model. The structures which are not needed can be deleted from the template..

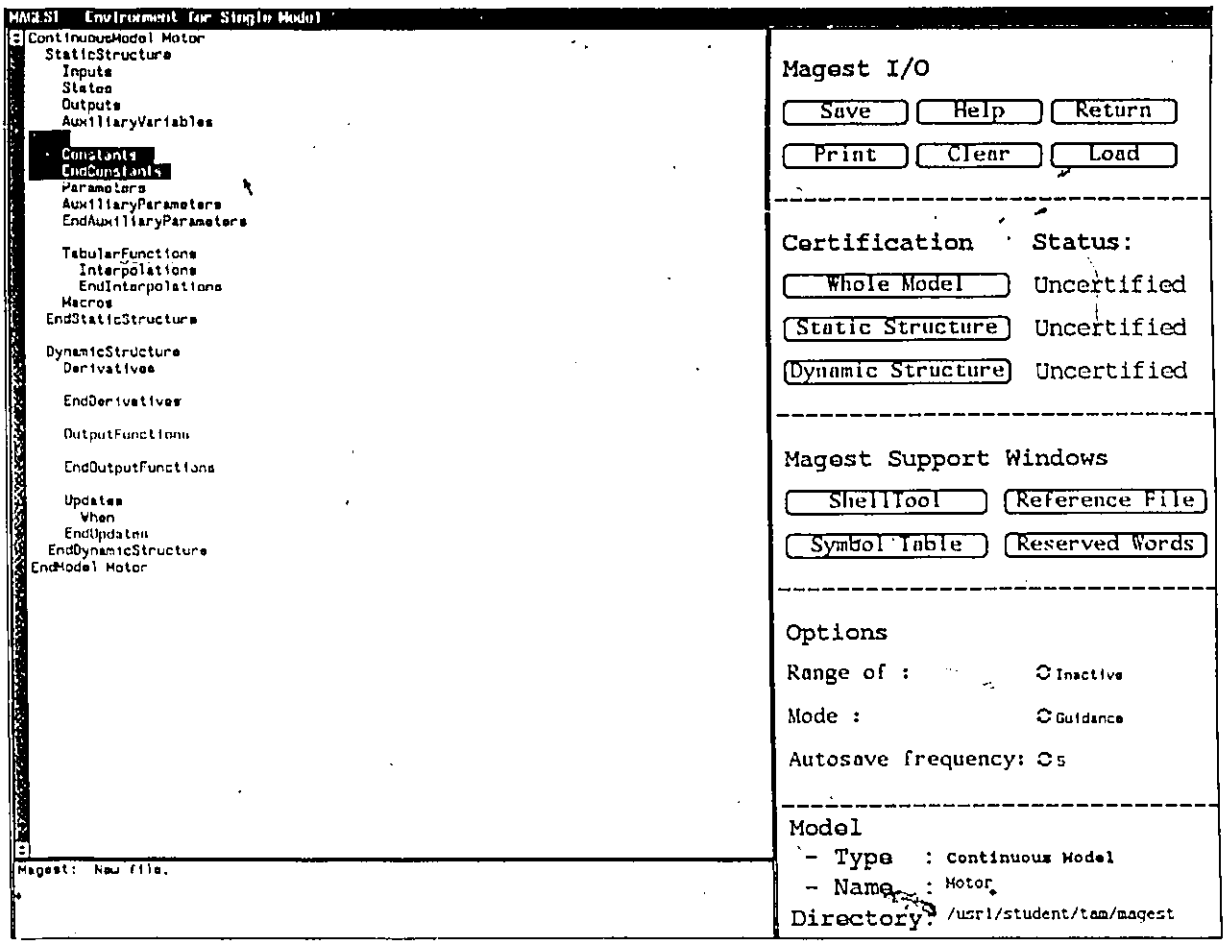

Figure 4.1.7 Deleting a block in the text editor window.

It is so easy to delete the structure we don't need. To do the text delete operation, follow the procedure below:

1) Press the left mouse button for the first character of the block.

2) Press the right mouse button for the last character of the block.

3) Then, press the function key **L10.**

To undo a deletion one can press the key **L4**. One can also select the functions from the text editor walking menu to put back some already deleted items. By pressing the right button in the text editor window, a

walking menu will appear.

After the deletions, move the mouse to the text editor window, press the left mouse button beside the "Inputs" specification to set the cursor position. Now, you can enter the Gest model specification line by line. When you have finished a line, press the return key to the next line.

Since you are under the guidance mode of Magest, everything you enter is analyzed by the system. If the entries are correct, the cursor will position to the end of the following line. However, if any entry contains errors, Magest will give an error message in the message window.

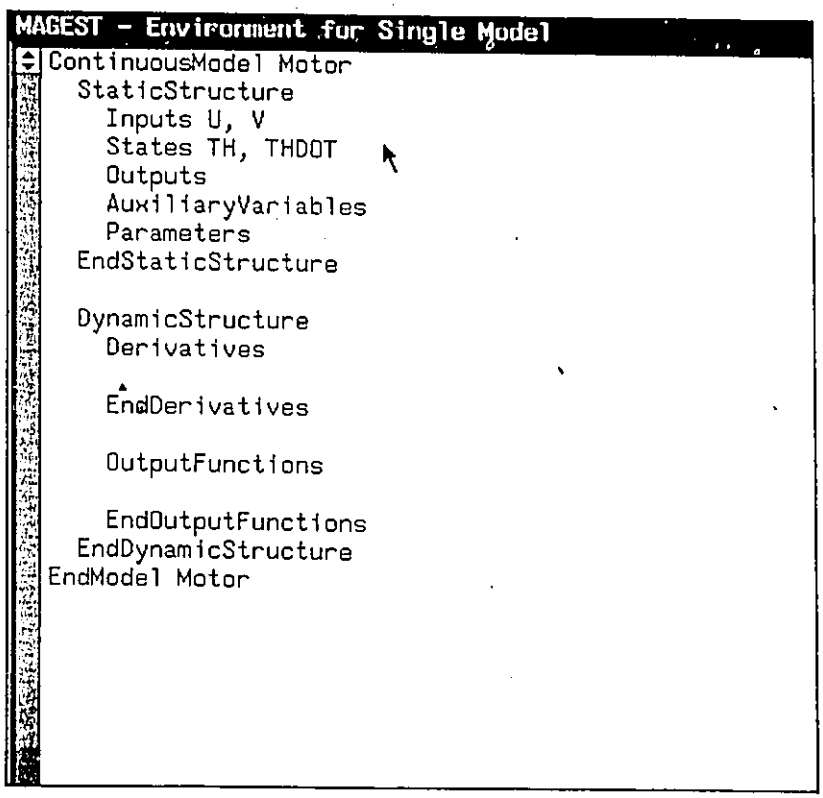

Figure 4.1.8 Typing in the text editor window.

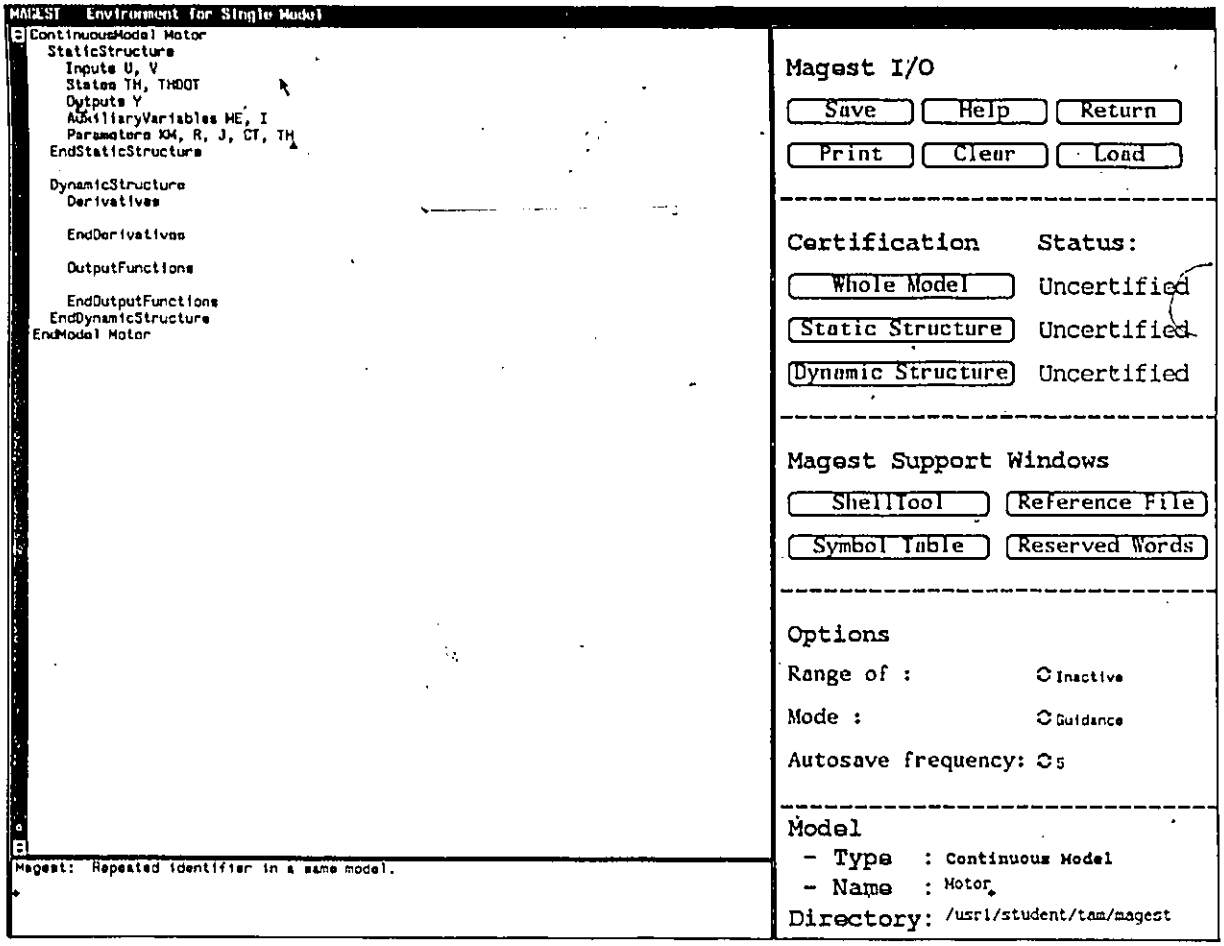

Figure 4.1.9 Error occurs in environment of single model.

After entering the parameters, an error occurred. It tells us the identifier "TH" has been defined more than once. If we look at the symbol table, the variable "TH" has been defined as a state variable. Therefore, we should decide which type of variable it should be. Finally, we decide to cancel "TH" in the parameter declaration.

During the time when the modeller is entering the specifications, the certification statuses are telling him the condition of the model as certified lastly. In guidance mode, the certification status shows the lack of errors from the beginning of a model until the present line as indicated by the cursor location. At any time, the modeller can activate one of the three certification buttons, to get the current status of acceptability, i.e., certification of parts or of the whole model.

If the modeller wants to see whether the model is completely certified, he has to activate the **Whole Model** button.

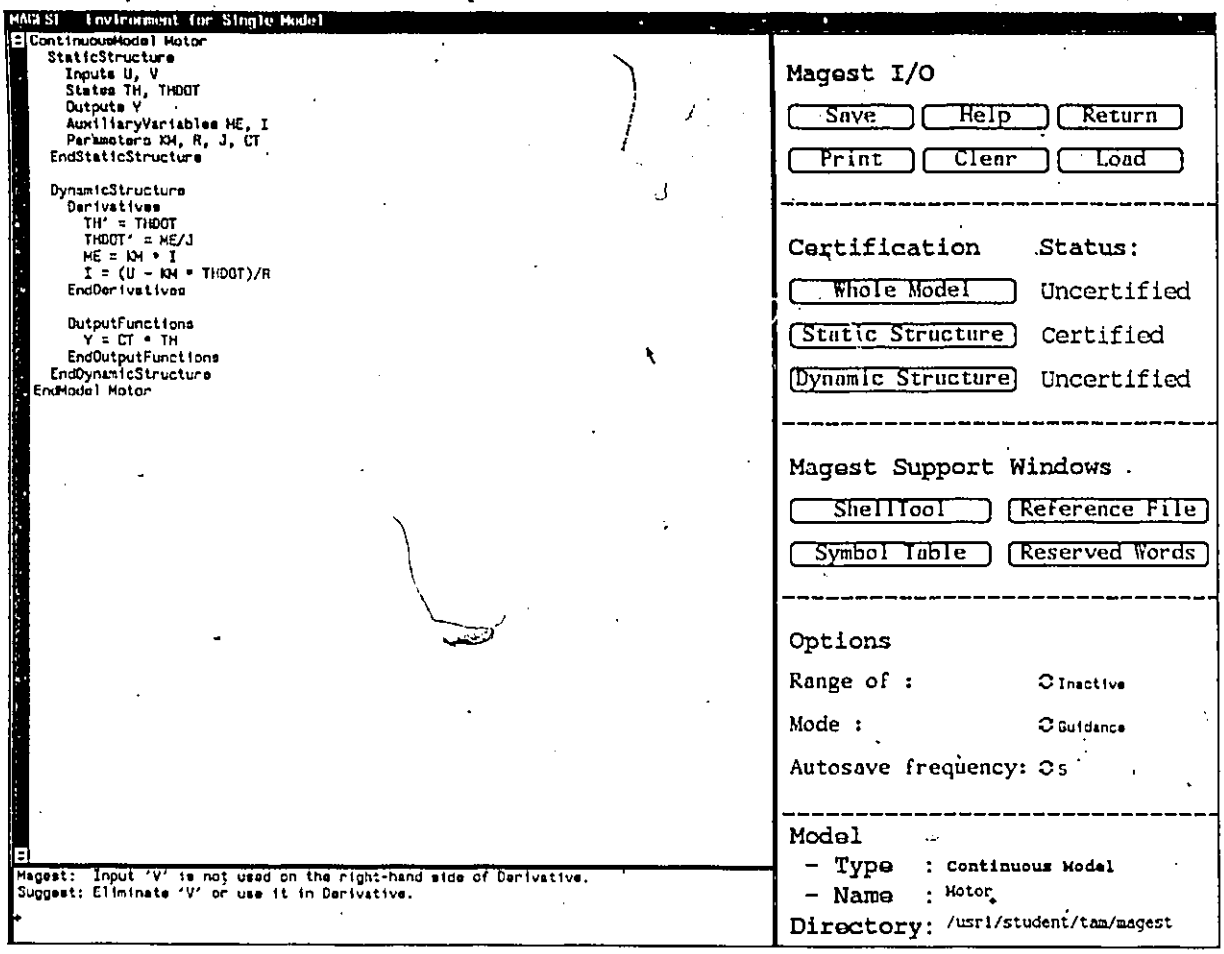

Figure 4.1.10 Semantic error occurred in continuous model.

In Figure 4.1.10, the model can not be certified because input "V" has not been used in the derivative section of the dynamic structure. Magest advises the modeller to either eliminate "V" or to use it in the derivative section. The modeller decides to erase the extra input variable "V". After he erases it, he presses the **Whole Model** button again. This time, the model is certified. This means that the specification of this continuous model is completely correct. As well as the quality of this model has been ensured by the Magest system.

As shows in Figure 4.1.11, this is the correct version of continuous

model "Motor". The correction of this model has been assisted by the Magest system, so that we have an enhance model free of syntectic errors. Furthermore, the model does not violate the semantic facts and rules of Gest environment.

```
ContinuousModel Motor
  StaticStructure
    Inputs U
    States TH, THDOT
    Outputs Y
    AuxiliaryVariables ME, I
    Parameters KM, R, J, CT
  EndStaticStructure

  DynamicStructure
    Derivatives
      TH' = THDOT
      THDOT' = ME/J
      ME = KM * I
      I = (U - KM * THDOT)/R
    EndDerivatives

    OutputFunctions
      Y = CT * TH
    EndOutputFunctions
  EndDynamicStructure
EndModel Motor
```

Figure 4.1.11 Enhanced version of continuous model "Motor".

4.2 Coupled Model Specification

To specify a coupled model, the modeller must have all the component models individually specified and certified before hand.

After this preparation, the user can activate the **coupled model** button (as shown in Figure 3.2.3). The system generates a template (as shown in Figure 4.2.1), to specify the external variables of the coupled model. The modeller specifies the external inputs, and external outputs. If there is any syntactic or semantic error, the system will advise the modeller in the message window.

Then the modeller specifies the names and types of the component models. As explained in section 3.5.8, the keywords associated with the type of component models can be entered by system functions to eliminate typing errors.

Once the modeller activates the **External Coupling** button, two activities are done by the system. (see section 3.4.1). If any component model refers to any macro model, the name(s) of such macro model(s) is(are) provided in a list of "MacroModels". The template for the external coupling is generated by the system. (see section 3.4.1). The modeller completes the right hand sides of the external couplings. If there are any syntactic or semantic errors, the system advises the user accordingly.

Afterwards, the user presses the **Internal Coupling** button for the system to prepare the template for internal coupling (see section 3.4.2). The user completes the right hand sides of the specifications. If there are any syntactic or semantic errors, the system advises the user accordingly.

Otherwise, the coupling is finished and the certification status of the coupled model becomes "Certified".

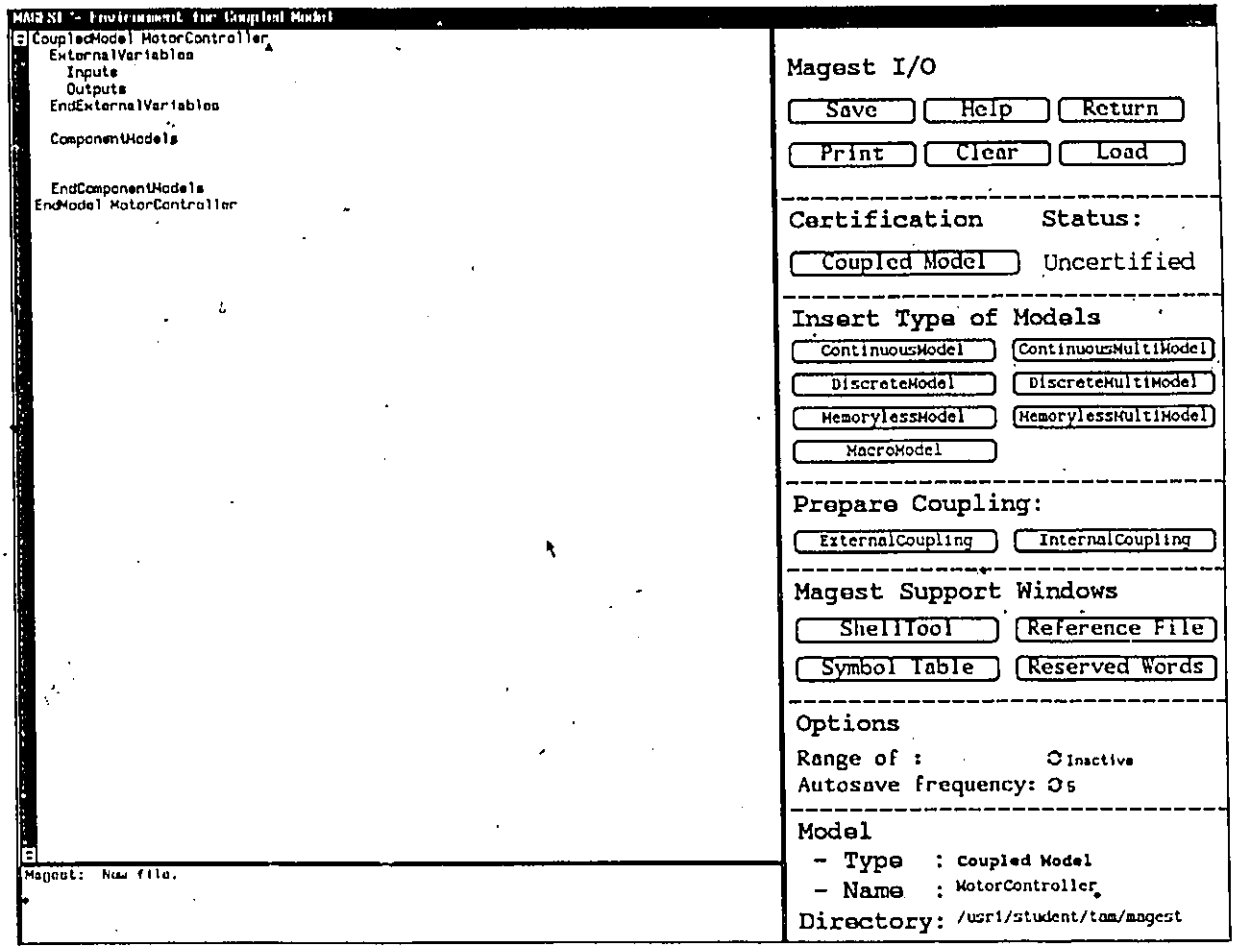

Figure 4.2.1 Environment for a coupled model.

4.3 Multi-rate model Specification

The example of a multi-rate model is taken from Barrett (1988) and adopted from Palusinski (1985). Figure 4.3.1 is a complete specificaton of the multi-rate model. Figure 4.3.2 and 4.3.3 represent Fast and Slow component models. The specification of a multi-rate model starts with the activation of **Multi-rate Model** button in the main menu, as shown in Figure 3.2.5. Then the following specifications are provided with the guidance of the Magest system:

- External variables
- Component models
- Macro models
- External couplings
- Internal couplings

The specification is similar to the specification of coupled models.

```
MultiRateModel ElectronicOscillator
  ExternalVariables
    Outputs S, X3
  EndExternalVariables

  ComponentModels
    FastContinuousModel Fast
    SlowContinuousModel Slow
  EndComponentModels

  ExternalCouplings
    ExternalOutputs
      ElectronicOscillator.S <-- Slow.S
      ElectronicOscillator.X3 <-- Slow.X3
    EndExternalOutputs
  EndExternalCouplings

  InternalCouplings
    Fast.S <-- Slow.S
    Fast.X3 <-- Slow.X3
    Slow.X2 <-- Fast.X2
  EndInternalCouplings
EndModel ElectronicOscillator
```

Figure 4.3.1 Multi-rate model "ElectronicOscillator".

```
ContinuousModel  Fast
   StaticStructure
     Inputs
       S,
       X3
     States
       Y1,
       Y2,
       X1,
       X2
     Outputs
       X2
     Parameters
       R1,          (* Resistance in kOhms          *)
       R2,          (* Resistance in kOhms          *)
       C1,          (* Capacitance in microFarads   *)
       C2           (* Capacitance in microFarads   *)
   EndStaticStructure
   DynamicStructure
     Derivatives
       Y1' = Y2
       Y2' = -64*Y1 + 7.5*(2 + S - Y1**2.0) * Y2
       X1' = 2*Y1*Y2/R1 + (X2-X1)/R1*C1
       X2' = X1/R2*C1 - (1/R2*C1 + 1/R2*C2)*X2 + X3/R1*C1
     EndDerivatives
   EndDynamicStructure
EndModel  Fast
```

Figure 4.3.2 ContinuousModel "Fast".

```
ContinuousModel  Slow
   StaticStructure
      Inputs
         X2
      States
         X3,
         X4,
         X5,
         X6,
         V,
         S
      Outputs
         X3,
         S
      Parameters
         R3,      (* Resistance in kOhms          *)
         R4,      (* Resistance in kOhms          *)
         R5,      (* Resistance in kOhms          *)
         R6,      (* Resistance in kOhms          *)
         C2,      (* Capacitance in microFarads   *)
         C3,      (* Capacitance in microFarads   *)
         C4,      (* Capacitance in microFarads   *)
         C5,      (* Capacitance in microFarads   *)
         C6       (* Capacitance in microFarads   *)
   EndStaticStructure
   DynamicStructure
      Derivatives
         X3' = X2/R3*C2 - (1/R3*C2 + 1/R3*C3)*X3 + X4/R3*C3
         X4' = X3/R4*C3 - (1/R4*C3 + 1/R4*C4)*X4 + X5/R4*C4
         X5' = X4/R5*C4 - (1/R5*C4 + 1/R5*C5)*X5 + X6/R5*C5
         X6' = X5/R6*C5 - (1/R6*C5 + 1/R6*C6)*X6
         V' = X6/C6
         S' = 0.02*Abs(36 - 5.29 * Sqrt(Abs(V))) ** 1.2 *
              Sign(Abs(36 - 5.29 * Sqrt(Abs(V))) - 0.02*S)
      EndDerivatives
   EndDynamicStructure
EndModel  Slow
```

Figure 4.3.3 ContinuousModel "Slow".

Chapter 5

SEMANTIC FACTS AND RULES USED IN MAGEST

5.1 Introduction to Semantic Facts and Rules

The semantic facts and rules present in this chapter is the knowledge base implemented in the Magest system. These facts and rules are based on the methodology of modelling and simulation provided by the Gest language. Most of the rules are taken from Ören and Sheng (1988). In addition to these facts and rules, there are over ten others which have been added to the system. They are included in this chapter. An example of a fact is as follows:

MS.1 FACT In a list of identifiers, there must be no duplication of names.

A semantic fact describes the fact that it must always be applied in modelling specification. Each rule and fact has been classified by a division code and number. The division code refers to its section. Table 5.1 below summarizes the division codes:

Division Code	Explanation
GG	General facts and rules for guidances mode.
GC	General facts and rules for certification mode.
MS	Facts and rules for static structure specification in modelling.
MD	Facts and rules for dynamic structure specification in modelling.
MC	Facts and rules for coupled model and multi-rate model specifications.

Table 5.1 Division codes of semantic facts and rules.

An example of a rule is as follows:

GC.1 RULE IF a user entry field is left blank,
THEN Magest expects the user to either delete the line or provide additional information.

The **IF** part of the rule is known as the *premise*, which is a logical combination of functions operating on facts. The **THEN** part of the rule is known as the *action*, which is a sequence of functions with side effects such as asserting a value for a fact or initiating a user interaction (Symonds, 1986). If the modeller violates these facts or rules, Magest informs him/her by giving an error message. The modeller must make the necessary correction in order to continue working with his/her model. Details of error messages are listed in appendix C.3.

The relationale of implementing these facts and rules is to provide built-in quality assurance to the model specification. This has been discussed in section 1.2. Facts and rules in the Magest system can be applied in any advanced simulation modelling environment and are presented in the following categories:

1) General,
2) Static Structure of Models,
3) Dynamic Structure of Models, and
4) Coupling

In this thesis only the semantic facts and rules for model specifications are implemented and documented. The semantic facts and rules for the parameter and experimentation specifications are being developed by Ören and Aşıkhan.[1]

[1] Ören and Aşıkhan. (Personal Communication)

5.2 Rules and facts in general

In guidance mode:

GG.1 FACT A user entry field is allowed to be left blank.

In certification mode:

GC.1 RULE IF a user entry field is left blank,
THEN Magest expects the user to either delete the line or provide additional information.

5.3 Facts and Rules Relating to the Static Structure of Models

MS.1 FACT In a list of identifiers, there must be no duplication of names.

MS.2 FACT Between any two lists of identifiers, there must be no duplication of names except for output variables which may be identical to state variables or auxiliary variables.

MS.3 FACT A continuous model requires at least one state variable.

MS.4 FACT A discrete model requires at least one state variable.

MS.5 FACT A memoryless model cannot have any state variables.

MS.6 FACT A memoryless model must have at least one input variable.

MS.7 FACT At least one output variable must be declared for any type of model.

MS.8 FACT A macro model must have at least one input variable.

MS.9 FACT Any auxiliary parameter necessitates the existence of at least one parameter.

MS.10 FACT An auxiliary parameter must be specified in an arithmetic statement.

MS.11 FACT An auxiliary parameter must be a function of at least one parameter.

MS.12 FACT Any interpolated variable necessitates the existence of at least one tabular function.

MS.13 RULE IF there is a tabular function which is not used in the specification of an interpolated variable,
THEN MAGEST requires either its elimination or the existence of an interpolated variable which refers to it.

MS.14 FACT A constant may be used before its declaration. However it has to be later declared in the block of constants.

MS.15 FACT A constant which appears on the right hand side of a constant assignment must be specified previously.

MS.16 FACT For each variable or parameter, the "RangeOf" specification may exist at most one.

MS.17 FACT A constant, auxiliary parameter, or interpolated variable must be specified exactly once.

5.4 Rules and Facts Relating to the Dynamic Structure of Models

MD.1 RULE IF the static structure is not certified,
THEN one cannot work on the dynamic structure.

MD.2 FACT A continuous model has a derivative block.

MD.3 FACT A discrete model has a state transition block.

MD.4 FACT A memoryless model has an output block.

MD.5 FACT A macro model has an output block.

MD.6 FACT All identifiers in a dynamic section must have been declared in the static section of the same model.

MD.7 FACT In memoryless model or in macro models input variables must be used on the right-hand side of output functions.

MD.8 FACT In memoryless models or in macro models, input variables may appear in condition specifications.

MD.9 FACT Input variables cannot be assigned values in the dynamic section.

MD.10 FACT In continuous models, an input variable must appear on the right-hand side of at least one derivative specification.

MD.11 FACT In discrete models, an input variable must appear on the right-hand side of at least one state transition function.

MD.12 FACT In the output block or in the update block of a continuous model or a discrete model, an input variable may appear only in a condition specification.

MD.13 FACT In the derivative block, the derivative of every state variable must be specified.

MD.14 RULE IF the derivative of a state variable is given in a high order,
THEN the lower order derivatives cannot be specified by the user.

MD.15 FACT In the dynamic section of a continuous model, state variables or their derivatives may appear on the right-hand side of assignments or in conditions.

MD.16 FACT In the dynamic section of a discrete model, state variables may appear on the right-hand side of assignments or in conditions.

MD.17 RULE IF in a continuous model or a discrete model there is at least one output variable distinct from state or auxiliary variables,
THEN there is an output function block.

MD.18 RULE IF an output block exists,
THEN each output variable distinct from state or auxiliary variables must have one and only one output function specification.

MD.19 FACT An auxiliary variable must be specified in, at most one assignment statement in the derivative block or in the output block.

MD.20 FACT An auxiliary variable cannot appear on the right-hand side of the assignment statement where it is specified.

MD.21 FACT An auxiliary variable specified in the derivative or output function block can appear on the right-hand side of an assignment statement or in a condition of the same and following blocks in the dynamic section.

MD.22 FACT Constants, parameters, and auxiliary parameters must be used at least once in the dynamic section and/or in the experiment specification

MD.23 FACT An interpolated variable must be used at least once in a dynamic section.

MD.24 FACT A macro which appears in the "macros used" list must be used at least once in a dynamic section.

MD.25 FACT In a continuous model, the derivative of a state variable may appear on the right hand side and/or condition specification of an output function and/or an update block.

MD.26 FACT In the update block, only the values of state variables and parameters can be reassigned.

MD.27 FACT The input of a macro model can be input, state, output, constant, parameter, auxiliary parameter, auxiliary variable and interpolation variable of the host component model from where the macro model is activated.

MD.28 FACT The output of a macro model can be state, output and auxiliary variable of the host component model from where the macro model is activated.

5.5 Rules and Facts Relating to Coupling

MC.1 FACT An external input variable can be connected to one or more internal input variable(s).

MC.2 FACT An external output variable can be connected to only one internal output variable.

MC.3 FACT For each input/output connection in the internal coupling specification, the component model which is specified as the source of input must be part of the coupled component models.

MC.4 FACT For each input/output connection in the internal coupling specification, the output variable which is specified as the source of the input must appear in the list of the output variables of the component model which is specified in the coupling as the source of the input.

MC.5 RULE IF an input variable of a component model has been connected in the external coupling specification,
THEN the input variable cannot appear in the internal coupling specification.

MC.6 RULE IF a coupled model has no input variable declared in the external variable specification,
THEN there is no external input block in the external coupling specification.

MC.7 RULE IF there is at least one component model with at least one input variable which is not connected in the external coupling specification,
THEN there exists internal coupling block.

MC.8 RULE IF there is one macro declared in a component model,
THEN the "MacroModels" block exists in the coupled model or in the multi-rate model specification.

MC.9 RULE IF a component model in a "ComponentModels" specification has not been certified,
THEN one can not work beyond component model specification.

Chapter 6

ARCHITECTURE OF THE MAGEST SYSTEM

This chapter presents the architecture of the Magest system in a finite state machine representation. The finite state machine representation of the architecture of an advanced simulation environment is discussed in Ören (1986a, b). The advantage of this representation is that it provides a top-down representation of the system. It also provides a flexible way to represent and modularize the architecture. Furthermore, a robust system is realized by this method.

The overall system architecture of the Gest environment is given in Figure 6.1. This figure reflects the architecture of the Gest Executive-Main Menu. There are two grey rectangle boxes in this figure. They indicate the system which has been implemented in this thesis. The detailed system architecture of the environment for single models, coupled models and multi-rate models is shown in Figures 6.2-6.12.

Figure 6.2, 6.3 and 6.4 elaborate the system architecture of single models, coupled models and multi-rate models respectively. Each box in this type of architecture may contain a detailed sub-architecture. If so, it is elaborated on in the next Figure. For instance, Figure 6.5 is a elaboration of a text editor, which is used in Figures 6.2, 6.3 and 6.4.

There is a grey rectangle box added in each of the Figures 6.6-6.11. The architectures which are inside those boxes are elaborations from previous figures.

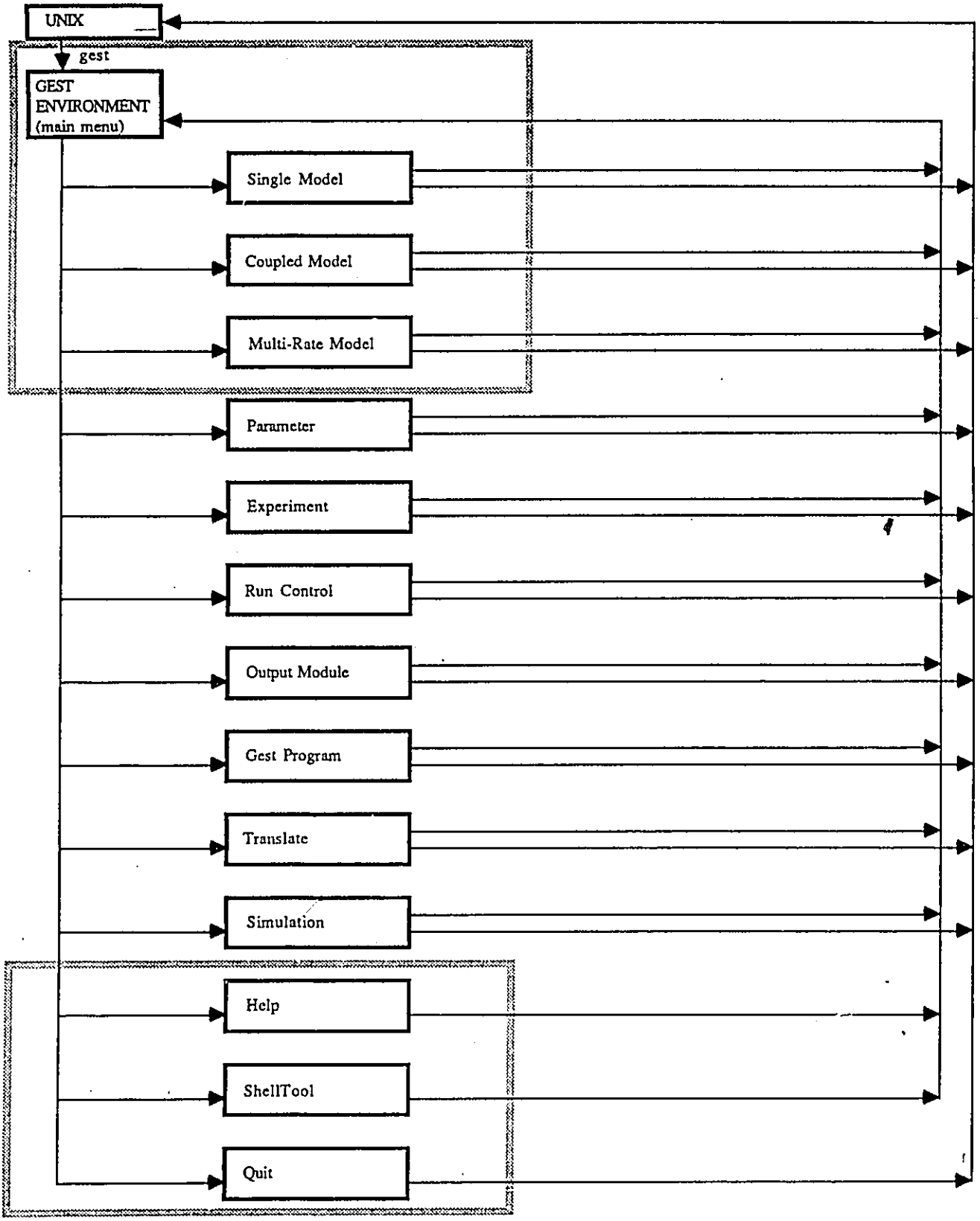

Figure 6.1 Gest Environment: Overall System Architecture

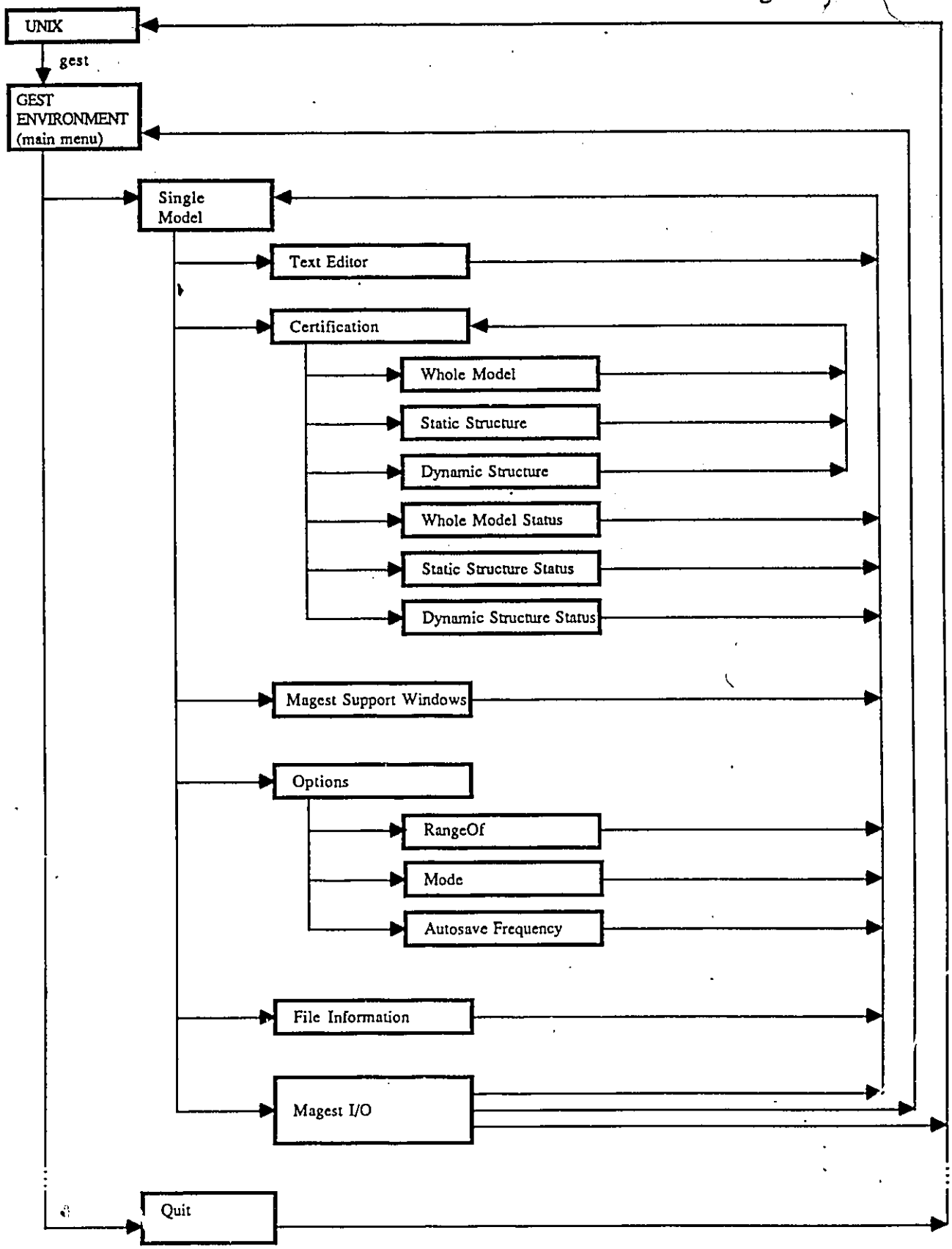

Figure 6.2 Magest Environment for Single Model: System Architecture

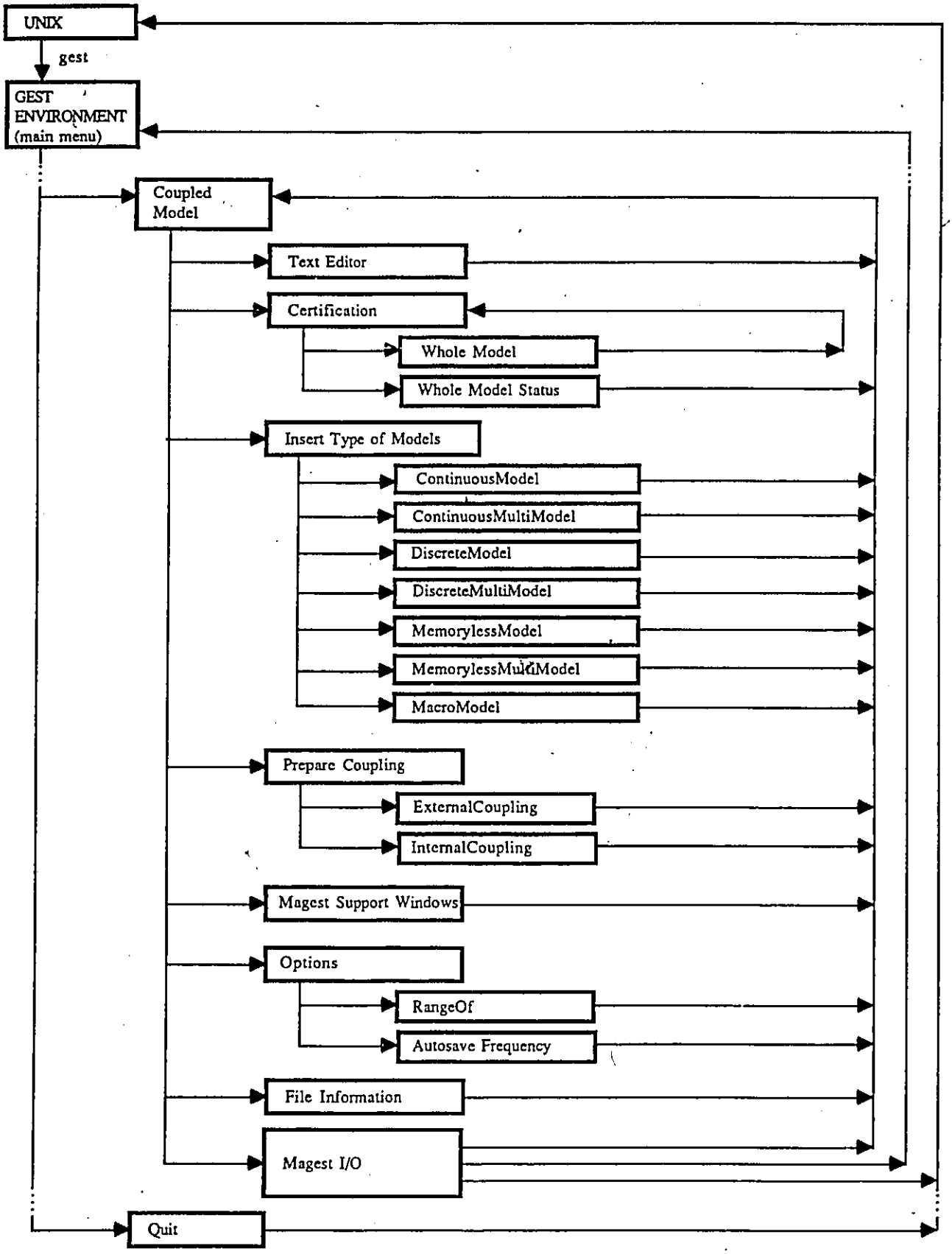

Figure 6.3 Magest Environment for Coupled Model: System Architecture

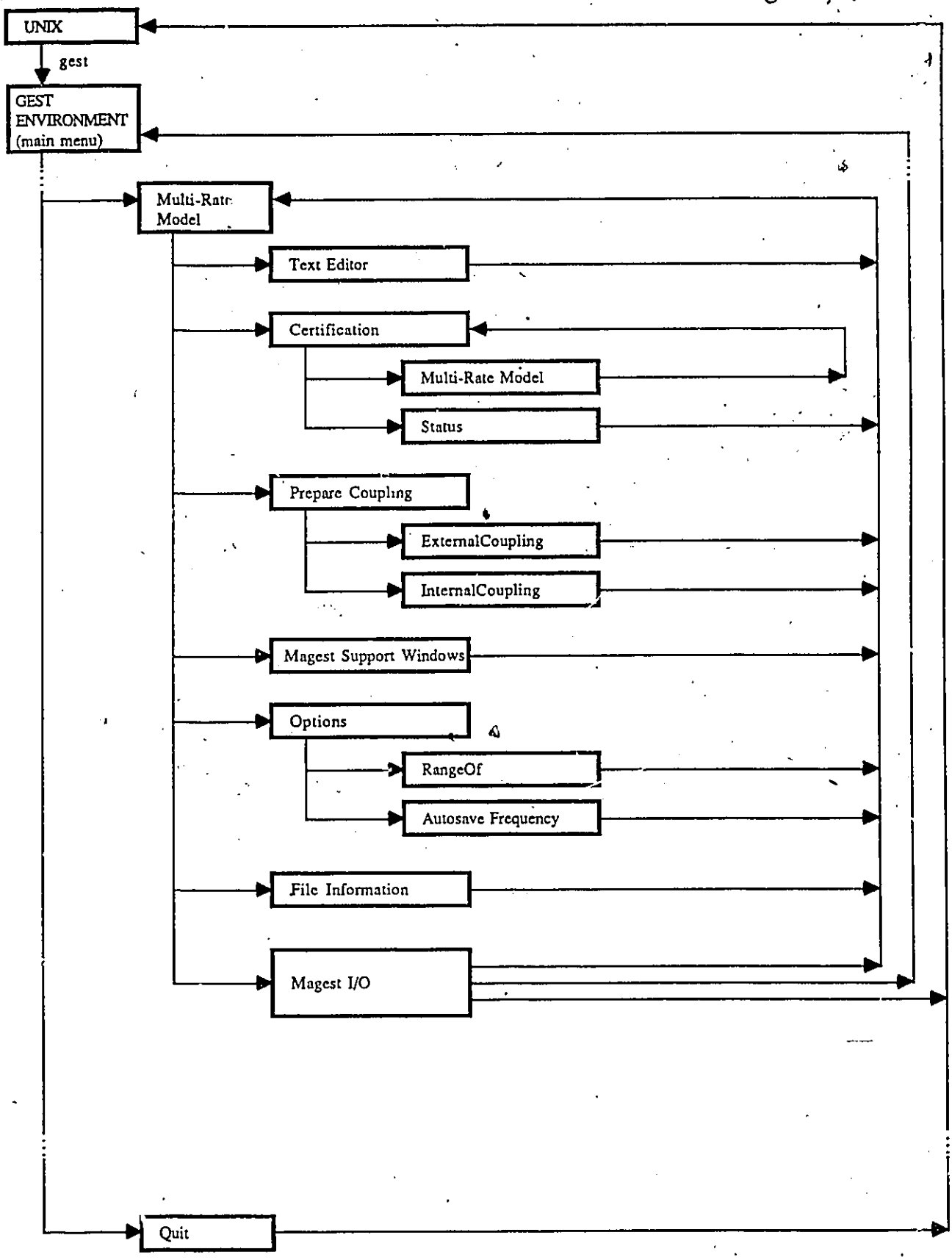

Figure 6.4 Magest Environment for Multi-Rate Model: System Architecture

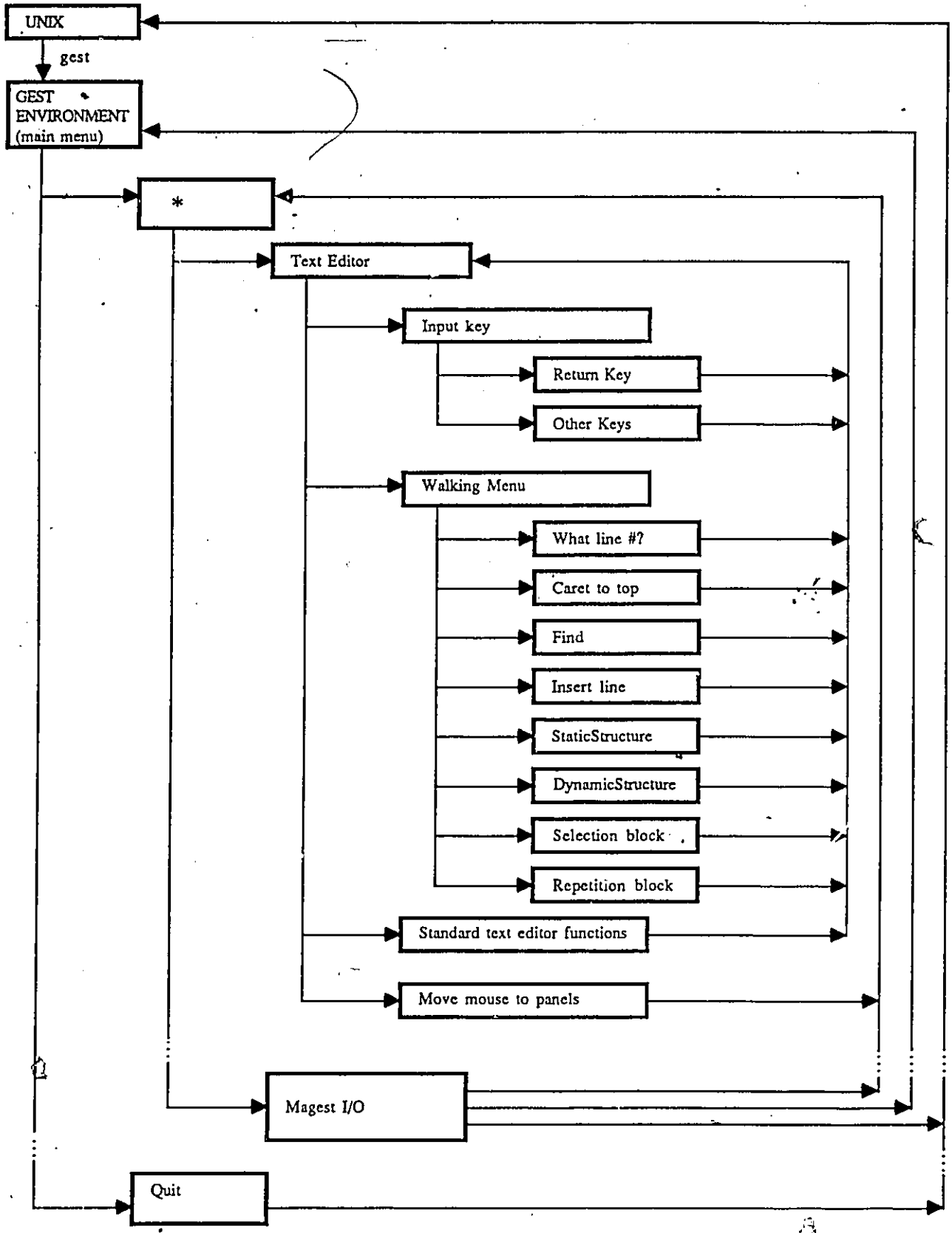

Figure 6.5 Magest Text Editor within the System Architecture
(* corresponds to Single Model, Coupled Model and Multi-Rate Model)

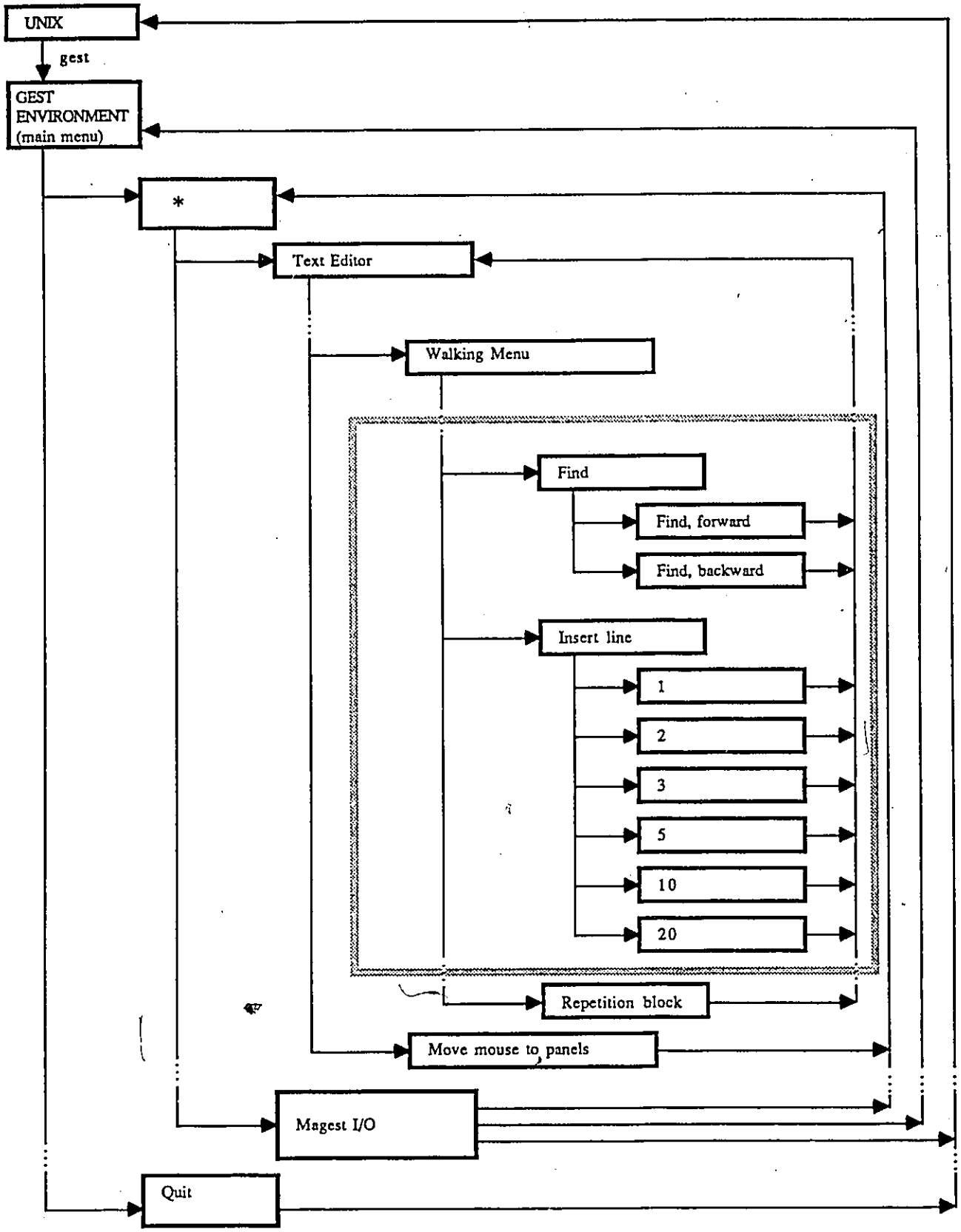

Figure 6.6 Find and Insert line within the System Architecture
(* corresponds to Single Model, Coupled Model and Multi-Rate Model)

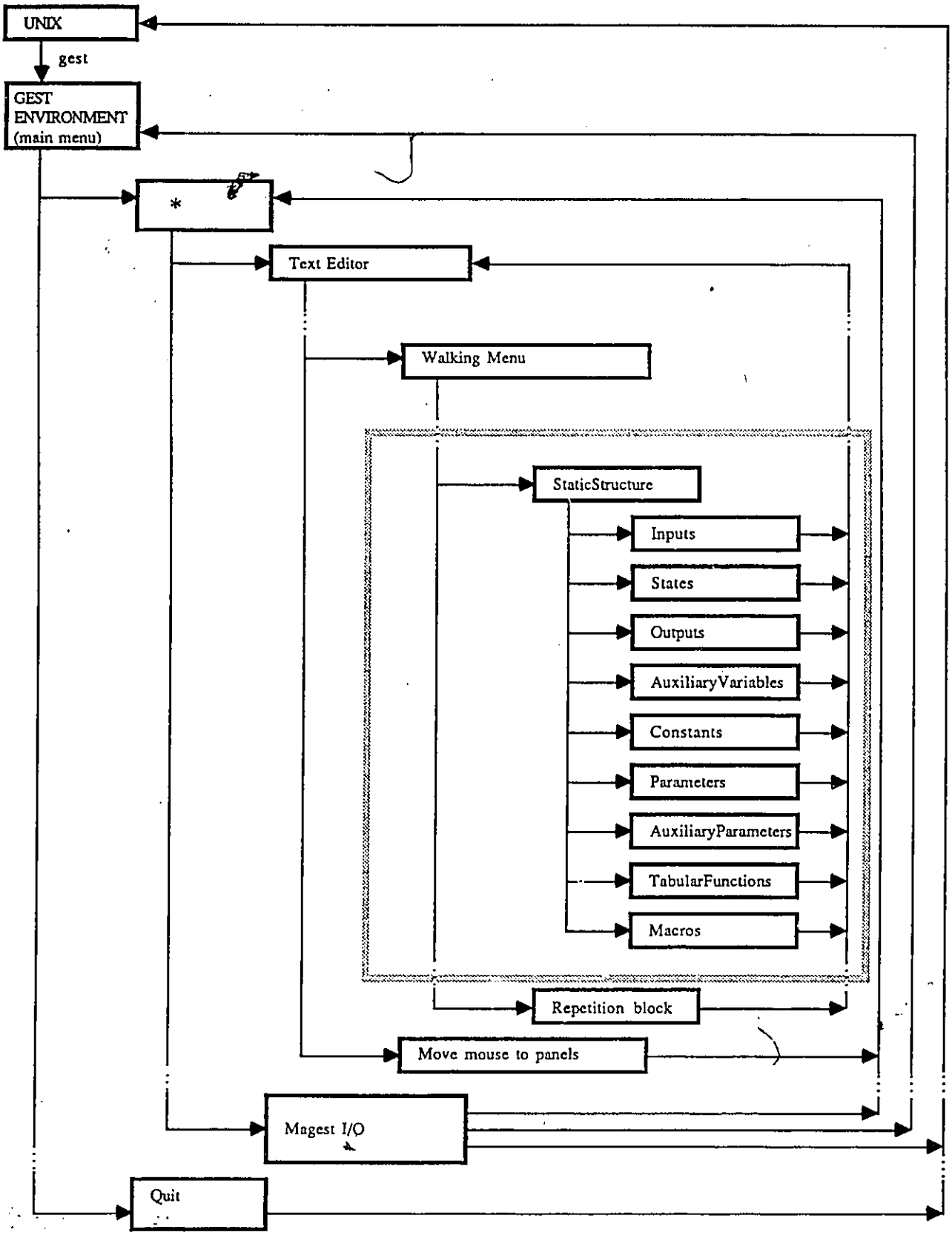

Figure 6.7 Static Structure within the System Architecture
(* corresponds to Single Model, Coupled Model and Multi-Rate Model)

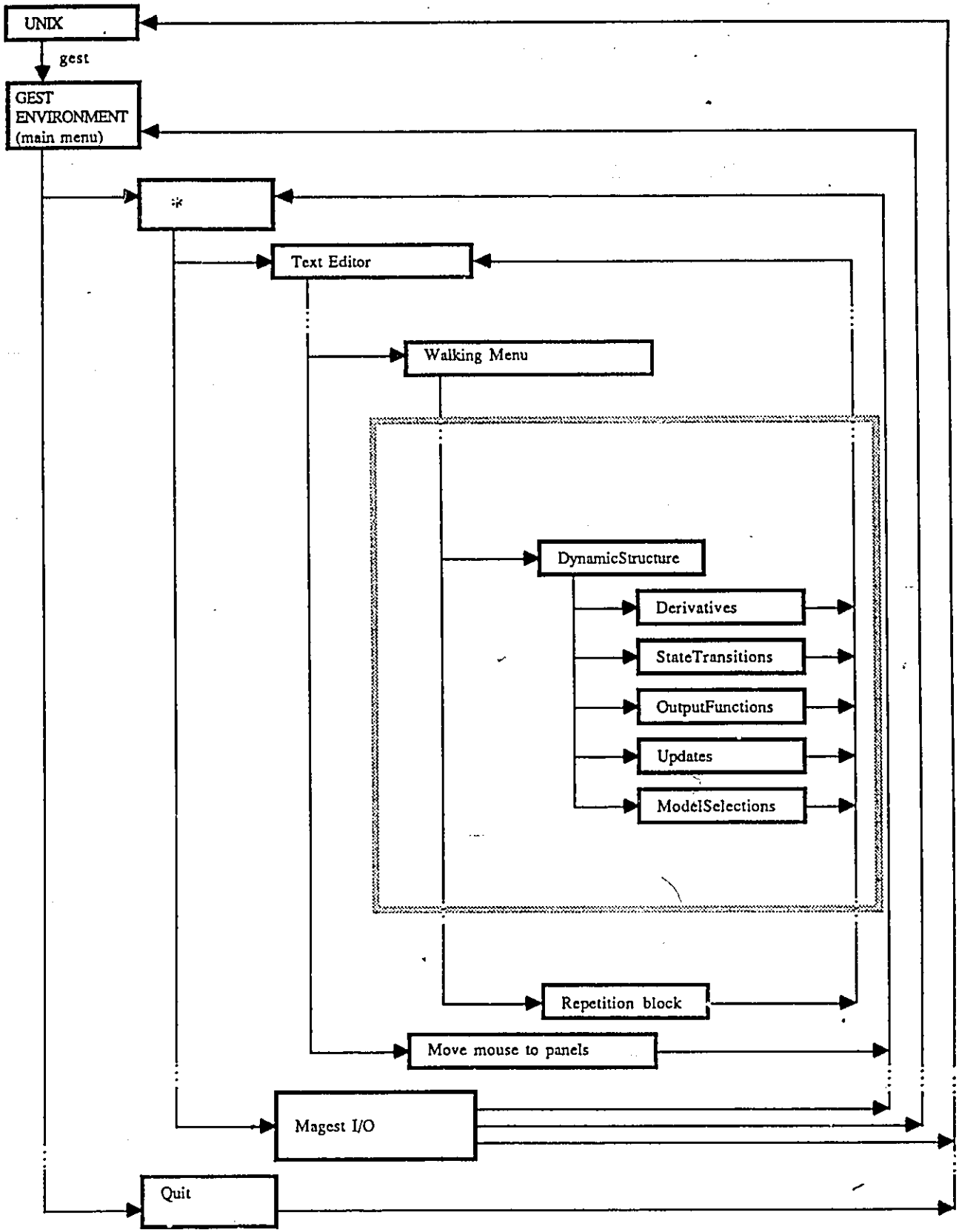

Figure 6.8 Dynamic Structure within the System Architecture
(* corresponds to Single Model, Coupled Model and Multi-Rate Model)

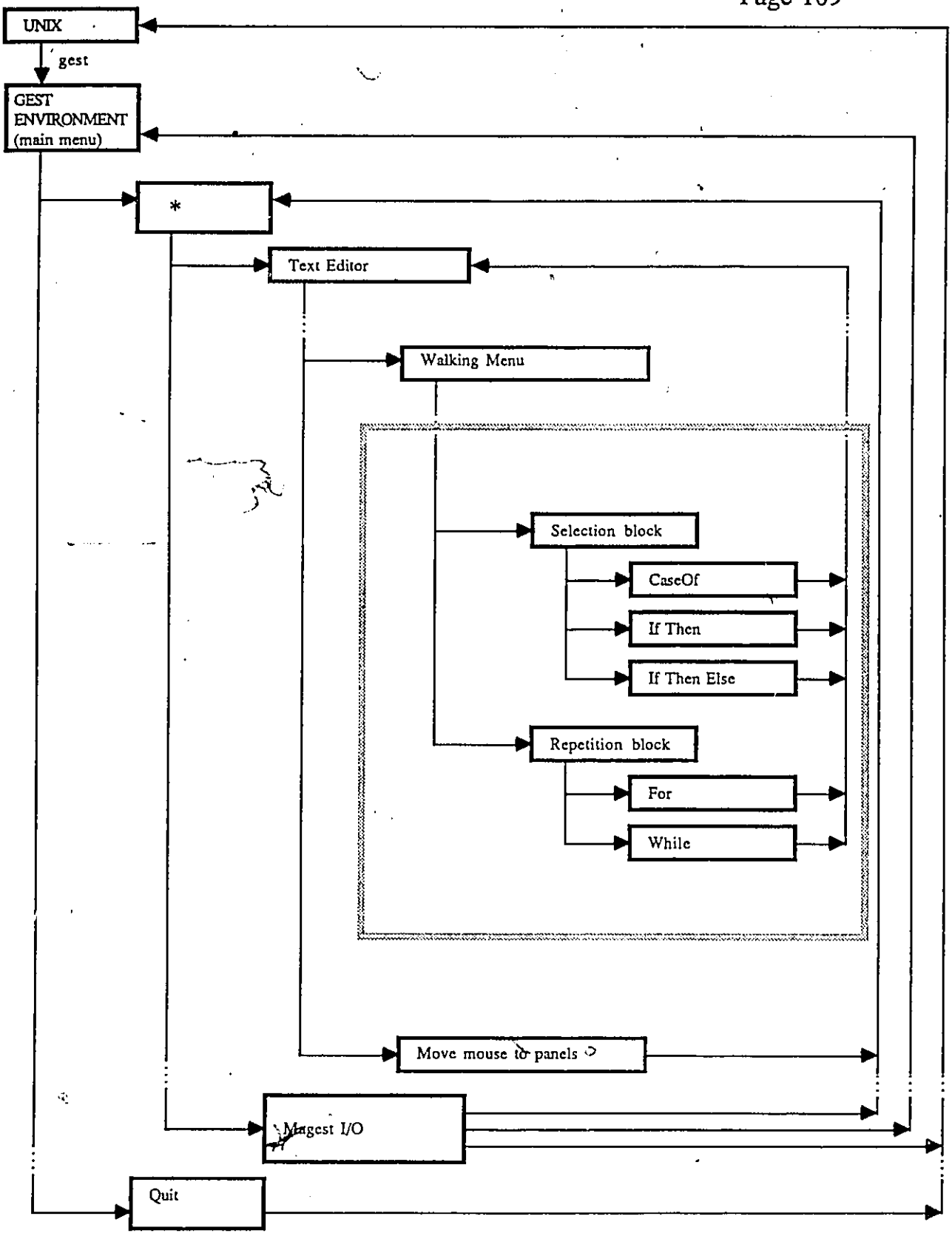

Figure 6.9 Selection block and Repetition block within the System Architecture
(* corresponds to Single Model, Coupled Model and Multi-Rate Model)

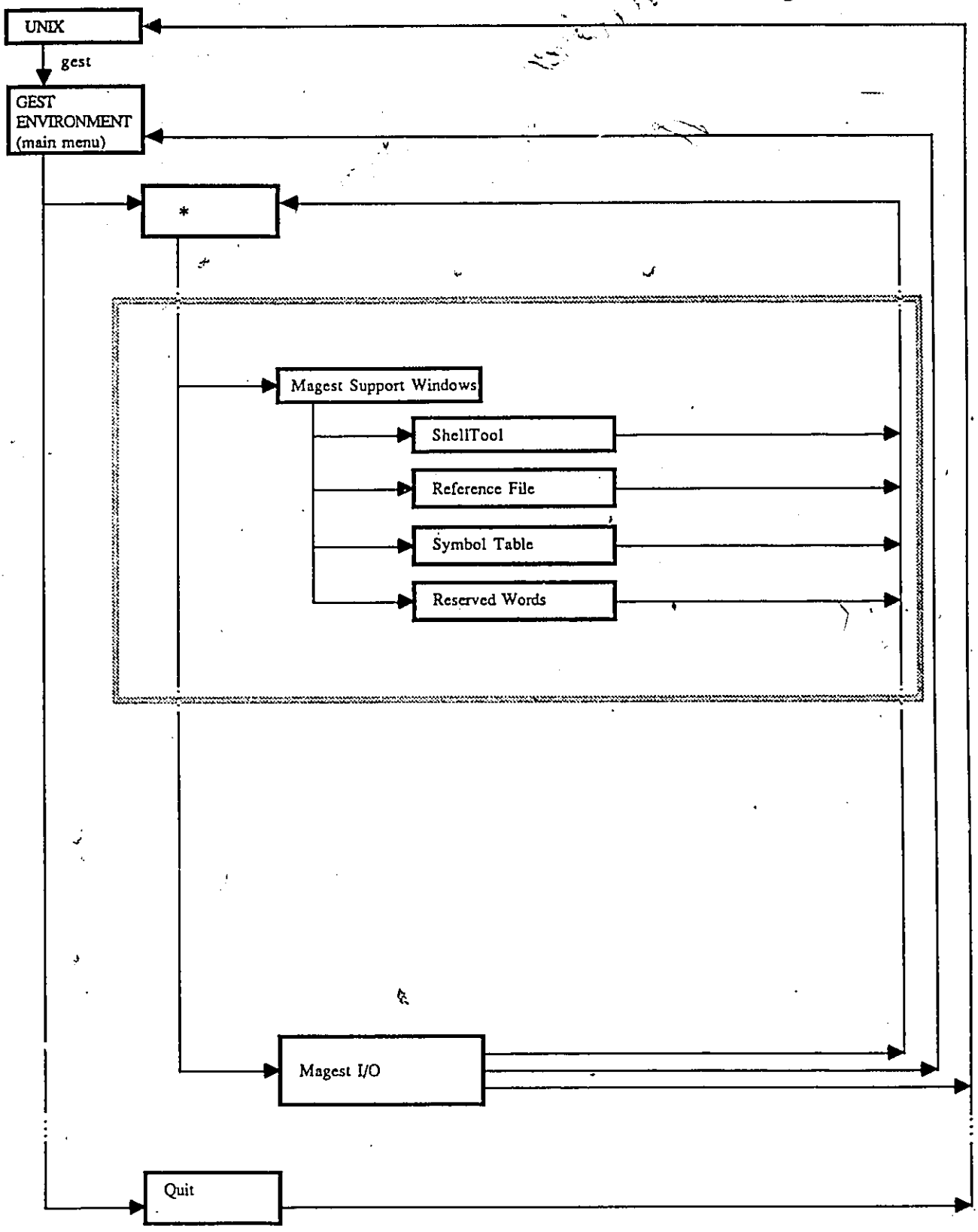

Figure 6.10 Magest Support Windows within the System Architecture
(* corresponds to Single Model, Coupled Model and Multi-Rate Model)

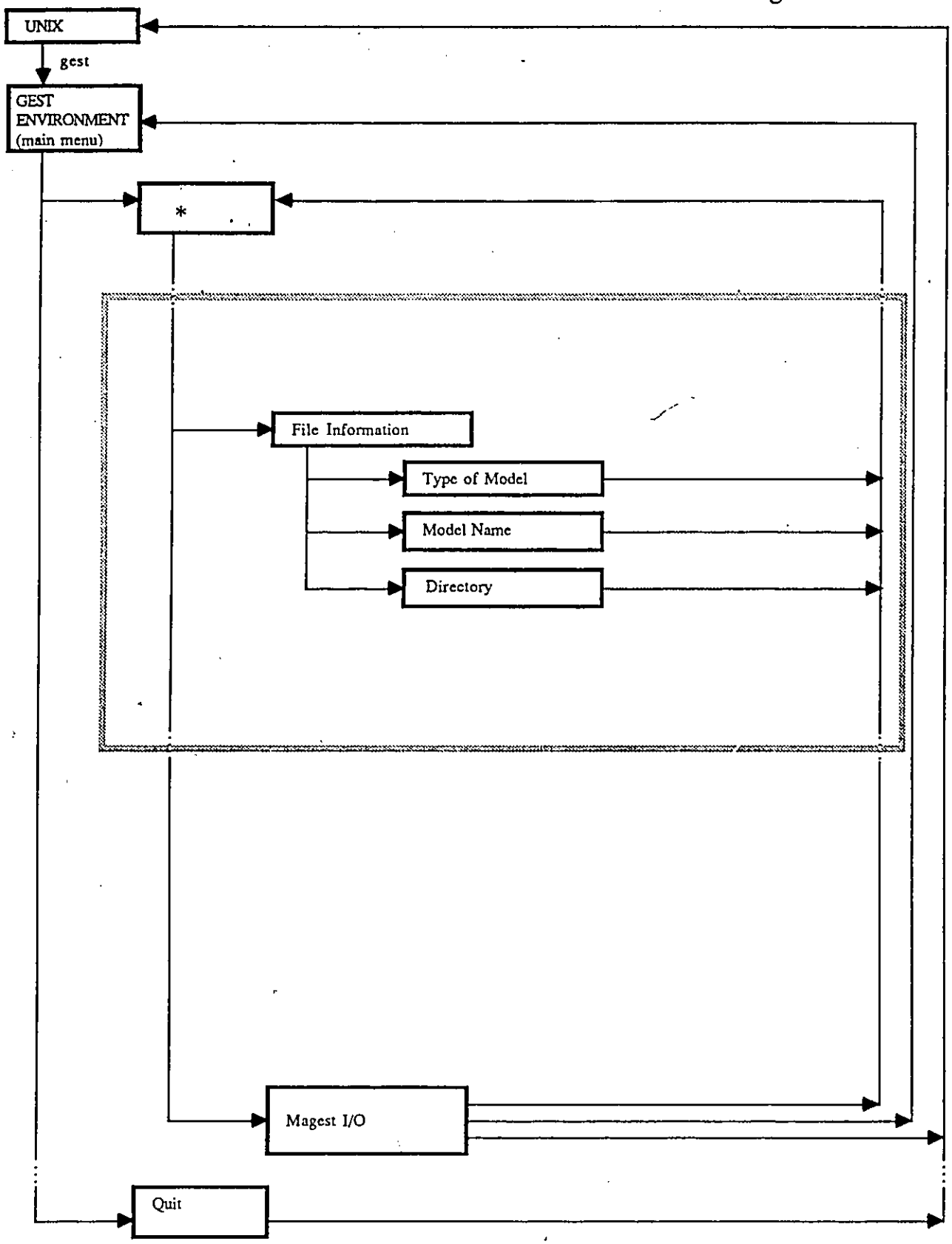

Figure 6.11 File Information within the System Architecture
(* corresponds to Single Model, Coupled Model and Multi-Rate Model)

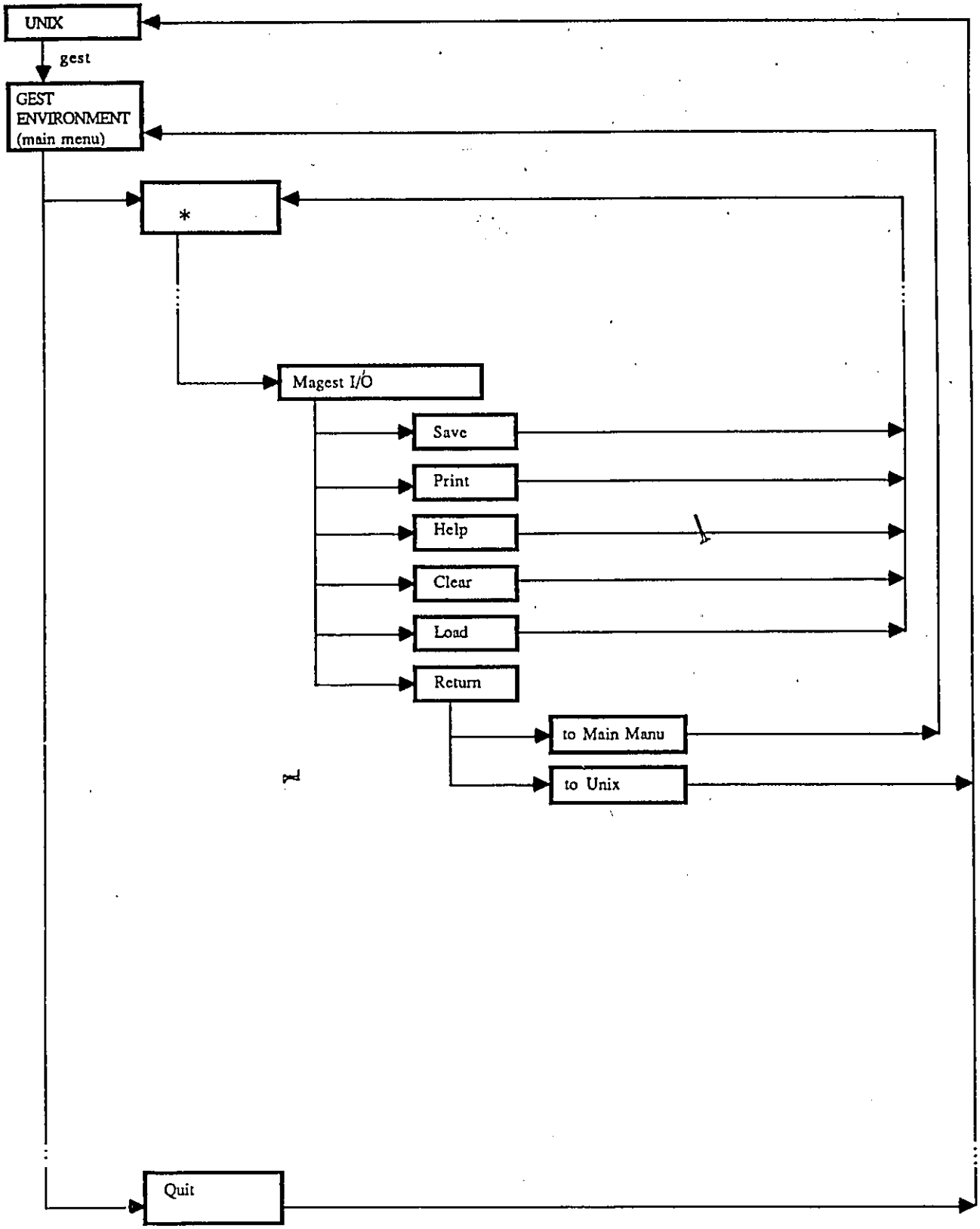

Figure 6.12 Magest I/O within the System Architecture
(* corresponds to Single Model, Coupled Model and Multi-Rate Model)

Chapter 7

IMPLEMENTATION ISSUES

7.1 Introduction to the Implementation of Magest

The Magest system is implemented in the C language. It requires the SunView window environment of the Unix operating system. Magest uses many window facilities provided by the SunView window environment. There are several SunView window functions and procedures called by the Magest system. Procedures and functions have also been developed for window management purpose.

The Magest source program is stored in seven files. The list of files and brief explanations are given in appendix A.2. The file called *define.c* is for global declaration uses. All of the constants, global variables and global data structures are declared in this file. The other six files are called *magest.c*, *pargest1.c*, *pargest2.c*, *semgest.c*, *confirm_gest.c* and *window_gest.c*. *Magest.c* contains the main program. The parser of the system is implemented in two parts, *pargest1.c* and *pargest2.c*. *Semgest.c* performs lexical analysis and semantic checks as well as database management functions. *Confirm_gest.c* handles pop up windows for confirmation use. *Window_gest.c* handles window management, message display, and several other functions. The total number of lines within these files is over fourteen thousand. Therefore, it is too long to list in this thesis. Instead, the headings of procedures and functions and relevant comments are copied from the original program listing. They are listed in appendix B. In addition, Table B.1 depicts the details of the files which

contain the procedures and functions. The appendix B, generated by a software tool called *proall*, contains for each one of the six files the names of the procedures and functions and brief documentation for each one of them. Detailed reading of appendix B can provide further information about the procedures and functions.

Figure 7.1.1 shows the internal structure of the Magest system. This diagram depicts the basic control flow and data flow of the system.

An event as shown in Figure 7.1.1 is any input from the user. It can be a keyboard input, a mouse movement, or a click of mouse's button. Events provide knowledge to the system.

As soon as an event occurs, it is analyzed by the corresponding notification procedure. The notification procedure is the highest-level controller of events. If the event does not require additional procedures to generate the solution, the notification procedure can analyze the event and directly provide solutions (Branch a). Otherwise, the notification procedure passes the control to another lower level procedure. There are two main possibilities. If model analysis is required, branch b is followed. Otherwise, branch c invokes either the walking menu functions or Magest I/O or Magest support windows. The notification procedures are also used to determine the correct knowledge, and make necessary updates of the knowledge-base.

Some user input (i.e. event from the point of view of the system) requires further information from the user. For instance, opening another window and/or giving addition information to update the knowledge-base

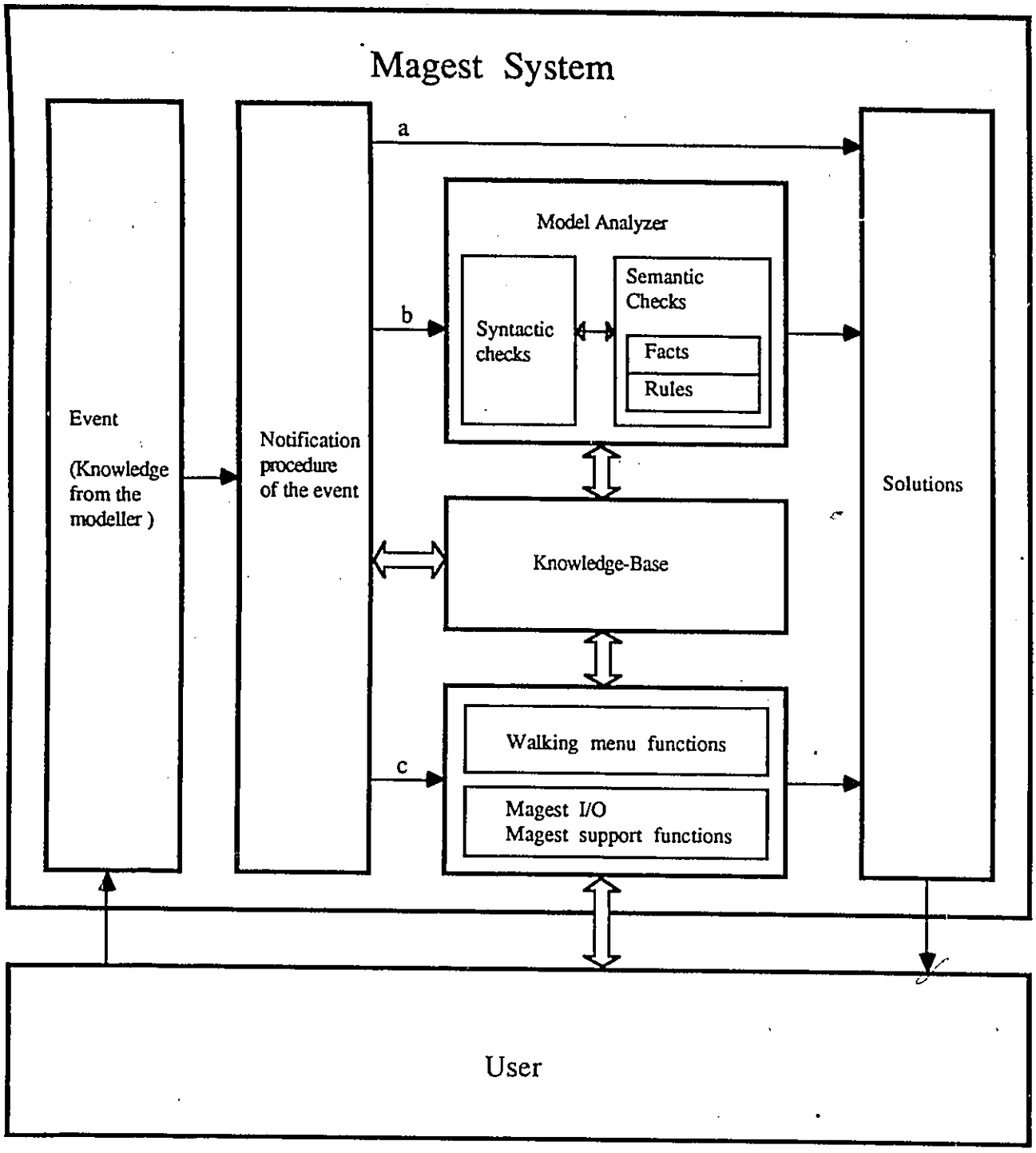

Figure 7.1.1 Anatomy of the Magest System

→ : Control flow

⟺ : Data flow

might be needed. For example, the "Print" procedure requires updates to the knowledge base about the print menu information, if the operation has been successfully performed.

The model analysis procedures are the main functions in the Magest system. They contain different types of procedures and functions such as procedures for lexical analysis, procedures for parsing, procedures for database management and procedures for system support. Entries into the text editor window in guidance mode, "Certification", "External Coupling" and "Internal Coupling" are examples of user input which require model analysis. They both require syntactic and semantic checks.

A solution is provided after the execution of necessary procedures and functions. The solution may be a dynamic template in the text editor window, a change of certification status in the system's control panel, a change of windows, or a message. The solution may also be displayed in a pop up window.

7.2 Knowledge Representation in Magest

There are two types of knowledge used in Magest for assisting the modeller. The first type of knowledge is based on the methodology of the Gest language and consists of syntactic and semantic facts and rules. The second type of knowledge that Magest uses is the incremental knowledge obtained from the modeller. Any user knowledge is scrutinized by Magest before being accepted.

The lexical analyzer is stored in the file called *semgest.c*. The parser is stored in the files called *pargest1.c* and *pargest2.c*.

The lexical analyzer separates the Gest model into tokens, and supplies tokens to the syntax analyzer. Then the syntax analyzer takes the tokens and determine the correctness of the syntax. The syntax analyzer is a recursive-decent parser (Tremblay, Sorenson, 1985). The reasons for choosing the recursive-decent parsing technique are based on the following advantages:

1) **Direct control from Magest** - As shown in semantic rule GG.1 in chapter 5, during the guidance mode some of the syntax may be left incomplete by the modeller, in order to continue further specification of a model. With a recursive decent parser, Magest can directly control the parsing of an unfinished specification.

2) **Ease of maintenance** - If there are any modifications in the Gest language definition, we can modify the parsing rules easily without affecting other rules already implemented in other parts of the parser.

3) **Effective communication with the semantic analyzer** - During the syntactic analysis phase, a recursive decent parser can transfer control

to a particular section of the semantic analyzer. Therefore, part of the semantic checks can be done as soon as the syntactic checks are finished.

The semantic rules are transformed into several semantic procedures and tables. A semantic procedure consists of some **if-then-else** and **case** statements to be applied under different semantic rule conditions.

Two tables, i.e., *semantic1_item* and *semantic2_item*, have special functions: *Semantic1_item* is a predefined table to verify whether a particular identifier is allowed to be specified in a particular structure or not. *Semantic2_item* is a small table to hold semantic attributes of a model and is updated based on the incremental knowledge received from the user.

Another table, i.e., the symbol table, also holds incremental knowledge received from the user. It is very useful for syntactic and semantic checks. Each item in the symbol table has the following attributes:

Name	Memory size	Explanation
name	9 words	Identifier name.
str	1 word	Nature, i.e. model type, input, state, etc. For example: 260 is continuous model.
type	1 word	Type of variable, i.e. integer, real, etc.
ref	1 word	The last position of the identifier.
rel_used	1 word	The identifier used in a particular semantic situation.
used	1 word	The identifier used in general.
dimension	1 word	A dimension variable or not.

Table 7.1 Description of an item in symbol table.

The **identifier** name is the primary key in the symbol table.

The *rel_used* is incremented by one when a particular situation happens. It is used to associate the name with a semantic rule. For example, the semantic rule MD.14 from chapter 5 would be implemented by the following algorithm:

<table>
<tr><td colspan="3">if the derivative of a state variable appears at the left hand side of the derivative block</td></tr>
<tr><td></td><td>then</td><td>if the rel_used for this state variable is equal to zero (i.e., the derivative of this state variable has not been specified.)</td></tr>
<tr><td></td><td></td><td><table><tr><td>then</td><td>increment rel_used by one</td></tr><tr><td>else</td><td>Magest informs the user that the derivative of the state variable has already been specified in the derivative section (message #157, see appendix C)</td></tr></table></td></tr>
<tr><td></td><td></td><td>end if</td></tr>
<tr><td colspan="3">end if</td></tr>
</table>

Figure 7.2.1 Algorithm to implement semantic rule MD.14 of chapter 5.

Before entering a dynamic structure (including different dynamic structures of a multi-model) the attribute *rel_used* is cleared to zero.

7.3 Window Structure and Management

Figure 7.3.1 shows the hierarchical representation of the main window structure in Magest. Each frame in the window structure is the base of a window. The Magest frame is the base frame (Sun Microsystems, 1986b). All of the other frames are subframes of the Magest frame. Table 7.2 below describes each of the subframes:

Name of subframe	Explanation
Main frame	The main menu - Gest executive.
Model frame	The environment of single models, coupled models and multi-rate models.
Gmf frame	The pop up window for a single model environment entry.
Tty frame	The ShellTool.
Symbol table frame	The symbol table subwindow.
Reference and help frame	The reference file subwindow and help subwindow.

Table 7.2 Subframe of Magest frame.

Each frame may contain more than one subwindow. The main frame contains the main canvas and the main panel (see Figure 3.1.3). The model frame contains the model text subwindow, the model message text subwindow and the model panel (see Figure 3.2.2, 3.2.4, 3.2.6). The gmf frame contains the gmf panel (see Figure 3.2.1). The gmf panel is the same size as the gmf frame. The tty frame contains the tty subwindow (see Figure 3.5.5). The tty subwindow is the same size as the tty frame.

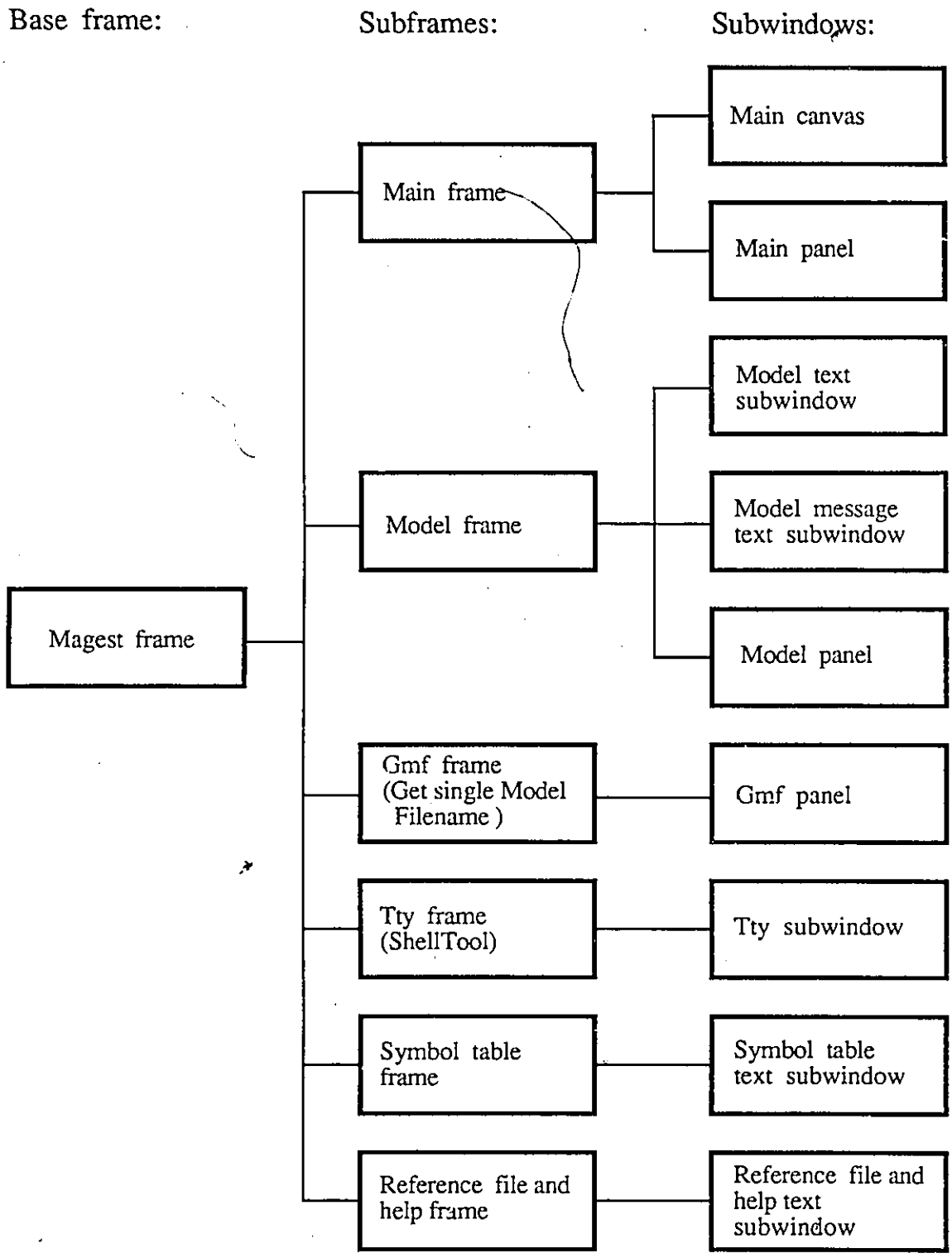

Figure 7.3.1 Main window structure in Magest

The symbol table frame contains the symbol table text subwindow (see Figure 3.5.8). The symbol table text subwindow is the same size as the symbol table frame. The reference file and help frame contain the reference file and help text subwindows (see Figure 3.3.3, 3.5.7). They are also the same size as their frames.

Due to the limitation of the window facilities in the Sun workstation, as explained in section 3.3.3, the reference file subwindow is shared with the help subwindow. Therefore, when this subwindow appears, the help subwindow disappears. Similarly when the help subwindow appears, the reference file subwindow disappears.

The management of these windows include operations to open, destroy, resize, redisplay, position and manage the content. The above window management is done in files called *magest.c* and *window_gest.c*.

As shown in Figure 7.3.2 there are three pop up windows, i.e., the confirm panel, the print panel and the get file panel which are supported by corresponding frames, i.e., the confirm frame, the print frame and the get file frame, respectively. These frames are also had their own base frames and are supported by SunView.

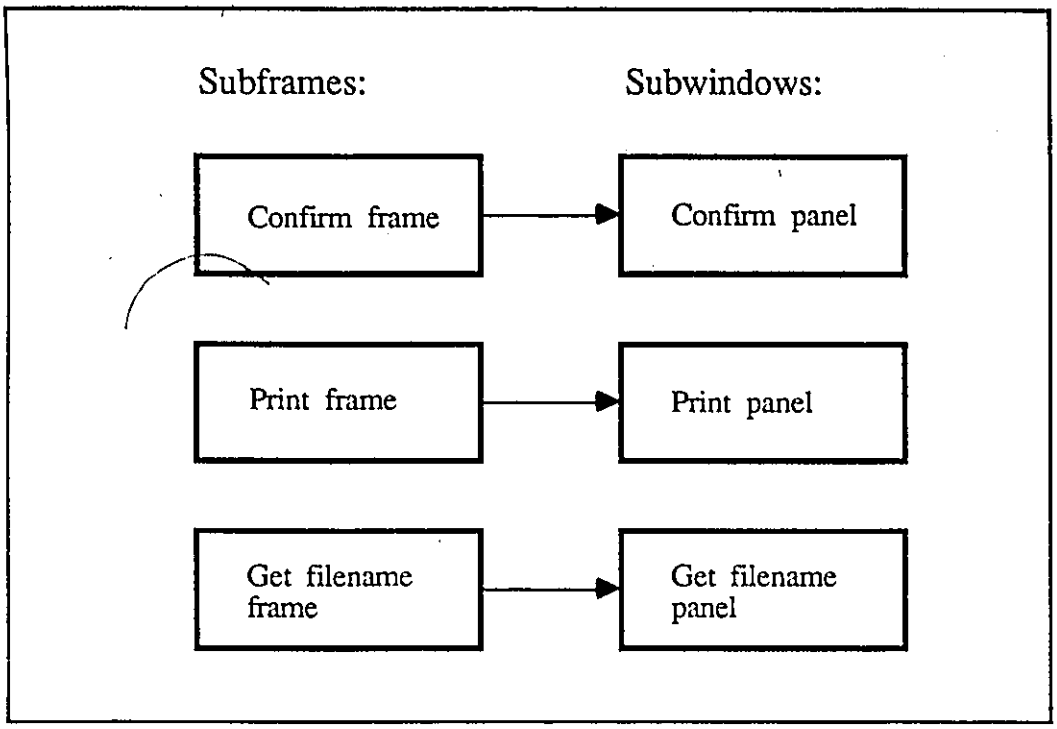

Figure 7.3.2 Structure of pop up windows.

The window management of these windows is done in a file called *confirm_gest.c*.

7.4 Limitations of Magest

Table 7.3 describes the values of the parameters used in Magest. They are assigned in the file called *define.c*. Most of the parameter values are arbitrary and can easily be modified in the file *define.c*. Once this is done, all one has to do is to recompile *define.c* and link edit it with the other object files which are *magest.c pargest1.c, pargest2.c, semgest.c, confirm_gest.c and window_gest.c.*

Furthermore, the diffential equations can not be expressed in matrix-vector form. They have to be specified by an assignment statement where the left-hand side consists of the highest derivative of a state variable.

Constant name	Limitation	Explanation
Max_Real_Pos_Exponent *	307	Maximum positive exponent power in real number.
Max_Real_Neg_Exponent *	319	Maximum negative exponent power in real number.
Max_RangeOf_Insert	2000 characters	Maximum length of RangeOf insert lines.
Max_Buf_Length	231072 bytes	Maximum length of a Gest model.
Max_Token_List	70000	Maximum tokens for a Gest model.
Max_Identifier	7000	Maximum number of user specified identifiers for a coupled model. (including all the identifiers of all the component models)
Max_Id_Length	36 characters	Maximum length of an user specified identifier.
Max_Temp_Storage	400	Maximum number of user specified identifiers for a model.
Max_Symbol_Table	40	Maximum number of component models (plus one) in a coupled model.
Max_Filename_Length	70 characters	Maximum length of a filename.
Max_Path_Length	100 characters	Maximum length of a directory.
Max_Line_Length	100 characters	Maximum length of a line in a text editor window.

Table 7.3 Constant limit of Magest.

* This limitation of the C language is checked within Magest at specification time.

Chapter 8

CONCLUSIONS AND FURTHER RESEARCH

In this thesis, an expert system for simulation modelling has been designed, implemented and discussed. It provides a knowledge-based modelling environment where methodological and user-provided knowledge are used by the system. The methodological knowledge includes the knowledge of the modelling methodology of the Gest language. User-given knowledge accepted by the system is also added incrementally to the usable knowledge of the modelling environment. This user-given knowledge is dynamically updated if the Magest system detects an inconsistency.

The Magest system has basically 3 modes of operation:

1) **The guidance mode** - In this mode, Magest guides the user in specifying a model. In this mode, Magest provides dynamic templates to the user to be filled-in. The templates are also dynamically tailored according to the user-given knowledge. In this mode, Magest performs syntactic and semantic checks before it accepts knowledge from the user. When an error is found, message appears in the message window indicating the error and possible corrections. The cursor of the text editor window is placed by Magest to the appropriate location to indicate a probable error.

2) **The certification mode** - The Magest system performs

complete syntactic and semantic checks. Furthermore semantic facts and rules are used to assure built-in quality assurance of the model specification. Each component model is certified separately. First the static structure is checked and certified then the dynamic structure is checked for certification. Once all the component models are certified, the Magest system can certify the coupling; i.e., input/output interface of component models between themselves and between the environment.

3) **The editor mode** - In this mode user can freely edit a Gest model without having any check performed. However, afterwards the model should be submitted for certification for possible detection of errors by Magest.

A number of system functions in Magest are presented. They include: the walking menu functions, the reference file function, the reserved word function, the symbol table function, the shelltool function, the magest i/o functions, the insert type of model functions and the prepare coupling functions. The system functions provide a full support to the user in the modelling environment.

The architecture of the system is conceived as a finite state machine in chapter six. The advantage of this representation is that it provides a top-down representation of the system architecture. It also provides a flexible way to represent and modularize the architecture. Moreover, a robust system is realized by this method.

The design and implementation method of the Magest system is

presented in chapter seven. It describes how the transformation of the knowledge base into a data structures, window structures and program codes.

I recommend a number of areas for further study:

1) **Graphic coupling** - A graphical editor for coupling is suggested for the future. The modeller can use the mouse to indicate the external and internal coupling on a graphical editor. The system can provide coupling specification from the graphical input. Furthermore, nested coupling ability can also be provided.

2) The system has to be completed by adding features to specify experimentations, parameters, and execution of simulation studies. The implementation of a translator is necessary to complete the system.

3) **Semantic facts and rules** - More semantic facts and rules can be added to the knowledge-base of the system models, experimentations, parameters as well as execution of simulation studies.

4) **Statistics** on error messages given to the user can be collected. - From the analysis of the statistics, we can find out how these could be avoided by improving the system, the Gest definition, and semantic facts and rules.

5) **List of identifier** for coupled and multi-rate models - The idea is to generate for coupled models and for multi-rate models, a combined list of identifiers used in all the component models.

6) After finishing the implementation of the Gest environment, a next step is to include **domain specific knowledge** in the knowledge-base of such a system. In this case, one may have specific knowledge such as ecological knowledge, or engineering domain-specific knowledge embedded in the knowledge base of the system.

Appendix A

FILES OF THE GEST ENVIRONMENT

There are different classes of files related to the design and implementation of this thesis: the object file of Gest, the implementation files, the documentation files, template files and help files.

A.1 Object File of the Gest Environment

File Name	Explanation
gest	Executable file of Gest Environment and Magest.

Table A.1 Object file of the Gest Environment.
(Contains the compiled versions of the files listed in A.2.)

A.2 Implementation Files

File Name	Explanation
define.c	System global variables declaration.
magest.c	Contains the main program. It administers the whole Gest Environment and Magest.
pargest1.c	First part of parser.
pargest2.c	Second part of parser.
semgest.c	The modules in this file perform different functions of data base management, lexical analysis and semantic checks.
confirm_gest.c	The modules in this file are responsible for three pop-up windows for confirmation use.
window_gest.c	The file contains most of the window management modules and miscellaneous functions, including "message" file.

Table A.2 Implementation files of the Magest system.
(Details of procedure are in Appendix B.)

A.3 Documentation Tool

File Name	Explanation
proall	An exec file to use the **prodoc** for all the procedure files given in table A.2.
prodoc	An executable file, to print out names and associated documentations of all procedures and functions given in a file.
prodoc.c	The source program of prodoc. It is written in C.

Table A.3 Documentation tools of magest system.

A.4 Template Files

Under directory **template**

File Name	Explanation
continuous.temp	Template for the Continuous model
continuous_multi.temp	Template for the Continuous multi-model
discrete.temp	Template for the Discrete model
discrete_multi.temp	Template for the Discrete multi-model
memoryless.temp	Template for the Memoryless model
memoryless_multi.temp	Template for the Memoryless multi-model
macro.temp	Template for the Macro model
coupled.temp	Template for the Coupled model
multirate.temp	Template for the Multirate model

Table A.4 Template files of the Magest system.

A.5 Help Files

Under directory **man**

File Name	Explanation
background.man	Background help manual
continuous.man	Continuous model help manual
continuous_multi.man	Continuous multi-model help manual
discrete.man	Discrete model help manual
discrete_multi.man	Discrete multi-model help manual
memoryless.man	Memoryless model help manual
memoryless_multi.man	Memoryless multi-model help manual
macro.man	Macro model help manual
coupled.man	Coupled model help manual
multirate.man	Multirate model help manual
parameter.man	Parameter specification help manual
experiment.man	Experiment specification help manual
run.man	Run help manual
output_module.man	Output module help manual
program.man	Program help manual

Table A.5 Help files of the Magest system.
(Only part of continuous.man is provided in this current implementation to provide an example of such a document.)

Appendix B

PROCEDURES AND FUNCTIONS IN MAGEST

The procedures and functions listed in this appendix are stored in six files. They are:

Name	Number of modules	Number of lines	Number of Kbytes	Description
magest.c	37	2878	84	Contains a main program. The modules in this file administer the whole Gest Environment and Magest.
pargest1.c pargest2.c	96	5210	134	The combination of these two files is the parser of Magest.
semgest.c	27	2364	72	The modules in this file perform different functions of data base management, lexical analysis and semantic checks.
confirm_gest.c	10	551	16	The modules in this file are responsible for three pop-up windows for confirmation use.
window_gest.c	41	2366	85	The file contains most of the window management modules and miscellaneous functions, including "messages".
Total	**211**	**13369**	**390**	

Table B.1 Magest source files and details.

The following lists of procedures and functions is copied from the original program listing. They are generated by a software tool called *proall*, which reads the original program listings and prints out the names and relevant comments of all procedures and functions.

Filename: magest.c

```
( 1) w_pr_init_main_frame
                /* Window initialization procedure for main_frame          */

( 2) w_pr_write_header
                /* To write header on canvas                               */

( 3) n_pr_s_model_main
                /* Activated by "Single Model" button in main_panel        */

( 4) n_pr_c_model_main
                /* Activated by "Coupled Model" button in main_panel       */

( 5) n_pr_m_model_main
                /* Activated by "Multi-Rate Model" button in main_panel    */

( 6) n_pr_parameter_main
                /* Activated by "Parameter" button in main_panel           */
                /*    - will be implemented as the 2nd phase of Magest -   */

( 7) n_pr_experiment_main
                /* Activated by "Experiment" button in main_panel          */
                /*    - will be implemented as the 2nd phase of Magest -   */

( 8) n_pr_run_main
                /* Activated by "Run Control" button in main_panel         */
                /*    - will be implemented as the 2nd phase of Magest -   */

( 9) n_pr_output_main
                /* Activated by "Output Module" button in main_panel       */
                /*    - will be implemented as the 2nd phase of Magest -   */

(10) n_pr_program_main
                /* Activated by "Gest Program" button in main_panel        */
                /*    - will be implemented as the 2nd phase of Magest -   */

(11) n_pr_translate_main
                /* Activated by "Translate" button in main_panel           */
                /*    - will be implemented as the 2nd phase of Magest -   */

(12) n_pr_simulation_main
                /* Activated by "Simulation" button in main_panel          */
                /*    - will be implemented as the 2nd phase of Magest -   */

(13) n_pr_quit_main
                /* Activated by "Quit" button in main_panel                */

(14) w_pr_init_sm
                /* Window initialization procedure for Single Model Mode   */

(15) n_pr_save_m
                /* Activated by "Save" button in m_panel                   */
```

```
(16) n_pr_print_m
                /* Activated by "Print" button in m_panel                    */
(17) n_pr_clear_m
                /* Activated by "Clear" button in m_panel                    */

(18) n_pr_quit_m
                /* Activated by "Return" button in m_panel                   */

(19) n_pr_load_m
                /* Activated by "Load" button in m_panel                     */

(20) pr_cert_error
                /* Procedure to check whether the beginning and ending token */
                /* were found in the file. If not, it is not possible to     */
                /* certify the model, and give error message.                */
                /* Input : certify_structure, token_number_b, token_number_e,*/
                /* Output: TRUE/FALSE,                                       */

(21) n_pr_ss_cert_sm
                /* Activated by "Static Structure" button in m_panel         */

(22) n_pr_ds_cert_sm
                /* Activated by "Dynamic Structure" button in m_panel        */

(23) n_pr_wm_cert_sm
                /* Activated by "Whole Model" button in m_panel in single    */
                /* model mode                                                */

(24) n_pr_osu
                /* Activated by Options menu                                 */

(25) w_pr_init_gmf_frame
                /* Window initialization procedure for gmf_frame             */
                /* ( Get model filename )                                 */

(26) n_pr_ok_gmf
                /* Activated by "OK" button in gmf_panel                     */

(27) n_pr_cancel_gmf
                /* Activated by "CANCEL" button in gmf_panel                 */

(28) n_pr_sm_textsw
                /* Notification procedure for accepting input text from      */
                /* single model text window                                  */

(29) pr_ret_ss_sm
                /* procedure to process information when user hit return     */
                /* key in single model text window                           */

(30) w_pr_init_cm
                /* Window initialization procedure for Coupled Model Mode    */

(31) n_pr_keyword_m
                /* Notification procedure for inserting keywords into text.  */
```

```
(32) n_pr_cm_textsw
                /* Notification procedure for accepting input text from      */
                /* Coupled Model text window                                 */

(33) pr_ret_ss_cm
                /* procedure to process informateion when user hit return    */
                /* key in coupled model text window                          */

(34) w_pr_init_mm
                /* Window initialization procedure for Multi-Rate Model      */
                /* Mode                                                      */

(35) pr_autosave
                /* Procedure for automatic saving file                       */

(36) d_pr_init_all_variables
                /* procedure for initialization of all variables             */

(37) d_pr_init_sys_variables
                /* procedure for initialization of system variables          */
```

Filename: pargest1.c

```
( 1) d_pr_init_parser_variables
                /* Data procedure for initializating variables each          */
                /* time before entering parser                               */

( 2) p_pr_get_token
                /* Parser procedure for getting token from token list.       */

( 3) p_pr_eol_before
                /* Parser procedure to search whether a EndOfLine char is    */
                /* between the last token and the present token              */
                /* Input : token_index, - the present token index            */
                /* Output: eol_position, found_eol_before                    */

( 4) p_pr_first_unmatch
                /* Parser procedure 1 for reinitializing token and next token*/
                /* by the given start_token                                  */

( 5) p_pr_nonterm_unmatch
                /* Parser procedure 2 for putting error information into     */
                /* variables.                                                */

( 6) p_pr_nonterm_sem_unmatch
                /* Parser procedure 2 for putting error information into     */
                /* variables.                                                */

( 7) p_pr_rela_oper
                /* Parser procedure of RelationalOperator                    */

( 8) p_pr_inte
                /* Parser procedure of integer                               */

( 9) p_pr_real
                /* Parser procedure of Real                                  */

(10) p_pr_mode_iden
                /* Parser procedure of ModelIdentifier                       */

(11) p_pr_macr_iden
                /* Parser procedure of MacroIdentifier                       */

(12) p_pr_func_name
                /* Parser procedure of FunctionName                          */

(13) p_pr_iden
                /* Parser procedure of Identification                        */

(14) p_pr_vari
                /* Parser procedure of Variable                              */

(15) p_pr_scal_vari
                /* Parser procedure of ScalarVariable                        */
```

```
(16) p_pr_cons
                /* Parser procedure of Constant                                  */

(17) p_pr_subs_vari
                /* Parser procedure of SubscriptedVariable                       */

(18) p_pr_deri_vari
                /* Parser procedure of DerivativeVariable                        */

(19) p_pr_dime_vari
                /* Parser procedure of DimensionedVariable                       */

(20) p_pr_rand_vari
                /* Parser procedure of RandomVariable                            */

(21) p_pr_dist_name
                /* Parser procedure of DistributionName                          */

(22) p_pr_list_scal_vari
                /* Parser procedure of ListOfScalarVariable                      */

(23) p_pr_list_scal_dime_vari
                /* Parser procedure of ListOfScalarOrDimensionedVariable         */

(24) p_pr_list_scal_subs_vari
                /* Parser procedure of ListOfScalarOrSubscriptedVariable         */

(25) p_pr_list_arit_expr
                /* Parser procedure of ListOfArithmeticExpression                */

(26) p_pr_arit_expr
                /* Parser procedure of ArithmeticExpression                      */

(27) p_pr_term
                /* Parser procedure of Term                                      */

(28) p_pr_fact
                /* Parser procedure of Factor                                    */

(29) p_pr_func_refe
                /* Parser procedure of FunctionReference                         */

(30) p_pr_libr_func_name
                /* Parser procedure of LibraryFunctionName                       */

(31) p_pr_bool_expr
                /* Parser procedure of BooleanExpression                         */

(32) p_pr_bool_term
                /* Parser procedure of BooleanTerm                               */

(33) p_pr_bool_fact
                /* Parser procedure of BooleanFactor                             */

(34) p_pr_bool_cons
                /* Parser procedure of BooleanConstant                           */
```

```
(35) p_pr_bool_vari
                /* Parser procedure of BooleanVariable                    */

(36) p_pr_time_expr
                /* Parser procedure of TimeExpression                     */

(37) p_pr_upda_cond
                /* Parser procedure of UpdateCondition                    */

(38) p_pr_stat
                /* Parser procedure of Statement                          */

(39) p_pr_assi_stat
                /* Parser procedure of AssignmentStatement                */

(40) p_pr_if_stat
                /* Parser procedure of IfStatement                        */

(41) p_pr_case_stat
                /* Parser procedure of CaseStatement                      */

(42) p_pr_for_stat
                /* Parser procedure of ForStatement                       */

(43) p_pr_for_clau
                /* Parser procedure of ForClause                          */

(44) p_pr_whil_stat
                /* Parser procedure of WhileStatement                     */

(45) p_pr_macr_refe
                /* Parser procedure of MacroReference                     */

(46) p_pr_read_stat
                /* Parser procedure of ReadStatement                      */

(47) p_pr_file_name
                /* Parser procedure of FileName                           */

(48) p_pr_cont_mode
                /* Parser procedure of ContinuousModel                    */

(49) p_pr_stat_stru_cont_mode
                /* Parser procedure of StaticStructureOfContinuousModel   */

(50) p_pr_dyna_stru_cont_mode
                /* Parser procedure of DynamicStructureOfContinuousModel  */

(51) p_pr_outp_func
                /* Parser procedure of OutputFunction                     */
                /* A sub-BNF definition from p_pr_dyna_stru_cont_mode,    */
                /* p_pr_dyna_stru_disc_mode, p_pr_dyna_stru_memo_mode     */
                /* and p_pr_dyna_stru_macr_mode                           */
```

```
(52) p_pr_upda
                /* Parser procedure of Update                                  */

(53) p_pr_disc_mode
                /* Parser procedure of DiscreteModel                           */

(54) p_pr_stat_stru_disc_mode
                /* Parser procedure of StaticStructureOfDiscreteModel          */

(55) p_pr_dyna_stru_disc_mode
                /* Parser procedure of DynamicStructureOfDiscreteModel         */
```

Filename: pargest2.c

```
( 1) p_pr_memo_mode
                /* Parser procedure of MoemorylessModel                    */

( 2) p_pr_stat_stru_memo_mode
                /* Parser procedure of StaticStructureOfMemorylessModel    */

( 3) p_pr_dyna_stru_memo_mode
                /* Parser procedure of DynamicStructureOfMemorylessModel   */

( 4) p_pr_cont_mult_mode
                /* Parser procedure of ContinuousMultiModel                */

( 5) p_pr_disc_mult_mode
                /* Parser procedure of DiscreteMultiModel                  */

( 6) p_pr_memo_mult_mode
                /* Parser procedure of MemorylessMultiModel                */

( 7) p_pr_mode_sele
                /* Parser procedure of ModelSelection                      */

( 8) p_pr_macr_mode
                /* Parser procedure of MacroModel                          */

( 9) p_pr_stat_macr_mode
                /* Parser procedure of StaticStructureOfMacroModel         */

(10) p_pr_dyna_stru_macr_mode
                /* Parser procedure of DynamicStructureOfMacroModel        */

(11) p_pr_coup_multirate_mode
                /* Parser procedure of Coupled and Multi-rate Model        */

(12) p_pr_cert_all_comp_mode
                /* Parser procedure to certify all component models in     */
                /* coupled model.                                          */

(13) p_pr_exte_vari
                /* Parser procedure of ExternalVariables                   */

(14) p_pr_comp_mode
                /* Parser procedure of ComponentModel                      */

(15) p_pr_list_comp_mode
                /* Parser procedure of ListOfComponentModel                */

(16) p_pr_macr_mode_coup_mode
                /* Parser procedure of MacroModelInCoupledModel            */

(17) p_pr_type_comp_mode
                /* Parser procedure of TypeOfComponentModel                */
```

```
(18) p_pr_exte_coup
                /* Parser procedure of ExternalCoupling                    */

(19) p_pr_inte_coup
                /* Parser procedure of InternalCoupling                    */

(20) p_pr_mode_iden_coup_mode
                /* Parser procedure of  ModelIdentifierOfCoupledModel      */

(21) p_pr_inpu_vari
                /* Parser procedure of InputVariable                       */

(22) p_pr_outp_vari
                /* Parser procedure of OutputVariable                      */

(23) fn_get_symbol_id
                /* Function to return the symbol table number of a given   */
                /* model name                                              */

(24) p_pr_type
                /* Parser procedure of Type                                */

(25) p_pr_inpu_decl
                /* Parser procedure of InputDeclaration                    */

(26) p_pr_stat_decl
                /* Parser procedure of StateDeclaration                    */

(27) p_pr_outp_decl
                /* Parser procedure of OutputDeclaration                   */

(28) p_pr_decl_othe_mode_desc
                /* Parser procedure of DeclarationOfOtherModelDescriptors  */

(29) p_pr_auxi_vari_decl
                /* Parser procedure of AuxiliaryVariableDeclaration        */

(30) p_pr_cons_decl
                /* Parser procedure of ConstantDeclaration                 */

(31) p_pr_cons_assi
                /* Parser procedure of ConstantAssignment                  */

(32) p_pr_para_decl
                /* Parser procedure of ParameterDeclaration                */

(33) p_pr_auxi_para_decl
                /* Parser procedure of AuxiliaryParameterDeclaration       */

(34) p_pr_auxi_para_comp
                /* Parser procedure of AuxiliaryParameterComputation       */

(35) p_pr_tabu_func_decl
                /* Parser procedure of TabularFunctionDeclaration          */
```

```
(36) p_pr_inte_vari_decl
                /* Parser procedure of InterpolatedVariableDeclaration      */

(37) p_pr_macr_used
                /* Parser procedure of  MacrosUsed                          */

(38) p_pr_list_macr_iden
                /* Parser procedure of ListOfMacroIdentifier                */

(39) p_pr_rang_decl
                /* Parser procedure of RangeDeclaration                     */

(40) p_pr_rang_limi
                /* Parser procedure of RangeLimit                           */

(41) p_pr_parser
                /* Parser procedure of parser                               */
                /*                                                          */
                /* Input : magest_mode, model_mode                          */
                /* Output: code                                             */
```

Filename: semgest.c

```
( 1) d_pr_init_id_table
               /* Data structure procedure for initialization of all       */
               /* symbol tables in Magest                                  */

( 2) d_pr_init_temp_table
               /* Data structure procedure for initialization of temporary */
               /* symbol tables in Magest                                  */

( 3) pr_insert2_message
               /* Procedure of inserting error message 1                   */

( 4) pr_insert_message
               /* Procedure of inserting error messages 2                  */

( 5) p_pr_in_sem_details
               /* Parser procedure of initialization semantic checks       */
               /* variables                                                */

( 6) d_fn_insert_macro
               /* Data procedure used in Coupled & Multi-Rage Model for    */
               /* inserting macro model name into give temp-table. Therefore*/
               /* it would be sorted in alp. order and repeated macro model */
               /* name wouldn't be inserted again.                         */

( 7) d_fn_pr_table_insert
               /*-----------------------------------------------------------*/
               /* Data flowing function and procedure to insert identifier  */
               /* into symbol table                                         */
               /*                                                           */
               /* Input : temp_table_index, token_index, str_insert,        */
               /*         type_insert, Max_Temp_Storage,                    */
               /*         token_list[].column_b, token_list[].column_e      */
               /*         in_buffer, temp_table[temp_table_index]           */
               /*                                                           */
               /* Output: temp_table[temp_table_index], if inserted         */
               /*         code = 0 , if insert failed                       */
               /*                1 , if insert succeeded                    */
               /*                                                           */
               /* Searching technique : Binary search                       */
               /*-----------------------------------------------------------*/

( 8) d_fn_table_search
               /* Data flowing function and procedure to search whether     */
               /* an identifier has been existed in a given symbol table    */
               /*                                                           */
               /* Input : table, index_b, index_e, string, length           */
               /* Output: index, str                -- if found --          */
               /*                                                           */
               /* Searching technique : Binary search                       */
```

```
( 9) d_fn_table_str_search
                /* Data flowing function and procedure to search whether     */
                /* an structure has been existed in a given symbol table     */
                /*                                                           */
                /* Input : table, index_b, index_e, str                      */
                /* Output: index                          -- if found --     */
                /*                                                           */

(10) d_pr_table_transfer
                /* Data flowing procedure to transfer data from              */
                /* temp_table[temp_table_index] to id_table[id_table_index]  */
                /*                                                           */
                /* Input  : temp_table, temp_table_index, id_table_index     */
                /* Output : id_table, element, code                          */

(11) l_fn_pr_get_char
                /* Lexical function and procedure to return a character from */
                /* in_buffer                                                 */

(12) l_fn_is_alp_num
                /* Lexical function to check whether the given character is  */
                /* alphanumeric or not.                                      */

(13) l_fn_is_cap_alp
                /* Lexical function to check whether the given character is  */
                /* alphabetical.                                             */

(14) l_fn_is_text
                /* Lexical function to check whether the given character is  */
                /* correct input text                                        */

(15) l_pr_number
                /* Lexical procedure to tokenize number                      */
                /*                                                           */
                /* Input  : in_buffer[], p_word_c, c                         */
                /* Output : int_value                                        */
                /*          real_value                                       */
                /*          code =   0 , error has found                     */
                /*                 454 , unsigned integer                    */
                /*                 453 , unsign real                         */

(16) l_pr_lexical
                /*  Procedure to give the token number of a given word.      */
                /*                                                           */
                /* Input :  in_buffer, buffer_length                         */
                /* Output : token_number, token_list[token_length].column_b, */
                /*          code                                             */
                /* Global auxiliary variable: c                              */
```

```
(17) l_pr_is_it_double_sign
                /* Lexical procedure to check if the token number is          */
                /* a double sign, then it might be one of the following sign */
                /*   1) Not equal sign                                       */
                /*   2) Smaller than and equal sign                          */
                /*   3) Greater than and equal sign                          */
                /*   4) Equal sign                                           */
                /*   5) Period sign                                          */
                /*   6) Multiple sign                                        */
                /*   7) Power sign                                           */
                /*   8) ExternalCouplingSign                                 */
                /*   9) InternalCouplingSign                                 */
                /*  10) Format code: I_ A_ L_ E_ F_ X_ B_                    */
                /* This procedure is called from l_pr_tokenize() only.       */

(18) l_pr_tokenize
                /* Lexical procedure for tokenize a string                   */
                /*                                                           */
                /* Input : magest_mode, in_buffer, buffer_length,            */
                /*         token_number_b, token_number_e                    */
                /* Output: token_list, token_length, error_pos, code,        */
                /*         token_b, token_e                                  */
                /* Global auxiliary variable: p_word_b, p_word_c,            */
                /*        token_number, c                                    */

(19) p_fn_sem_stat_stru
                /* Parser procedure of performing semantic check for when    */
                /* Static Structure is finished                              */

(20) p_fn_sem_dyna_stru
                /* Parser procedure of performing semantic check for when    */
                /* Dynamic Structure is finished                             */

(21) p_fn_sem_syn
                /* Parser function to check whether the variable should be   */
                /* an subscripted variable                                   */

(22) p_fn_sem_lhs
                /* Parser function to check Left Hand Side of semantics      */

(23) p_fn_sem_rhs
                /* Parser function to check right hand side of semantics     */

(24) p_fn_sem_cond
                /* Parser function to check condition statement of semantics */

(25) p_fn_sem_details
                /* Parser function to check semantic details                 */

(26) p_fn_sem_coupled_io
                /* Parser function to check semantic details for input and   */
                /* output variable of coupled model                          */

(27) p_pr_sem_coupled_ma
                /* Parser procedure to check the existence of Macro Model    */
                /* in coupled model                                          */
```

Filename: window_gest.c

```
( 1) w_pr_init_tty_frame
                /* Window initialization procedure for tty_frame            */
                /* ( ShellTool subwindow )                                  */

( 2) n_pr_tty
                /* Activated by "ShellTool" button                          */

( 3) w_pr_init_sym_frame
                /* Window initialization procedure for sym_frame            */

( 4) n_pr_sym
                /* Activated by "Symbol Table" button                       */

( 5) w_pr_init_res_frame
                /* Window initialization procedure for res_frame            */

( 6) n_pr_done_res
                /* Activated by Done in frame menu of res_frame             */

( 7) n_pr_res
                /* Activated by "Reserved Words" button                     */

( 8) w_pr_init_ref_frame
                /* Window initialization procedure for ref_frame            */

( 9) n_pr_done_ref
                /* Activated by Done in frame menu of ref_frame             */

(10) n_pr_print_ref
                /* Activated by Print in frame menu of ref_frame            */

(11) n_pr_ref
                /* Activated by "Reference File" button                     */

(12) w_pr_put_ref_file
                /* Put an reference on the reference window                 */
                /* Input : ref.directory, ref.filename                      */
                /* Output: on reference frame, ref_label                    */

(13) n_pr_help
                /* Activated by "Help" button MAGEST                        */
                /*              -- not complete yet --                      */

(14) w_pr_display_help
                /* Window procedure for displaying help file on ref_frame   */

(15) fn_stat_file
                /* Procedure to check is the given existed.                 */
                /* Input : filename                                         */
                /* Output: 0, file is not exist.                            */
```

```
(16) w_pr_out_message
                /* Window procedure for outputting message on message window */

(17) w_pr_clear_text
                /* Window procedure for clearing a given text subwindow      */
                /* Input : textsw, directory, filename                       */
                /* Output: a integer, 0 - Cancel operation                   */

(18) w_pr_cursor_position
                /* Window procedure for putting the cursor position on       */
                /* text window                                               */

(19) w_pr_cursor_eol
                /* Window procedure to move cursor to end of line of a text  */
                /* in sunwindow.                                             */

(20) pr_count_blank
                /* Procedure to count copy the leading blank from the given  */
                /* line to the blank_line.  If there are no leading blank in */
                /* the given line, the blank line will contain just a single */
                /* '\n'.                                                     */
                /* Input:  given_line                                        */
                /* Output: blank_line,                                       */

(21) w_pr_next_position
                /* Window function for searching the next cursor position    */
                /* and insert it into text-subwindow                         */

(22) w_pr_init_rangeof_insert
                /* Window procedure for initialization of Range Of and       */
                /* assignment statement insertion string                     */

(23) w_pr_rangeof_concat
                /* Window procedure for concatating RangeOf line to          */
                /* rangeof.line                                              */

(24) pr_fit_string
                /* Procedure to fit the given string to the given length by  */
                /* adding more blanks or truncate it.                        */

(25) d_pr_get_structure_type
                /* Data procedure to get the character string for a given    */
                /* structure type                                            */

(26) pr_add_filetype
                /* Procedure to add filetype to the filename                 */

(27) w_fn_subwindow_menu
                /* Window function for removing frame menu in text window    */

(28) w_fn_frame_menu
                /* Window function for removing frame menu in frame          */

(29) w_fn_text_menu
                /* Window function for removing text menu in text window     */
```

```
(30) n_pr_insert_str_m
                /* Activated by the walking menu in text subwindow of        */
                /* single model mode.                                        */

(31) n_pr_insert_sel_rep
                /* Activated by the walking menu in text subwindow of        */
                /* single model mode.                                        */

(32) n_pr_insert_line
                /* Activated by "Insert Line" menu in text subwindow         */

(33) fn_position_of_m
                /* Function to give the order position of a given value.     */

(34) w_pr_get_line
                /* Window procedure to retrieve the content of a line in     */
                /* textsw, and return the begin and end position of line.    */

(35) w_pr_update_panel
                /* Window procedure to update the certification status       */

(36) n_pr_ext_coupling
                /* Activated by "ExternalCoupling" button in m_panel         */

(37) fn_get_all_macro_name
                /* Function for coupled and multi-rate model, to read and    */
                /* insert all macro model name into temp-table 1             */

(38) fn_check_cp_inputs_exist
                /* Function for coupled model to check whether any external  */
                /* input variable in coupled model exists                    */

(39) n_pr_int_coupling
                /* Activated by "InternalCoupling" button in m_panel         */

(40) fn_check_co_input_exists
                /* Function for coupled model to check whether any unused    */
                /* input variable in component model exists                  */

(41) d_pr_init_messages
                /* Data structure procedure for initialization of error      */
                /* messages                                                  */
```

Filename: confirm_gest.c

```
( 1) w_pr_confirm_gest
                /* Window procedure for procedure a box with three buttons - */
                /* Yes, No and Cancel.                                       */
                /* Input : message1, message2                                */
                /* Output: answer, 0 - Cancel button is being pressed        */

( 2) w_pr_init_con_frame
                /* Window initialization procedure for con_frame             */

( 3) n_pr_yes_no_cancel_con
                /* Activated by "Yes", "No" or "Cancel" button in con_panel  */

( 4) w_pr_print_confirm
                /* Window procedure for print filename, directory, options   */
                /* with three buttons - Ok and Cancel.                       */
                /* Input : directory, filename                               */
                /* Output: to printer

( 5) w_pr_init_pr_frame
                /* Window initialization procedure for pr_frame              */

( 6) n_pr_ok_cancel_pr
                /* Activated by "OK" or "CANCEL" button in pr_panel          */

( 7) w_pr_file_confirm
                /* Window procedure for requesting filename, directory, with */
                /* three buttons - Ok and Cancel.                            */
                /* Input : message1, message2                                */
                /* Output: answer, 0 - Cancel button is being pressed        */

( 8) w_pr_init_file_frame
                /* Window initialization procedure for file_frame            */

( 9) n_pr_ok_cancel_file
                /* Activated by "OK" or "CANCEL" button in file_panel        */

(10) w_pr_center_frame
                /* Window procedure to center a given frame on the screen    */
```

Appendix C

MESSAGES IN THE MAGEST SYSTEM

A list of two hundred and fourty six messages used in the Magest system are given in this appendix. Messages in the Magest system are initialized in a subroutine at the beginning of program execution. The messages are centralized in the subroutine called *d_pr_init_messages* which exists in file *window_gest.c*

All messages are stored in a string array called *message*. The system currently can have a maximum of 300 messages, each of length 200 characters. To modify the size of the array *message*, it is sufficient to change the constant values of *Max_Message_Length* and *Max_Messages* in file *define.c*. The index used in the array *message* is the internal number used to identify the messages and is also given in the following list. In their internal forms, messages are stored in numerical order.

Magest has three types of messages:

1) Messages for pop up windows,

2) System error messages, and

3) Syntax and semantic error messages.

C.1 Messages for pop up windows

27 Do you want to save your model?

140 Please enter model name:

139 Model :

138 Filename :

141 CoupledModel :

28 Enter filename please:

142 Multi-RateModel :

271 Do you want to delete your old model?

23 Magest: New file.

C.2 System error messages

0 Magest: Can't create base frame window.
Suggest: Close some of your windows.

4 Magest: Can't create error message text window.
Suggest: Close some of your windows.

1 Magest: Can't create help message window.
Suggest: Close some of your windows.

3 Magest: Can't create message panel window.
Suggest: Close some of your windows.

6 Magest: Can't create panel button.
Suggest: Close some of your windows.

5 Magest: Can't create selection panel window.
Suggest: Close some of your windows.

2 Magest: Can't create text editor window.
Suggest: Close some of your windows.

7 Magest: Can't create TTY window.
Suggest: Close some of your windows.

20 Magest: Program size too large, symbol table overflow.
Suggest: Reduce your program size or switch to editing mode

16 Magest: Sorry, help documentation is not yet implemented.
Suggest: Consult with your system support stuff.

17 Magest: System error in Magest, refer to lexgest.c of declaration of terminal non alphabetical order error.
Suggest: Consult with your system support stuff.

8 Magest: System out of memory.
Suggest: Consult with your system support stuff.

240 Magest: File '' does not found.

244 Magest: File '' is empty.

69 Magest: File cannot be certified since 'ContinuousModel' is missing.

72 Magest: File cannot be certified since 'ContinuousMultiModel' is missing.

130 Magest: File cannot be certified since 'CoupledModel' is missing.

70 Magest: File cannot be certified since 'DiscreteModel' is missing.

73 Magest: File cannot be certified since 'DiscreteMultiModel' is missing.

131 Magest: File cannot be certified since 'DynamicStructure' is missing.

132 Magest: File cannot be certified since 'DynamicStructuresOfSubmodels' is missing.

133 Magest: File cannot be certified since 'EndStaticStructure' is missing.

75 Magest: File cannot be certified since 'MacroModels' is missing.

71 Magest: File cannot be certified since 'MemorylessModel' is missing.

74 Magest: File cannot be certified since 'MemorylessMultiModel' is missing.

277 Magest: File cannot be certified since 'MultiRateModel' is missing.

24 Magest: File does not exist, please check your directory from ShellTool.

25 Magest: File does not exist, please save your file before you want to print.

76 Magest: Give a name to the model.

C.3 Syntax and Semantic error messages

Note: In the following messages, appropriate information taken from the user's program are displayed within ''.

155 Magest: '' is not allowed to use here.

19 Magest: '*)' expected for ending comments, add '*)' at the end of your comments.

180 Magest: '[' expected.

186 Magest: 'OutputFunctions' block expected.

232 Magest: 'Submodel' expected..

229 Magest: 'Submodels' expected.

18 Magest: Alphabetic is following numeric, check your program.

192 Magest: An auxiliary variable cannot appear on the right-hand side of the assignment statement where it is specified.

31 Magest: Arithmetic expression expected.

40 Magest: Arithmetic factor expected.

39 Magest: Arithmetic term expected.

210 Magest: Array declaration expected.

123 Magest: Assignment statement expected.

106 Magest: Assignment statement or For block expected.

108 Magest: Auxiliary Parameter assignment statement expected in Auxiliary Parameter block.

201 Magest: AuxiliaryParameter '' has not been specified in AuxiliaryParameters block.
Suggest: Eliminate '' or give it a specification.

173 Magest: AuxiliaryVariable '' is not specified in Dynamic Structure.
Suggest: Eliminate '' or specifys it in Dynamic Structure.

181 Magest: AuxiliaryVariable has been specified twice.

43 Magest: Boolean expression expected.

45 Magest: Boolean factor expected.

44 Magest: Boolean term expected.

242 Magest: Can't open file ''.

14 Magest: Character inside quotation is not text character, use alphabetic, numeric, sign, tab or bell

246 Magest: Component model '' has not been certified.
Suggest: Please certified '' in single model mode.

252 Magest: Component model type expected.

105 Magest: Constant assignment statement expected in Constant block.

190 Magest: Constant cannot specified for more than once.

198 Magest: Constants '' has not been specified in Constants block.
Suggest: Eliminate '' or give it a specification.

177 Magest: Derivative of state is expected.

127 Magest: Distribution name expected.

194 Magest: Eliminate multiple specification of AuxiliaryVariable.

195 Magest: Eliminate multiple specification of interpolated variable.

193 Magest: Eliminate multiple specification of random variable.

185 Magest: Eliminate Output Function block.

10 Magest: Exponent overflow, range of exponent is between -319

and 307.

11 Magest: Exponent underflow, range of exponent is between -319 and 307.

147 Magest: External output for coupled model has been connected.

276 Magest: FastContinuousModel or SlowContinuousModel can not used in Coupled Model.

280 Magest: FastContinuousModel expected.

278 Magest: FastContinuousModel expected, only allow to have one FastContinuousModel and one SlowContinuousModel.

41 Magest: Identifier in Auxiliary Variable and Output should have the same type.

22 Magest: Identifier in State and Output should have the same type.

179 Magest: Illegal Derivative variable.

146 Magest: Input for component model has been connected.

164 Magest: Input '' is not used on the right-hand side of Derivative.
Suggest: Eliminate '' or use it in Derivative.

161 Magest: Input '' is not used on the right-hand side of Output Function.
Suggest: Eliminate '' or use it in Output Function.

167 Magest: Input '' is not used on the right-hand side of State Transition.
Suggest: Eliminate '' or use it in State Transition.

187 Magest: Input variable expected.

13 Magest: Integer value overflow, range of integer is between -2147483648 and 2147483647

204 Magest: Interpolated variable '' has not been specified in Interpolation block.
Suggest: Eliminate '' or give it a specification.

158 Magest: Interpolated variable '' is not used in Dynamic Structure.
Suggest: Eliminate '' or use it at least one in the Dynamic Structure.

112 Magest: Interpolation assignment statement expected in Interpolation block.

149 Magest: Macro ' ' is not used in Dynamic Structure.
Suggest: Eliminate '' or use it at least once in the Dynamic Structure.

207 Magest: Macro '' is not specified Macro Reference in Dynamic Structure.
Suggest: Eliminate '' or specify in Dynamic Structure.

148 Magest: Macro model has not used in component model.

267 Magest: Model name expected.

272 Magest: No more internal coupling will exist.

211 Magest: Non-array declaration expected.

196 Magest: Non-array type of variable expected.

115 Magest: Number or scalar variable expected.

9 Magest: Numerical value overflow.

265 Magest: Only component model can use in InternalCouplings.

273 Magest: Only component model can use on the right hand side of ExternalCouplings.

257 Magest: Only coupled model can use on the left hand side of ExternalCouplings.

275 Magest: Only FastContinuousModel or SlowContinuousModel can use in Multi-Rate Model.

269 Magest: Only input variable can use here.

274 Magest: Only output variable can use here.

152 Magest: Output '' is not specified in Output Function.
Suggest: Eliminate '' or specify it in Output Function.

270 Magest: Output variable expected.

182 Magest: Parameter expected on the right-hand side of an auxiliary parameter specification.

26 Magest: Please clear your present text editor window before loading another file.

237 Magest: Put 'ComponentModel' here.

59 Magest: Put 'Do' here.

79 Magest: Put 'DynamicStructure' here.

107 Magest: Put 'EndAuxiliaryParameters' at the end of 'AuxiliaryParameters' block.

54 Magest: Put 'EndCase' at the end of 'CaseOf' block.

253 Magest: Put 'EndComponentModel' at the end of 'ComponentModel' block.

104 Magest: Put 'EndConstants' at the end of 'Constants' block.

85 Magest: Put 'EndDerivatives' at the end of 'Derivative' block.

80 Magest: Put 'EndDynamicStructure' here.

231 Magest: Put 'EndDynamicStructuresOfSubmodels' at the end of 'DynamicStructuresOfSubmodels' block.

264 Magest: Put 'EndExternalCouplings' at the end of 'ExternalCouplings' block.

261 Magest: Put 'EndExternalInputs' at the end of 'ExternalInputs' block.

263 Magest: Put 'EndExternalOutputs' at the end of 'ExternalOutputs' block.

251 Magest: Put 'EndExternalVariables' at the end of 'ExternalVariables' block.

55 Magest: Put 'EndFor' at the end of 'For' block.

50 Magest: Put 'EndIf' at the end of 'If' block.

266 Magest: Put 'EndInternalCouplings' at the end of 'InternalCouplings' block.

111 Magest: Put 'EndInterpolations' at the end of 'Interpolations' block.

256 Magest: Put 'EndMacroModels' at the end of 'MacroModels' block.

81 Magest: Put 'EndModel' here.

91 Magest: Put 'EndModelSelections' at the end of 'ModelSelections' block.

87 Magest: Put 'EndOutputFunctions' at the end of 'OutputFunctions' block.

128 Magest: Put 'EndRandom' at the end of 'EndRandom' block.

89 Magest: Put 'EndStateTransitions' at the end of 'StateTransitions' block.

78 Magest: Put 'EndStatucStructure' here.

93 Magest: Put 'EndSubmodel' at the end of 'Submodel' block.

94 Magest: Put 'EndUpdates' at the end of 'Update' block.

60 Magest: Put 'EndWhile' at the end of 'While' block.

238 Magest: Put 'ExternalCouplings' here.

262 Magest: Put 'ExternalOutputs' here.

236 Magest: Put 'ExternalVariables' here.

63 Magest: Put 'In:' inside the macro reference.

239 Magest: Put 'InternalCouplings' here.

65 Magest: Put 'Out:' inside the macro reference.

224 Magest: Put 'Select' in here for Model Selections.

77 Magest: Put 'StaticStructure' here.

49 Magest: Put 'Then' here.

135 Magest: Put 'To' here.

98 Magest: Put 'Tolerance' here.

99 Magest: Put 'When' in here for Update condition.

227 Magest: Put 'With' here.

258 Magest: Put a '.' here.

260 Magest: Put a '<--' here.

62 Magest: Put a '(' here.

32 Magest: Put a ')' here or reorganize your program.

66 Magest: Put a ')' here.

125 Magest: Put a ';' here.

259 Magest: Put a '-->' here.

47 Magest: Put a ',' here.

57 Magest: Put a '=' here.

29 Magest: Put a '[' here or reorganize your program.

137 Magest: Put a ']' here.

33 Magest: Put a "'" here for derivative of a state variable.

121 Magest: Put a dimensioned variable here. eg. AR(1..15).

67 Magest: Put a filename beside 'ReadFrom'.

42 Magest: Put a library function name here, or change the expression.

61 Magest: Put a macro identifier after 'Macros'.

114 Magest: Put a macro variable here, or delete the ','

129 Magest: Put a number ere, or delete the ','.

136 Magest: Put a number or constant variable here.

120 Magest: Put a number or identifier here.

96 Magest: Put a positive or negative real number here.

119 Magest: Put a range limit here, or delete the ','.

102 Magest: Put a scalar or dimensioned variable here.

52 Magest: Put a scalar variable after 'CaseOf'.

68 Magest: Put a scalar variable here.

37 Magest: Put a scalar variable here, or delete the ',' .

36 Magest: Put a scalar variable in Random block.

116 Magest: Put a variable after 'RangeOf'.

255 Magest: Put all the names of macro model which has been used in component modes here.
Suggest: Press prepare 'External Coupling' button.

38 Magest: Put an arithmetic expression here, or delete the ',' .

100 Magest: Put an assignment statement here, or delete the ','.

56 Magest: Put an identifier after 'For'.

134 Magest: Put an integer here for index.

35 Magest: Put an integer here for index, or delete the '['.

12 Magest: Put an integer in exponent part.

34 Magest: Put an integer in here as dimension.

225 Magest: Put an unsigned integer here for 'Select'.

228 Magest: Put an unsigned integer here for total number of submodels.

233 Magest: Put an unsigned integer here for submodel number.

53 Magest: Put an unsigned integer or identifier as 'Case' identification.

30 Magest: Put an unsigned integer or identifier here, or delete the ','.

124 Magest: Put at least one 'Case' in 'CaseOf' block.

226 Magest: Put at least one assignment statement in Model Selections block.

101 Magest: Put at least one assignment statement in Update block.

254 Magest: Put at least one component model here.

64 Magest: Put at least one scalar variable here.

110 Magest: Put at least one tabular function name here.

46 Magest: Put one of relational operators here. (=,Not=,<,<=,>,>=)

118 Magest: Put one or two range limit here. eg. >5, <=8.

214 Magest: Random '' is not specified in Random block.
Suggest: Eliminate '' or give it a specification.

126 Magest: Random assignment statement(s) expected in Random block.

184 Magest: RangeOf specification has been defined more than one.

21 Magest: Repeated identifier in a same model.

122 Magest: Scalar or dimensioned variable expected.

281 Magest: SlowContinuousModel expected.

279 Magest: SlowContinuousModel expected, only allow to have one FastContinuousModel and one SlowContinuousModel.

212 Magest: Specify Output Function for '' or eliminate it from Outputs list.

170 Magest: State '' is not specified the derivative in Derivative block.
Suggest: Eliminate '' or specify the derivative.

157 Magest: State has been specified in Derivative Section.

178 Magest: State has been specified in State Transition Section.

109 Magest: Subscripted-variable expected.

143 Magest: Tabular Function ' ' is not used for any interpolated variable.
Suggest: Eliminate ' ' or use it for an interpolated variable.

183 Magest: Tabular Function expected on the right-hand side of Interpolation specification.

113 Magest: Tabular function name expected.

249 Magest: Too much component models, maximum 39 component can be use.

189 Magest: Undefine Constant.

268 Magest: Undefine model name.

15 Magest: Undefined character, use alphabetic, numeric, sign, tab or bell

188 Magest: Undefined Identifier.

191 Magest: Undefined Model name.

51 Magest: Variable is expected at the beginning of an assignment statement.

197 Magest: Variable should be declared as boolean.

234 Magest: You must declare MacroModel '' in MacroModels block.

58 Magest: You must either have 'To' or 'DownTo' here.

97 Magest: You must either have a real number or scalar variable here.

117 Magest: You must have '=' for RangeOf declaration.

84 Magest: You must have 'Derivatives' block in the dynamic structure of continuous model.

230 Magest: You must have 'DynamicStructuresOfSubmodels' block in the dynamic structure of multi-models.

90 Magest: You must have 'ModelSelections' block in the 'Submodel' block.

223 Magest: You must have 'OutputFunctions' block in the dynamic structure of macro model.

86 Magest: You must have 'OutputFunctions' block in the dynamic structure of memoryless model.

88 Magest: You must have 'StateTransitions' block in the dynamic structure of discrete model.

92 Magest: You must have 'Submodel' block in the dynamic structure of multi-models.

48 Magest: You must have a '=' for assignment statement.

95 Magest: You must have a scalar variable between 'When' and 'Crosses'.

221 Magest: You must have Input declaration in a macro model.

219 Magest: You must have Input declaration in a memoryless model.

103 Magest: You must have Interpolation declaration in Tabular Function block.

83 Magest: You must have Output declaration in a continuous model.

250 Magest: You must have Output declaration in a coupled model.

218 Magest: You must have Output declaration in a discrete model.

222 Magest: You must have Output declaration in a macro model.

220 Magest: You must have Output declaration in a memoryless model.

82 Magest: You must have State declaration in a continuous model.

217 Magest: You must have State declaration in a discrete model.

Appendix D

SYSTEM TEMPLATES

Several templates are available within the Magest modelling environment. The template names and the names of the files where they are stored are as follows:

Template Name	File Name
Continuous model	continuous.temp
Continuousmulti-model	continuous_multi.temp
Discrete model	discrete.temp
Discrete multi-model	discrete_multi.temp
Memoryless model	memoryless.temp
Memoryless multi-model	memoryless_multi.temp
Macro model	macro.temp
Coupled model	coupled.temp
Multi-rate model	multirate.temp

Table D.1 The name and filename of templates

The templates are dynamic, i.e., depending on the need, they can be expanded or retracted under the control of the Magest system.

D.1 Template for continuous Models

```
ContinuousModel
  StaticStructure
    Inputs
    States
    Outputs
    AuxiliaryVariables

    Constants
    EndConstants
    Parameters
    AuxiliaryParameters
    EndAuxiliaryParameters

    TabularFunctions
      Interpolations
      EndInterpolations
    Macros
  EndStaticStructure

  DynamicStructure
    Derivatives

    EndDerivatives

    OutputFunctions

    EndOutputFunctions

    Updates
      When
    EndUpdates
  EndDynamicStructure EndModel
```

D.2 Template for continuous multi-models

```
ContinuousMultiModel
  With  Submodels

 StaticStructure
  Inputs
  States
  Outputs
  AuxiliaryVariables

  Constants
  EndConstants
  Parameters
  AuxiliaryParameters
  EndAuxiliaryParameters

  TabularFunctions
   Interpolations
   EndInterpolations
  Macros
EndStaticStructure

DynamicStructuresOfSubmodels
  Submodel
   DynamicStructure
    Derivatives

    EndDerivatives

    OutputFunctions

    EndOutputFunctions

    Updates
     When

    EndUpdates

    ModelSelections
     Select
      When

    EndModelSelections
   EndDynamicStructure
  EndSubmodel
EndDynamicStructuresOfSubmodels EndModel
```

D.3 Template for discrete models

```
DiscreteModel
  StaticStructure
    Inputs
    States
    Outputs
    AuxiliaryVariables

    Constants
    EndConstants
    Parameters
    AuxiliaryParameters
    EndAuxiliaryParameters

    TabularFunctions
      Interpolations
      EndInterpolations
    Macros
  EndStaticStructure

  DynamicStructure
    StateTransitions

    EndStateTransitions

    OutputFunctions

    EndOutputFunctions

    Updates
      When
    EndUpdates
  EndDynamicStructure EndModel
```

D.4 Template for discrete multi-models

```
DiscreteMultiModel
  With  Submodels

 StaticStructure
  Inputs
  Outputs
  AuxiliaryVariables

  Constants
  EndConstants
  Parameters
  AuxiliaryParameters
  EndAuxiliaryParameters

  TabularFunctions
    Interpolations
    EndInterpolations
  Macros
 EndStaticStructure

 DynamicStructuresOfSubmodels
  Submodel
    DynamicStructure
      StateTransitions

      EndStateTransitions

      OutputFunctions

      EndOutputFunctions

      Updates
        When

      EndUpdates

      ModelSelections
        Select
          When

      EndModelSelections
    EndDynamicStructure
  EndSubmodel
 EndDynamicStructuresOfSubmodels EndModel
```

D.5 Template for memoryless models

```
MemorylessModel
  StaticStructure
    Inputs
    Outputs
    AuxiliaryVariables

    Constants
    EndConstants
    Parameters
    AuxiliaryParameters
    EndAuxiliaryParameters

    TabularFunctions
      Interpolation
      EndInterpolations
    Macros
  EndStaticStructure

  DynamicStructure
    OutputFunctions

    EndOutputFunctions

    Updates
      When
    EndUpdates
  EndDynamicStructure EndModel
```

D.6 Template for memoryless multi-models

```
MemorylessMultiModel
  With  Submodels

StaticStructure
  Inputs
  Outputs
  AuxiliaryVariables

  Constants
  EndConstants
  Parameters
  AuxiliaryParameters
  EndAuxiliaryParameters

  TabularFunctions
    Interpolations
    EndInterpolations
  Macros
EndStaticStructure

DynamicStructuresOfSubmodels
  Submodel
    DynamicStructure
      OutputFunctions

      EndOutputFunctions

      Updates
        When

      EndUpdates

      ModelSelections
        Select
          When

      EndModelSelections
    EndDynamicStructure
  EndSubmodel
EndDynamicStructuresOfSubmodels EndModel
```

D.7 Template for macro models

```
MacroModel
  StaticStructure
    Inputs
    Outputs
    AuxiliaryVariables

    Constants
    EndConstants
    Parameters
    AuxiliaryParameters
    EndAuxiliaryParameters

    TabularFunctions
      Interpolations
      EndInterpolations
    Macros
  EndStaticStructure

  DynamicStructure
    OutputFunctions

    EndOutputFunctions

    Updates
      When
    EndUpdates
  EndDynamicStructure EndModel
```

D.8 Template for coupled models

```
CoupledModel
  ExternalVariables
    Inputs
    Outputs
  EndExternalVariables

  ComponentModels


  EndComponentModels EndModel
```

D.9 Template for multi-rate models

```
MultiRateModel
  ExternalVariables
    Inputs
    Outputs
  EndExternalVariables

  ComponentModels
    FastContinuousModel
    SlowContinuousModel
  EndComponentModels EndModel
```

References

Aytac, Z.K., Ören, T.I. (1986). **Magest: A Model-Based Advisor and Certifier for Gest Programs.** In: Modelling and Simulation Methodology in the Artificial Intelligence Era, M.S. Elzas, T.I. Ören, B.P. Zeigler (Eds.) North Holland, pp. 299-307.

Balci, O. (Ed.) (1987). **Methodology and Validation.** Proceedings of the Conference on Methodology and Validation, 6-9 April, 1987, Orlando, FL, Simulation Series 19:1 (Jan.), SCS, San Diego, CA.

Banks, J., Gerstein, D., Searles, S.P. (1987). **Modeling Processes, Validation, and Verification, of Complex Simulations: A Survey.** In: Methodology and Validation, O. Balci (Ed.), Simulation Series 19:1 (Jan.), SCS, San Diego, CA., pp. 13-18.

Barrett, M.I. (1988). **Numerical Integration Routines for the Gest Simulation Environment.** Master's Thesis, Computer Science Department, University of Ottawa, Ottawa, Ontario, Canada.

Jackson, P. (1985). **Introduction to Expert Systems.** Addison-Wesley Publishing Company, Inc., U.S.A.

Ören, T. I. (1971). **GEST: A Combined Digital Simulation Language for Large Scale Systems.** In: Proceedings of the 1971 AICA (Association Internationale pour le Calcul Analgique) symposium on simulation of complex systems, Tokyo, Japan, Sept. 3-7, 1971, pp. B-11-14.

Ören, T. I. (1981). **Concepts and Criteria to Assess Acceptability of Simulation Models: A Framework.** CACM, 24:4, 180-189.

Ören, T. I. (1984). **Gest: A Modelling and Simulation Language Based on System Theoretic Concepts.** In: Simulation and Model Based Methodologies: An Integrative View, T.I. Ören, B.P. Zeigler, M.S. Elzas (Eds.), Springer Verlag, Heidelberg, West Germany, pp. 281-335.

Ören, T. I. (1986a). **Knowledge Bases for an Advanced Simulation Environment.** In: Intelligent Simulation Environments, P.A. Luker,

H.H. Adelsberger (Eds.) SCS, San Diego, CA, U.S.A., pp. 16-22.

Ören, T. I. (1986b). **Software Requirements of Knowledge Bases of an Advanced Simulation Environment.** In: Proceedings of JSST International Conference on Recent Advances in Simulation on Comples Systems, Tokyo, Japan, July 15-17, 1986, pp. 283-288.

Ören, T. I. (1987a). **Quality Assurance Paradigms For Artificial Intelligence In Modelling and Simulation.** Simulation, 48:4, 149-151.

Ören, T. I. (1987b). **Model Update: A Model Specification Formalism With A Generalized View of Discontinuity.** Summer Computer Simulation Conference, Montreal, Canada, July 1987, pp. 689-694.

Ören, T.I., Aytac, K.Z. (1985). **Architecture of MAGEST: A Knowledge-based Modelling and Simulation System.** In: Simulation in Research and Development, A. Javor (Ed.), North-Holland, Amsterdam, pp. 99-109.

Ören, T.I., Elzas, M.S., Sheng, G. (1985). **Model Reliability and Software Quality Assurance in Simulation of Nuclear Fuel Waste Management System.** In: Waste Management '85, Vol.1, R.G. Post (Ed.), 381-396.
Reprinted In: ACM Simuletter, 16:4, 4-19.

Ören, T.I., Sheng, G. (1988). **Semantic Rules and Facts for an Expert Modelling and Simulation System.** In: Preprints of IMACS Congress, Paris, France, July 1988.

Ören, T.I., Tam, J.C.M. (1988). **An Expert Modelling and Simulation System on Sun Workstation.** In: Simulation Environments, Nice, France, June 1-3, 1988, SCSI, San Diego, California, U.S.A, pp. 255-260.

Palusinski, O.A. (1985). **Simulation of Dynamic Systems Using Multi-Rate Integration Techniques.** Transactions of the Society For Computer Simulation, 2:4, 257-273.

Sun Microsystems. (1986a). **Windows and Window Based Tool: Beginner's Guide.** Sun Microsystems, Inc., Mountain View,

California, U.S.A.

Sun Microsystems. (1986b). **Sun View Programmer's Guide.** Sun Microsystems, Inc., Mountain View, California, U.S.A.

Symonds, A.J. (1986). **Introduction to IBM's Knowledge-Systems Products.** IBM Systems Journal, 25:2, 134-146.

Tremblay, J.P., Sorenson, P.G. (1985). **The Theory and Practice of Compiler Writing,** Mc Graw-Hill, U.S.A.

Wymore, A. W. (1967). **A Mathematical Theory of Systems Engineering: The Elements.** John Wiley, New York, 353.

Ye, Y.C., Ören, T.I. (1988, In Preparation). **A Translator for Gest language. Technical Report TR-88-xx.** Computer Science Department, University of Ottawa, Ottawa, Ontario, Canada.