GEST - A MODELLING AND SIMULATION LANGUAGE
BASED ON SYSTEM THEORETIC CONCEPTS*

Tuncer I. Ören
Computer Science Department
University of Ottawa
Ottawa, Ontario, K1N 9B4
Canada

ABSTRACT

GEST is the first model and simulation specification language. Specifications of the model and the experiment are totally separated. The modelling world view is based on the axiomatic system theory of Wymore which provides an excellent basis for simulation modelling and symbolic model processing. This chapter has two aims: 1) To present the GEST language and the robust and rich modelling paradigm it provides even for non-simulation application areas, as well as 2) to foster design and development of other GEST-like modelling and simulation languages which would provide other modelling formalisms within comprehensive modelling and simulation systems.

* This study has been sponsored by the Natural Sciences and Engineering Research Council of Canada (NSERC) by the Operating Grant A8117.
1. INTRODUCTION

GEST is a modelling and simulation language based on general system theoretic concepts. It was conceived in 1969 and first document about it was published in 1971 (Ören 1971). Its definition has been updated a few times. The last version of it is GEST 81 (Ören 1981, 1982a).

GEST, even at its first version, departed radically from other simulation languages. It is the first model-based simulation language. In GEST, specifications of the model and the experiment are totally separated. The modelling world view of GEST is based on the axiomatic system theory of Wymore (1967, 1976).


This chapter has two aims: 1) To present the GEST language, and 2) to foster design and development of other GEST-like modelling and simulation languages which would provide other modelling formalisms within comprehensive modelling and simulation systems. The rationale for developing such advanced tools within comprehensive modelling and simulation software systems is given in chapter 1 of this volume (Ören 1984).

A list of references where GEST is treated directly, or where GEST has been referred to is given at the end of this chapter.

2. WORLD VIEW OF GEST 81

GEST is a model and simulation specification language. Therefore a GEST program is highly descriptive and acts as a documentation (for communication among humans) as well as a specification (for man-machine communication). This documentation ability of GEST will become apparent in the sequel. If looked only superficially, some elements of GEST may appear to be cumbersome to specify, such as "END COUPLING FOR model-identifier." However, GEST has to be conceived differently.
First of all GEST, due to its world view is a good basis for a comprehensive modelling and simulation software system. Within such a system, the computer-assisted modelling module would have all the necessary information to generate "END COUPLING FOR model-identifier" as soon as the specification of all the input-output relationships have been completed by the user, based on system-initiated prompts.

Conceived within a computer-assisted modelling system, some GEST instructions are totally or partially generatable by the system for the convenience of the user. Thus, in addition to be user-friendly and convenient, computer assistance in specifying GEST models can assure completeness as well as consistency checks of the specifications. Another model-based language where user input is minimized through computer- assistance is SEMA (Oren and Collie 1980).

In GEST, a program consists of three distinct parts, i.e.,
1) Mathematical model,
2) Experiment(s), and
3) Output module(s).

The "model" consists of a parametric model and associated set(s) of parameter values. The "experiment" is the specification of experimental conditions (or experimental frames) which have to be applied to a model. The "output module" is the specification of the output program to be used to display the result of the simulation study.

Expressed in Backus-Naur Form (BNF) the definition of a GEST program is as follows: (The meta-language used to describe GEST, is given in the appendix.)

```
program =
    "PROGRAM" identifier
    model
    experiment
    output-module-specification
    "END PROGRAM" identifier ";" .
```
3. MODEL

3.1 Background

A specific GEST model, is a pair of parametric model and model parameter set. As seen in Figure 1, consists of two parts: 1) a parametric model, and 2) model parameter set(s). The optional model parameter set may appear more than once. A BNF specification of "model" follows:

\[
\text{model} = \\
\text{parametric-model-specification} \\
[\{ \text{model-parameter-set} \}].
\]

**Parametric Model:** A parametric model associated with a parameter set constitutes a specific model that one can use in a simulation study.

A modeller, during the formulation of a parametric model, needs only to specify the names of the parameters of a model. At this stage, the actual values of the parameters need not be specified.

A parametric model may consist of one or several component model(s). A component model may be continuous, discrete, or memoryless. A coupled model consists of a set of component model(s) and their input/output interface which is also called the coupling specification.

**Component Model:** A component model consists of two sections: In the first section the static structure (or the descriptive structure) of the model, is expressed. In the second section the dynamic structure (or the predictive structure) of the model is specified. This latter section consists of the state transition and the optional output function(s).

The static structure of each component model is described basically in terms of model descriptive variables such as state-, input-, and output-variables. However, autonomous models, by definition, do not need inputs to operate. And in some cases explicit output variables may
Specific GEST model

Parametric model

Component models

Component model #1

...

Component model #n

End component models

Coupling (i.e., input/output interface) of all component model(s)

End parametric model

Model parameter set(s)

Parameter set #1

...

Parameter set #p

End parameter set(s)

End specific GEST model

Figure 1. Parts of a specific GEST model
not exist. In this last case, some or all of the state-variables and/or auxiliary variables may be considered to be the output-variables of the model. In memoryless component models, state variables do not exist. In this case, current output is computed based on the values of the current inputs.

Furthermore, the static structure of a model requires other declarations, such as type and range of values of the descriptive variables of the model. Several modelling formalisms can be used to express component models, such as ordinary differential equations with or without discontinuities in their state-variables and/or their derivatives, difference equations, or combined continuous and discrete-change models.

3.2 Continuous model

Basics

As is shown in Figure 2, the specification of a continuous component model consists of two parts: 1) The static structure and 2) The dynamic structure of the model.

The specification of the static structure of a continuous model consists basically of the declaration of the descriptive variables of the model under the following categories:

- input variable
- state variable
- output variable
- auxiliary variable
- constant
- parameter
- auxiliary parameter
- tabular function declaration, and
- interpolated variable declaration
Figure 2. Layout of a continuous component model specification in GEST
The type of every descriptive variable can be specified separately. The default type is accepted to be real. The ranges of the values of the descriptive variables can also be specified as part of a model in order to enforce some automatic consistency checks. Both external input variables (those variables which are not provided by some component models of a system) and parameters can be stochastic. In this case, it is possible to declare the distribution function to be used to generate them. Another possibility is the ability of declaring tabular functions and the associated interpolation requirements.

The dynamic structure consists of two blocks, i.e. 1) The derivative block which contains the specifications of the derivatives of the state variables and the computations of the necessary auxiliary variables. 2) The output block contains the transformations of the state and/or auxiliary variables into output variables.

Examples:

Some elementary examples of continuous models expressed in GEST are given in Figures 3 and 4.

CONTINUOUS MODEL MIXED_LOGISTIC_GROWTH

STATIC STRUCTURE
STATES Y1, Y2;
OUTPUTS Y1, Y2;
PARAMETERS R1, R2, A1, A2, B1, B2;
END STATIC STRUCTURE;

DYNAMIC STRUCTURE
DERIVATIVES
Y1' = R1*Y1*(1.0 - A1*Y1 - B1*Y2);
Y2' = R2*Y2*(1.0 - A2*Y2 - B2*Y2);
END DERIVATIVES;
END DYNAMIC STRUCTURE;
END MODEL MIXED_LOGISTIC_GROWTH;

Figure 3. A continuous model expressed in GEST
Some relevant definitions in BNF follow:
continuous-model =

"CONTINUOUS MODEL" model-identifier

"STATIC STRUCTURE"

[ {input-declaration} ]

[ {state-declaration} ]

[ {output-declaration} ]

[ {auxiliary-variable-declaration} ]

[ {constant-declaration} ]

[ {parameter-declaration} ]

[ {auxiliary-parameter-declaration} ]

[ {tabular-function-declaration} ]

"END STATIC STRUCTURE" ; ;

"DYNAMIC STRUCTURE"

"DERIVATIVES"

{statement}

"END DERIVATIVES" ; ;

[ "OUTPUT FUNCTION"

{statement}

"END OUTPUT FUNCTION" ; ; ]

"END DYNAMIC STRUCTURE" ; ;

"END MODEL" model-identifier ; ; .

The upper and lower limits of the variables are experimental parameters. In this case they are certified in experiments.

Examples:

RANGE OF ST
RANGE OF D
RANGE OF P
RANGE OF VE
Definitions of type, random-variable-declaration, and range, expressed in BNF follow:

type = "INTEGER" | "REAL" | "BOOLEAN" | "LITERAL" .

random-variable-declaration =
   "RANDOM" ["REAL"|"INTEGER"] list-of-scalar-variable ";"
   [ variable "=" distribution-name "(" list-of-parameter ")" ";" ]
   "END RANDOM" ";" .

range-declaration =
   ( "RANGE OF" (scalar-variable | array-variable) "=" type
   "(" range-limit ")" ";" )
   | ( "RANGE OF" scalar-variable "=" "LITERAL"
   "(" list-of-identification ")" ";" )
   | ( "RANGE OF" array-variable "="
   "ARRAY" ("REAL" | "INTEGER" ) array-variable ";" ) .

range-limit =
   ([boundary] ".." boundary) (boundary ".." [boundary]) .

boundary = real | integer | scalar-variable .

The upper and lower boundaries of a range can be given in a model. In this case they are considered absolute and cannot be modified in an experimental frame. However, if the boundary values may depend on experimental conditions they may be declared as parameters to be specified in experimental frames. In the last case, the range of their acceptable values should be given as the range of the associated parameters.

Examples:

RANGE OF STATUS = LITERAL(ON,OFF);
RANGE OF D = REAL( .. 32.5 );

RANGE OF PRIORITY = INTEGER ( 1 .. MAX_PRIORITY );
RANGE OF VELOCITY = REAL ( 0.0 .. 327.4 );
Definitions of input, state, and output variables follow:

**Input declaration**:

\[
\text{input-declaration = ( "INPUT"["S"] [type] list-of-variable ");" ) | ( "INPUT"["S"] random-variable-declaration ");" ) .}
\]

**Examples:**

```
INPUTS A,B,C;
RANGE OF A = REAL ( 0.00 .. );
RANGE OF B = REAL ( .. 32.80);
RANGE OF C = REAL ( 0.00 .. 100.00);
INPUTS ARRAY REAL K(1..15), L(1..10), M(1..15);
INPUT AR, P(1..20);
INPUT SWITCH;
RANGE OF SWITCH = LITERAL (ON, OFF);
INPUTS RANDOM REAL CUS_ARRIVAL, X;
CUS_ARRIVAL = EXPONENTIAL(LAMBDA);
X = NORMAL(XM, XS);
END RANDOM;
```

**State declaration**:

\[
\text{state-declaration = ( "STATE"["S"] [type] list-of-variable ");" ) .}
\]

**Examples:**

```
STATE POPULATION;
STATES YEAST, ALCOHOL;
STATE MASS_FLOW_DISCHARGE;
STATE REAL POPULATION (1..10);
```

**Output declaration**:

\[
\text{output-declaration = ( "OUTPUT"["S"] [type] list-of-variable ");" ) .}
\]

**Example:**

```
OUTPUTS REAL L (1..15), N(1..LIM_N);
```

Output variables do not depend on input variables. State variables or auxiliary variables which are used as output must be explicitly declared.
Definitions of auxiliary variables, constants, and parameters follow:

auxiliary-variable-declaration =
   "AUXILIARY VARIABLES" list-of-variables ;

constant-declaration =
   "CONSTANT" ["S"] [type] list-of-scalar-variable ";" .

Example:

    CONSTANTS REAL K, L ;

constant-assignment =
   scalar-variable "=" arithmetic-expression ";" .

Examples:

    PI       = 3.1415926;
    PI80     = PI/180.0;
    LATITUDE = 52.0;
    LONGITUDE = -5.0;

parameter-declaration =
    
    (*PARAMETER ["S"] [type] list-of-variable ";" )
    | (*PARAMETER ["S"] random-variable-declarations ).

Examples:

    PARAMETER REAL PA(1..3,1..5), PI;

    PARAMETER
       RANDOM REAL KE, KD;
       KE = NORMAL(200., 20.);
       KD = NORMAL(XM, SD);
    END RANDOM;
Auxiliary parameters are defined in terms of parameters and constants. Once the values of the parameters are given, the values of the auxiliary parameters can be computed by the system and made available for the convenience of the user.

**Examples:**

```
AUXILIARY PARAMETERS K, L;
  K = PI/OMEGA;
  L = 1.00/DIST;
END AUXILIARY PARAMETER;
```

```
interpolated-var: ...) | ( "INTERPOLATION"
  scalar:
    "(" list-of-function-name "")"
    *= "arithmetic-expression ";")
  "END INTERPO IATION"
```

Auxiliary parameters may depend on constants, parameters, and other auxiliary parameters.

**Examples:**

```
tabular-function-declaration =
  [*DISCONTINUOUS*]
  "TABULAR" ( "FUNCTION"["S"] | "FUNCTION_2")
  list-of-function-name ";".
```

DISCONTINUOUS and FUNCTION_2 introduce discontinuous and two-dimensional tabular functions, respectively.
Examples:

```
TABULAR FUNCTIONS GROWTH_RATE, FK;
TABULAR FUNCTION_2 DIFF_COEF;
DISCONTINUOUS TABULAR FUNCTION RAIN;
```

```plaintext
interpolated-variable-declaration =
  ( "INTERPOLATION"
    scalar-variable "=" function-name
    "(" list-of-scalar-variable ")" ";" )
| ( "INTERPOLATIONS"
  [ scalar-variable "=" function-name
    "(" list-of-scalar-variable ")" ";" ]
  "END INTERPOLATION" ";" ) .
```

Examples:

```
INTERPOLATION RATE = CURVE_1 (TEMPERATURE) ;

INTERPOLATIONS
  RATE1 = ABC (T);
  RK = FUN (T);
END INTERPOLATIONS;
```
3.3 Discrete and Memoryless Models

Basics

Both discrete and memoryless models, like continuous model, have two parts: 1) The static structure and 2) the dynamic structure.

Discrete models allow specification of systems expressed by a set of first order difference equations.

In discrete models, the static structure is like the static structure of a continuous model. However, a discrete model differs from a continuous model in the dynamic structure where the derivative block is replaced by the following block:

```
STATE TRANSITION
    statements
END STATE TRANSITION;
```

A memoryless model differs in two ways from a continuous model: 1) The static structure does not have declaration of any state variable, and 2) the dynamic structure has only output function specification.

A memoryless model transforms instantaneously its inputs and parameters into some output. For the convenience of naming, if a memoryless model has one output only, the same name can be used to designate the model and its output.
Example:

MEMORYLESS MODEL BIRTH\_RATE

STATIC STRUCTURE

INPUTS

\[ S, \quad (* \text{MATERIAL STANDARD OF LIVING} *) \]
\[ NE, \quad (* \text{EFFECTIVE POLLUTION} *) \]
\[ P; \quad (* \text{POPULATION} *) \]

OUTPUT BIRTH\_RATE;

PARAMETERS K20, K21, K22, K23, K24, K25;

END STATIC STRUCTURE;

DYNAMIC STRUCTURE

\[ B = K20 - K21*S - K22*NE - K23*P; \]
IF \( B \geq K24 \) AND \( B < K25 \)
THEN BIRTH\_RATE = B;
ELSE IF \( B < K24 \)
THEN BIRTH\_RATE = K24;
ELSE BIRTH\_RATE = K25;
END IF;
END IF;

END DYNAMIC STRUCTURE;

END MODEL BIRTH\_RATE;

Figure 5. A memoryless model expressed in GEST
3.4 Coupling

Basics

A coupled model (or a resultant model) consists of a set of component model(s) and their coupling which specifies input/output relationships of the component models.

An example of a coupled model is given in Figure 6. A GEST representation of the coupled model represented in Figure 6, is provided in Figure 7.

The layout of a coupled model specification in GEST is given in Figure 8.

![Figure 6. A coupled model](image)
COUPLED MODEL Z

EXTERNAL

INPUT IN;
   RANGE OF IN = REAL (0.0 .. 10.0);
OUTPUT OUT;
   RANGE OF OUT = REAL (40.0 .. 75.0);
END EXTERNAL;

COMPONENT MODELS M, N;

(* Detailed specifications of the component models M, N
   would appear herebelow: *)
MODEL M
   ...
END MODEL M;

MODEL N
   ...
END MODEL N;

END COMPONENT MODELS;

EQUIVALENCING

INPUTS Z.IN = M.B;
OUTPUTS Z.OUT = N.H;
END EQUIVALENCING;

COUPLING FOR Z

M.A <-- N.H;
M.B <-- Z.IN;  (* Z.IN IS AN EXTERNAL INPUT *)
M.C <-- M.E;  (* FEED-BACK COUPLING *)
N.F <-- M.D;
N.G <-- N.K;  (* FEED-BACK COUPLING *)
END COUPLING FOR Z;
END MODEL Z;

Figure 7. GEST representation of the coupled model given in Figure 6.
Figure 8. Layout of a coupled model specification in GEST
Some relevant BNF definitions follow:

coupled-model =
  "COUPLED MODEL" model-identifier
  [ external-variables ]

  "COMPONENT MODELS" list-of-component-models ";"
  { component-model }
  "END COMPONENT MODELS" ";"

  [ equivalencing-external-and-internal-variables ]

coupling
  "END MODEL" model-identifier ";"

external-variables =
  "EXTERNAL"
    [ input-declaration [[ range-declaration ]] ]
    [ output-declaration [[ range-declaration ]] ]
  "END EXTERNAL" ";".

In the external variables section, the names of the external input and output variables of the coupled model are declared. For every input or output variable, range of the values may also be specified.

An implication of the ability to declare external input or output variables is the possibility to specify nested couplings where at least one of the component models is itself a coupled model.

Another implication is the ability to declare the ranges of the external input and output variables independent of the ranges of the corresponding variables of component models. The compatibility of external and corresponding internal variables can be checked algorithmically.
list-of-component-models =
    model-identifier [ ":" unsigned-integer "TO" unsigned-integer ]
    [ ",” model-identifier
        [ ":" unsigned-integer "TO" unsigned-integer ]
    ] .

Example: COMPONENT MODELS M:1 TO 15, N, P:3 TO 7;

This statement causes the generation of 15 replicas of a model M. The
generated copies are then named, by the system, M:1, M:2, ..., M:15.
Similarly, five replicas of the component model P are generated and
named P:3, P:4, ..., and P:7.

equivalencing-external-and-internal-variables =
    "EQUIVALENCING"
    equivalencing-inputs
    equivalencing-outputs
    "END EQUIVALENCING" ; ; .

In equivalencing external and internal inputs, one has to consider
that an input to the coupled model can be the input to one or several
component models. However, every output of the coupled model is an
output of one component model only.

equivalencing-inputs =
    "INPUTS"
    { model-identifier "." input-variable "="
        model-identifier [ ":" unsigned-integer "." input-variable
        [ [ "," model-identifier [ ":" unsigned-integer ]
            [ ]; ]; ] .
    }

equivalencing-outputs =
    "OUTPUTS"
    { model-identifier "." output-variable "="
        model-identifier [ ":" unsigned-integer
            [ "." output-variable ";" ] .
    }

output-variable = scalar-variable | dimensioned-variable.
Coupled Model Formalism of GEST as a Top-Down Model Conception and Stepwise Model Refinement Tool

"Coupled Model" formalism provided in GEST, facilitates top-down model conception and stepwise model refinement. Figures 9 a-f show steps of model conception and corresponding GEST modeling statements for the example model given in Figure 7.

In step 1, one specifies the name of the model, the input and output variables of the model, and their ranges of acceptable values (Figure 9c, 9d).

In step 2, the names of the component models are specified (Figure 9b).

In step 3, each component model is specified separately. The model can either be specified from scratch or can be fetched from a model base (Figure 9c).

In step 4, the equivalencing of external and internal variables are specified.

In this step, an input to the coupled model (i.e., an external input to the resultant model) provides values to an input of one or several component models.
Then, one specifies, for every output of the coupled model (i.e., for every external output), the names of the output variable and of the component model which provides the values (Figure 9e).

In step 5, the coupling (i.e., the input/output relationships of the component models) is specified as follows (Figure 9f):

For every component model do
  For every input variable do
    Specify input-output relationship
  Loop
Loop

Some Implications

In a nested coupling at least one component model is a coupled model. Since the resultant model has its input(s) and output(s) declared it can act as a component model in a nested coupling. The concept of nested coupling introduced in GEST in 1970 (Ören 1970) allows both top-down model refinement and bottom-up model synthesis.

Several copies of similar models can be created automatically. For example, suppose that a model called, say "M" has already been specified. If the user wants to create n (to be specific let n=15) replicas of M, in the list of component models all one has to specify is M:1 TO 15. The created n replicas may be identical or similar models depending whether or not they have identical parameter values or not. Furthermore similar models thus created (i.e., replicas of the same generic model) can have same or different specific frames (see section 4.2).

Computer-assistance is straightforward in both specifying and checking the consistencies of coupled models. In a computer-assisted modelling system even some of the checks need not be done manually, due to the guidance of the modelling system. In such a system, for every component system, every input variable is listed (by the system) after the name of the component model. Therefore it is even not possible to misspell the names. Furthermore if an input (input of the resultant model) in Figure 8, the line:

M.B <-- Z.IN;

has to be generated, also the possible ranges of the value specified, for every input variable of the system can also be checked, set of the range of the value specified, for every input variable of the system can also be checked, set of the range of the value specified, for every input variable of the system can also be checked, set of the range of the value specified, for every input variable of the system can also be checked, set of the range of the value specified, for every input variable of the system can also be checked, set of the range of the value specified, for every input variable of the system can also be checked, set of the range of the value specified, for every input variable of the system can also be checked.
The coupled model of the output provides the val-

Relationships of the (Figure 9f):

A coupled model.

(s) declared it

The concept of (Figure 9f) allows both

automatically. For

ready been speci-

let n=15) reple-

s to specify is

or similar models

values or not.

as of the same

(see section

coupling specification can be very useful in the documentation of sys-

tems consisting of large number of interacting component systems. A

documentation module would display each component model separately

according to Figure 11 where external input(s), output(s), feedback

and other input/output interface are easily displayed.

Documentation of models specified partially by part of their static

structures (input and output variables only) can be very useful in

several application areas such as documentation of organizations, large software, or hardware systems.

misspell the names of component models or the input variables. Fur-

thermore if an input variable has been declared as an external input

(input of the resultant system) then the system can finish the speci-

fication of input/output relationship. For example in the model given

in Figure 8, the line

M.B <--- Z.IN; (* EXTERNAL INPUT *)

has to be generated fully by the modelling system, thus eliminating

also the possibility of a wrong input/output connection. If the

ranges of the values of the input and output variables are also speci-

fied, for every (input, output) pair specified in the coupling, the

system can also check whether or not the range of the output is a sub-

set of the range of the input and hence detect inconsistencies.

Documentation

Due to GEST's well-structured nature, computer generated structured

flowcharts of GEST models can easily be obtained. ORGEST (Organized

Representation of GEST programs) is being implemented for this purpose

(Oren et al. 1983). As an example, Figure 10 depicts a structured

flowchart of the memoryless model of Figure 5.

Coupling specification can be very useful in the documentation of sys-

tems consisting of large number of interacting component systems. A

documentation module would display each component model separately
STEP 1 - Specify: 1) name of the model,  
2) input and output variables, and  
3) ranges of acceptable values of input and output variables

PICTORIAL REPRESENTATION:

GEST MODELLING:

COUPLED MODEL Z

EXTERNALS

INPUT IN;  
RANGE OF IN = REAL(>= 0.00, <= 100.00);

OUTPUT OUT;  
RANGE OF OUT = TEAL(>= 40.00, < 75.00);

END EXTERNALS;

Figure 9a. Step 1 in top-down model conception and step-wise model refinement in GEST
STEP 2 - Specify names of component models

PICTORIAL REPRESENTATION:

GEST MODELLING:

COMPONENT MODELS M, N;

Figure 9b. Step 2 in top-down model conception and step-wise model refinement in GEST
STEP 3 - Specify each component model separately

PICTORIAL REPRESENTATION:

GEST MODELLING:

CONTINUOUS MODEL M;
STATIC STRUCTURE
  INPUTS A, B, C;
  STATE ...;
  OUTPUTS D, E;
  ...;
END STATIC STRUCTURE;
DYNAMIC STRUCTURE
  DERIVATIVES
  ...;
END DERIVATIVES;
OUTPUT FUNCTION
  ...;
END OUTPUT FUNCTION;
END DYNAMIC STRUCTURE;
END MODEL M;

Figure 9c. Step 3 in top-down model conception and step-wise model refinement in GEST
STEP 3 - Specify each component model separately

PICTORIAL REPRESENTATION:

GEST MODELLING:

CONTINUOUS MODEL N;

STATIC STRUCTURE

INPUTS F, G;
STATES ...
OUTPUTS H, K;
...
END STATIC STRUCTURE;

DYNAMIC STRUCTURE

DERIVATIVES
...
END DERIVATIVES;

OUTPUT FUNCTION
...
END OUTPUT FUNCTION;

END DYNAMIC STRUCTURE;

END MODEL N;

Figure 9d. Step 3 (for the second component model of Z)
STEP 4 - Specify

1) for every external input
   corresponding internal input(s)

2) for every external output
   corresponding internal output

PICTORIAL REPRESENTATION:

GEST MODELLING:

EQUIVALENCING

INPUTS Z.IN = M.B;
OUTPUTS Z.OUT = N.H;
END EQUIVALENCING;

Figure 9e. Step 4 in top-down model conception and step-wise model refinement in GEST
STEP 5 - Specify coupling of component models, i.e.,
for every component model
for every internal input
specify
from which output variable of which component model the values are provided

PICTORIAL REPRESENTATION:

GEST MODELLING:

COUPLING FOR Z
M.A <- N.H;
M.B <- Z.IN; (* EXTERNAL INPUT *)
M.C <- M.E;
N.F <- M.D;
N.G <- N.K;
END COUPLING FOR Z;

Figure 9f. Step 5 in top-down model conception and step-wise model refinement in GEST
MEMORYLESS MODEL BIRTH_RATE

STATIC STRUCTURE

INPUTS
S, (* MATERIAL STANDARD OF LIVING *)
NE, (* EFFECTIVE POLLUTION *)
P; (* POPULATION *)
OUTPUT BIRTH_RATE;
PARAMETERS K20, K21, K22, K23, K24, K25;

END STATIC STRUCTURE;

DYNAMIC STRUCTURE

B = K20 - K21*S - K22*NE - K23*P;

IF B >= K24 AND B <= K25
THEN BIRTH_RATE = B;
ELSE IF B < K24
THEN BIRTH_RATE = K24;
ELSE BIRTH_RATE = K25;
END IF;
END IF;
END DYNAMIC STRUCTURE;
END MODEL BIRTH_RATE;

Figure 10. Structured documentation of the memoryless model given in Figure 5
Figure 11. Documentation of the input/output relationships of one of the component models of a coupled model (which is represented in Figure 6)
3.5 Model Parameter Sets

In a parameter set basically two things are done: 1) The values of the model parameters are specified and 2) The tabular functions used in a model are defined point by point. There are four possibilities for the specification of model parameter sets, i.e., specification of single or multiple parameter set for a model consisting of one or several component models. Examples of model parameter set specifications are included in the system models given in section 5. It is also possible to specify discontinuous tabular functions. For every discontinuity point, two points have to be given with the same abscissa as follows:

\[(X_c, Y_l) / (X_c, Y_r)\]

where

- \(X_c\) is the common abscissa
- \(Y_l\) is the left-hand value of the ordinate
- \(Y_r\) is the right-hand value of the ordinate

In a computer-aided documentation system two- or three-dimensional tabular functions can also be displayed graphically for the user's convenience. Graphic display of the tabular functions may also help the user to detect some of the specification errors.

4. EXPERIMENTATION

4.1 Basics

The specification of the experiments, as shown in Figure 12, may comprise up to three sections which are:

1. experimental frames
2. model/frame pairs (or simulation runs)
3. post study section
The values of the functions used in a possibility for the specification of single or several discontinuities for every discontinuity specification may also help for the user's needs.

It is also possible to specify a three-dimensional experimental frame for experimental conditions may also help for the user's needs.

Figure 12. Elements of specifications of experiments in GEST

<table>
<thead>
<tr>
<th>Experiment frame(s)</th>
<th>Post study section</th>
<th>End run section</th>
<th>Simulation run(s) (i.e., (model, frame) pairs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>(parametric model, parameter set)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Experimental frame #m</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Specific frame for component model #p</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Global frame</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Experimental frame #1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Specific frame for component model #1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>End run #1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Experimental frame #l</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Specific frame for component model #l</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>End run #l</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Simulation run(s)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>End run #1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Post study section</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Experiment frame(s)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Specific frame for component model</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>End experimental frame #m</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Specific frame for component model #m</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>End experimental frame #1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Specific frame for component model #1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>End run #1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Simulation run(s)</td>
</tr>
</tbody>
</table>
An experimental frame specifies the simulation experiments. It can be conceived of having two parts which are global frame and specific frame.

In the global frame, overall specifications such as time unit declaration, termination condition specification, etc, are given.

Specific frames given for every component model separately, include initialization of the state variables, data collecting requirements, and specification of communication intervals and input values.

A model-frame pair specifies the combination of the parametric model, the parameter set, and the experimental frame to be used in a simulation study. It also includes a post run section.

The post run section includes specification of the additional computations to be performed after the simulation and outputting the results of a simulation run. The post study section resembles to a post-run section, with an additional level of generalization. In a post-study section, one can refer to data generated during different simulation runs. The output module consists of the specifications for displaying the results of the simulation study on different types of output units. Examples are included in the three system models given in section 5.

4.2 BNF Definitions

Some relevant definitions in BNF follow:

```
exper iment =
    { experimental-frame }
    { model-frame-pair }.
```
It can be

me and specific

e unit declara-

tion, includ-
ging requirements,
values.

Parametric model,
used in a simula-

Download the results
to a post-run
In a post-study

ent simulation
for displaying

otypes of output
given in sec-

experimental-frame =
  single-experimental-frame
| multiple-experimental-frame .

single-experimental-frame =
  "FRAME" identification "FOR" model-identifier
  global-frame
  { specific-frame }
  "END FRAME" identification ";" .

multiple-experimental-frame =
  "MULTIPLE FRAME" multiple-identification
  "FOR" model-identifier
  global-frame
  { specific-frame }
  "END MULTIPLE FRAME" multiple-identification ";" .

global-frame =
  "GLOBAL"
  time-unit-declaration
  | termination-condition
  | integration
  | activation-of-discrete-models
  | parameter-assignment
  | data-for-tabular-function
  | interpolation-type
  | communication-interval
  | renaming-declaration
  "END GLOBAL" ";" .
specific-frame =
  "MODEL" model-identifier
   [ "":" unsigned-integer "TO" unsigned-integer ]
   parameter-assignment
   | data-for-tabular-function
   | interpolation-type
   | communication-interval
   | renaming-declaration
   | initialization
   | input-scheduling
   | activation-of-discrete-model
   | output-sampling
   | data-collecting
   "END MODEL" model-identifier ";" .

model-frame-pair =
  single-run
  | multiple-run [ post-study-specification ] .

single-run =
  "RUN" identification
   "TO OBSERVE MODEL" model-identifier
   [ "WITH PARAMETER SET" identification ]
   "IN FRAME" identification
   [ "WITH POST RUN"
     [ statement | output-module-reference ]
   "END POST RUN" ";" ]
   "END RUN" identification ";" .

Examples:
  RUN 1 TO OBSERVE MODEL POPULATION_GROWTH
       WITH PARAMETER SET HIGH_BIRTH_RATE
       IN FRAME LOW_EMPLOYMENT;
   END RUN 1;

  RUN 7 TO OBSERVE MODEL EVAPORATION_FROM_SOIL
       WITH PARAMETER SET CLAY
       IN FRAME RAINY_SEASON ;
   END RUN 7;
5. EXAMPLES:

The following examples of complete GEST programs are provided. In all of them comments are kept at a minimum level, i.e., to provide a list of variables only.

Example 1 - DEER POPULATION

Study consists of:

One component model
One parameter set, and
One experimental frame

Example 2 - EVAPORATION FROM SOIL

Study consists of:

One component model
One parameter set, and
One experimental frame

Example 3 - MOTOR CONTROLLER

Study consists of:

A coupled model
One parameter set, and
One experimental frame
PROGRAM STUDY_OF_DEER_POPULATION


CONTINUOUS MODEL DEER_POPULATION

STATIC STRUCTURE
STATE DP;
OUTPUT DP, PP, DKPP;
AUXILIARY VARIABLES
DNI,
DPR,
DD;
PARAMETERS AREA,
NIR;
TABULAR FUNCTIONS
PPT,
DKPPT;
INTERPOLATIONS
PP,
DKPP;
PP = PPT(TIME);
DKPP = DKPPT(DD);
END INTERPOLATIONS;
END STATIC STRUCTURE;

DYNAMIC STRUCTURE
DERIVATIVES
DP' = DNI - DPR;
DNI = NIR*DP;
DPR = PP*DKPP;
DD = DP/AREA;
END DERIVATIVES;
END DYNAMIC STRUCTURE;
END MODEL DEER_POPULATION;

PARAMETER SET I
AREA = 3000
TABULAR FUNCTION
(* GIVE (1860,)
END TABULAR
TABULAR FUNCTION
(* GIVE
(0.0,
(0.015,
END TABULAR
END PARAMETER SET I

FRAME 1 FOR DEER POPULATION
GLOBAL
TIME UNITS
SIMULATE
INTEGRATE
END GLOBAL;
MODEL DEER_POPULATION
INITIALIZE
SAVE DP,
INTERPOLATE
END MODEL DEER_POPULATION
END FRAME 1;

RUN 1 TO OBSERVE
WITH POST RUN
OUTPUT MODULE
END POST RUN;
END RUN 1;

OUTPUT MODULE 1
PRINT 1 HEADLINE
STUDY OF DEER POPULATION
PLOT DP, PP, IN
END OUTPUT MODULE
END PROGRAM STUDY_OF_
PARAMETER SET 1 FOR DEER_POPULATION
AREA = 800000.0; NIR = 0.2;
TABULAR FUNCTION PPT WITH 2 POINTS
(* GIVES PP - PREDATOR POPULATION AS A FUNCTION OF TIME *)
(1800., 300.)(1960., 300.)
END TABULAR FUNCTION PPT;
TABULAR FUNCTION DKPPT WITH 6 POINTS
(* GIVES DKPP - DEER KILL PER PREDATOR
AS A FUNCTION OF DD - DEER DENSITY *)
(0.0, 0.0) (0.005, 3.0) (0.010, 13.0)
(0.015, 32.0) (0.020, 51.0) (0.025, 56.0)
END TABULAR FUNCTION DKPPT;
END PARAMETER SET 1;

FRAME 1 FOR DEER_POPULATION
GLOBAL
TIME UNIT IS YEAR
SIMULATE UNTIL TIME = 1970, START TIME = 1880;
INTEGRATE BY RUNGE KUTTA, REL_ERROR = 0.001;
END GLOBAL;
MODEL DEER_POPULATION
INITIALIZE STATE DP = 4000.
SAVE DP, PP, DKPP AT EVERY YEAR;
INTERPOLATION LINEAR PP, DKPP;
END MODEL DEER_POPULATION;
END FRAME 1;

RUN 1 TO OBSERVE MODEL DEER_POPULATION WITH PARAMETER SET 1 IN FRAME 1
WITH POST RUN
OUTPUT MODULE 1 ON PRINTER;
END POST RUN;
END RUN 1;

OUTPUT MODULE 1
PRINT 1 HEADING LINE;
STUDY OF DEER POPULATION
PLOT DP, PP, DKPP VERSUS TIME;
END OUTPUT MODULE 1;
END PROGRAM STUDY_OF_DEER_POPULATION;
PROGRAM STUDY_OF_EVAPORATION_FROM_SOIL
(* This model is adopted from:
The concept of matric flux potential applied to simulation
GEST 78 Version of this model appeared in Oren and Den Dulk 1978.
(To ease the comparison with the original version,
the units used in the original version are kept.
However, for SI units, note that 1 mbar = 100 Pascal)

LIST OF IDENTIFIERS:
EA - WATER VAPOR PRESSURE IN ATMOSPHERE (MBAR) (MILLIBAR)
EC - WATER VAPOR PRESSURE AT SOIL SURFACE
     (CURRENT PRESSURE) (MBAR)
ES - WATER VAPOR PRESSURE AT SOIL SURFACE
     (AT SATURATION) (MBAR)
F - FACTOR ACCOUNTING FOR THE EFFECT OF
    WIND ON EVAPORATION (CM/DAY.MBAR)
FK - HYDRAULIC CONDUCTIVITY AS A FUNCTION
    WATER CONTENT
FLW - FLUX OF WATER (CM**3/(CM**2).DAY)
FLW(I) - EVAPORATION (I.E. FLOW FROM TOP COM-
    PARTMENT (CM**3/(CM**2).DAY)
FPSI - PSI AS A FUNCTION OF WATER CONTENT
K(-) - HYDRAULIC CONDUCTIVITY (CM/DAY)
MFLP(-) - MATRIX FLUX POTENTIAL (CM**3/DAY) * -1
MFLPT - MFLP AS A FUNCTION OF WATER CONTENT
PSI(-) - WATER POTENTIAL (CM H2O) * 1
RDF(-) - RECIPROCAL OF THE DISTANCE BETWEEN
     CENTRES OF COMPARTMENTS (1/CM)
RTC(-) - RECIPROCAL OF COMPARTMENT THICKNESS (1/CM)
TCM(-) - COMPARTMENT THICKNESS (CM)
TEVAP - ACCUMULATED EVAPORATION (CM)
W(-) - WATER CONTENT (FRACTION) (CM**3/(CM**3)

COMMENT ON THE PHYSICAL LAW FOR FLUX:
ALL FLUXES ARE CALCULATED WITH DARCY'S LAW FOR
VERTICAL FLOW OF WATER IN THE SOIL:

\[ \text{FLUX} = \text{COND} \times \left( \frac{\text{-D}_\text{PSI}}{\text{D}_2} \right) \times \text{-1} \]
CONTINUOUS MODEL EVAPORATION FROM SOIL

STATUT STRUCTURE

STATE

TEVAP,
W(1..18);  (* ACCUMULATED EVAPORATION *)
(* CURRENT WATER CONTENT *)

OUTPUT

TEVAP,
W(1..18),
PSI(1..8),  (* WATER POTENTIAL *)
K(1..8),   (* HYDRAULIC CONDUCTIVITY *)
FLW(1);  (* FLUX OF WATER FROM TOP COMPARTMENT, I.E. EVAPORATION *)

AUXILIARY VARIABLE

FLW(1..19), K(1..19),
MFLP(1..18), PSI(1..18);

PARAMETER

TCM(1..18);  (* COMPARTMENT THICKNESS *)

AUXILIARY PARAMETERS

RTC(1..18),  (* RECIPROCAL OF COMPARTMENT THICKNESS *)
RDF(1..18);  (* RECIPROCAL OF THE DISTANCE BETWEEN CENTERS OF COMPARTMENTS *)

FOR I=1 TO 18 DO

RTC(I) = 1.0/TCM(I);
LOOP

RDF(1) = RTC(1);
FOR I=2 TO 18 DO

RDF(I) = 2.0/ (TCM (I-1) + TCM (I));
LOOP
END AUXILIARY PARAMETERS;

TABULAR FUNCTIONS

FK,  (* HYDRAULIC CONDUCTIVITY AS A FUNCTION OF WATER CONTENT *)
FPSI,  (* PSI AS A FUNCTION OF WATER CONTENT *)
MFLPT;  (* MFLP AS A FUNCTION OF WATER CONTENT *)

---

led to simulation
Sci., PP. 63-82.
Oren and Den Dulk 1978.
version, kept.
100 Pascal)

(MBAR) (MILLIBAR)
E
(MBAR)
E
(MBAR)

(cm/DAY . MBAR)

(cm**3/(cm**2), DAY)

(cm**3/(cm**2), DAY)

(cm/DAY)
(cm**3/DAY) * -1

(cm H2O) * 1

(1/cm)
(1/cm)
(cm)
(cm)
(cm**3/cm**3)

FOR

*)
INTERPOLATIONS
K(I) = FK(W(I));
PSI(I) = FPSI(W(I));
MFLP(I) = MFLPT(W(I));
END INTERPOLATIONS;
END STATIC STRUCTURE;

DYNAMIC STRUCTURE
DERIVATIVES
TEVAP' = FLW(I);

FOR I = 1 TO 18 DO
  W(I)' = (FLW(I+1) - FLW(I))*RTC(I);
END LOOP

(* CALCULATION OF EVAPORATION
(I.E. FLOW FROM TOP COMPARTMENT = FLW(I))
EQUATION IS:
EVAPORATION = F*(EC-EA)/(ES-EA)
F = 0.8 CM/DAY (DEPENDS ON WINDSPEED)
ES = 31.45 MBAR (25C, AT SATURATION)
EA = 7.06 MBAR (20C, 30] RELATIVE HUMIDITY)
F/(ES-EA) = 0.8/(31.45-7.06) = 0.0328
EC = ES/EXP(PSI*7.127E-7) *)

FLW(I) = MAX ( 0.0,
            0.0328*(31.45/EXP(7.127E-7 * PSI(I)) - 7.06)

(* CALCULATION OF FLUX, USING WATER POTENTIAL
AND HYDRAULIC CONDUCTIVITY INDEPENDENTLY *)

FOR I = 2 TO 18 DO
  FLW(I) = (RDF(I) * (PSI(I-1) - PSI(I)) - 1.)
      * (K(I-1) + K(I))/2.0;
END LOOP

FLW(19) = 0.0;
END DERIVATIVE;
END DYNAMIC STRUCTURE;
END MODEL EVAPORATION_FROM_SOIL;
PARAMETER SET 1 FOR EVAPORATION FROM SOIL

$T_{CM(1..18)} = 5*1.0, 5*1.5, 3*2.5, 4*5.0, 10.0;$

**TABULAR FUNCTION FK WITH 17 POINTS**

(*GIVES THE HYDRAULIC CONDUCTIVITY K(-) AS A FUNCTION OF WATER CONTENT W(-)*)

<table>
<thead>
<tr>
<th>$W$</th>
<th>$K$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.03</td>
<td>2.4E-10</td>
</tr>
<tr>
<td>0.145</td>
<td>2.4E-6</td>
</tr>
<tr>
<td>0.2175</td>
<td>6.0E-4</td>
</tr>
<tr>
<td>0.2925</td>
<td>1.0E-1</td>
</tr>
<tr>
<td>0.4</td>
<td>7.0</td>
</tr>
</tbody>
</table>

END TABULAR FUNCTION FK;

**TABULAR FUNCTION FPSI WITH 13 POINTS**

(*GIVES THE MATRIX WATER POTENTIAL (-1) PSI(-) AS A FUNCTION OF WATER CONTENT W(-)*)

<table>
<thead>
<tr>
<th>$W$</th>
<th>$PSI$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>1.0E7</td>
</tr>
<tr>
<td>0.165</td>
<td>1.0E4</td>
</tr>
<tr>
<td>0.275</td>
<td>300.0</td>
</tr>
<tr>
<td>0.41</td>
<td>-600.0</td>
</tr>
</tbody>
</table>

END TABULAR FUNCTION FPSI;

**TABULAR FUNCTION MFLPT WITH 30 POINTS**

(*GIVES THE MATRIC FLUX POTENTIAL (*-1) MFLP(-) AS A FUNCTION OF WATER CONTENT W(-)*)

<table>
<thead>
<tr>
<th>$W$</th>
<th>$MFLP$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>43.0</td>
</tr>
<tr>
<td>0.189</td>
<td>42.380</td>
</tr>
<tr>
<td>0.2275</td>
<td>41.65</td>
</tr>
<tr>
<td>0.261</td>
<td>38.88</td>
</tr>
<tr>
<td>0.281</td>
<td>33.68</td>
</tr>
<tr>
<td>0.297</td>
<td>27.7</td>
</tr>
<tr>
<td>0.307</td>
<td>19.0</td>
</tr>
<tr>
<td>0.32</td>
<td>5.3</td>
</tr>
</tbody>
</table>

END TABULAR FUNCTION MFLPT;

END PARAMETER SET 1;
FRAME 1 FOR EVAPORATING_FROM_SOIL

GLOBAL
    SIMULATE UNTIL TIME = 5.0;
    INTEGRATE BY RUNGE_KUTTA;
END GLOBAL;

MODEL EVAPORATION FROM SOIL
    INITIALIZE STATES
        TEVAP = 0.;
        W(1..18) = 18*0.2925;

    COMMUNICATE AT EVERY 0.25 TIME UNIT;

    SAVE W(1..16), PSI(1..8), K(1..8), FLW(1), TEVAP;

    INTERPOLATION LINEAR K, PSI, MFLP;

END MODEL EVAPORATION_FROM_SOIL;

END FRAME 1;

RUN 1 TO OBSERVE MODEL EVAPORATION_FROM_SOIL
    WITH PARAMETER SET 1 IN FRAME 1
    WITH POST RUN
        OUTPUT MODULE 1 ON PRINTER
END POST RUN
END RUN 1

OUTPUT MODULE 1
    PRINT FOR FIRST PAGE 1 HEADING LINE;
    STUDY OF EVAPORATION FROM SOIL - VERSION 1
    LIST TIME, W(1..16), PSI(1..8), K(1..8), FLW(1);
    PLOT TEVAP VERSUS TIME;
    PLOT FLW(1) VERSUS TIME WHILE (TIME <= 1.0);
    PLOT W(1) VERSUS TIME WHILE (TIME <= 0.5);
END OUTPUT MODULE 1;

END PROGRAM STUDY_OF_EVAPORATION_FROM_SOIL;
PROGRAM STUDY_OF_MOTOR_CONTROLLER

(* This model is adapted from:
Report 7502, Dept. of Automatic Control,
Lund Institute of Technology, Sweden *)

COUPLED MODEL MOTOR_CONTROLLER

EXTERNAL

INPUT YREF;
END EXTERNAL;

COMPONENT MODELS PID_CONTROLLER, MOTOR;

CONTINUOUS MODEL PID_CONTROLLER;

STATIC STRUCTURE

INPUTS YREF, Y;
STATES I, X;
OUTPUT U;
AUXILIARY VARIABLES E, P, D;
PARAMETERS G, GD, TD, TI;
END STATIC STRUCTURE;

DYNAMIC STRUCTURE

DERIVATIVES

I' = E/TI;
X' = -GD/TD*(X - y);
E = YREF - Y;
END DERIVATIVES;

OUTPUT FUNCTION

U = P+I+D;
P = G*E;
D = -GD*(Y - X);
END OUTPUT FUNCTION;
END DYNAMIC STRUCTURE;
END MODEL PID_CONTROLLER;
CONTINUOUS MODEL MOTOR

STATIC STRUCTURE
INPUT U;
STATES TH, THDOT;
OUTPUT Y;
AUXILIARY VARIABLES ME, I;
PARAMETERS KM, R, J, CT;
END STATIC STRUCTURE;

DYNAMIC STRUCTURE
DERIVATIVES
TH' = THDOT;
THDOT' = ME/J;
ME = KM*I;
I = (U-KM*THDOT)/R;
END DERIVATIVES;

OUTPUT FUNCTION
Y = CT*TH;
END OUTPUT FUNCTION;
END DYNAMIC STRUCTURE;
END MODEL MOTOR;

END COMPONENT MODELS;

EQUIVALENCING
INPUTS
MOTOR_CONTROLLER.YREF = PID_CONTROLLER.YREF;
END EQUIVALENCING;

COUPLING FOR MOTOR_CONTROLLER
PID_CONTROLLER.YREF <--- MOTOR_CONTROLLER.YREF;
(* EXTERNAL INPUT *)
PID_CONTROLLER.Y <--- MOTOR.Y;
MOTOR.U <--- PID_CONTROLLER.U;
END COUPLING FOR MOTOR_CONTROLLER;

END MODEL MOTOR_CONTROLLER;
PARAMETER SET 1 FOR MOTOR CONTROLLER

MODEL PID_CONTROLLER
G = 1.0;
GD = 1.0;
TD = 1.0;
TI = 1.010;
END MODEL PID_CONTROLLER;

MODEL MOTOR
KM = 6.2 E-3;
R = 5.3;
J = 7.5 E-7;
CT = 0.033;
END MODEL MOTOR;

END PARAMETER SET 1;

FRAME 1 FOR MOTOR CONTROLLER
GLOBAL
TIME UNIT IS SECOND;
SIMULATE UNTIL TIME = 5.0;
INTEGRATE BY RUNGE KUTTA, REL_ERROR = 0.0001;
COMMUNICATE AT EVERY 0.01 SECOND;
END GLOBAL;

MODEL PID_CONTROLLER
INITIALIZE STATES TO ZERO;
SAVE U;
END MODEL PID_CONTROLLER;

MODEL MOTOR
INITIALIZE STATES
TH = 0.0;
YHDOT = 0.0;
SAVE Y;
END MODEL MOTOR;

END FRAME 1;
RUN 1 TO OBSERVE MODEL MOTOR_CONTROLLER
 WITH PARAMETER SET 1 IN FRAME 1;

 WITH POST RUN
 OUTPUT MODULE 1 ON PRINTER;
 END POST RUN;
 END RUN 1;

OUTPUT MODULE 1

PRINT FOR FIRST PAGE 1 HEADING LINE;
 STUDY OF MOTOR_CONTROLLER
 PLOT AND LIST U, V VERSUS TIME;

END OUTPUT MODULE 1;

END PROGRAM STUDY_OF_MOTOR_CONTROLLER;

Aytaç, K., Özer (1976)
MAGEST: A Knowledge-Based System

Bleha, L.J. (1977)
Combined Digital and Analog Simulation
University of Ottawa, Ont.

Continuous-Chain Method
Ottawa, Ont.

Elzas, M.S. (1975)
Simulation methodology in Speech
Elzas, G.J. Berlin
57-91.

Hettenis, D.L. (1977)
Simulation of Speech
Congress, Soc.
pp. 291-298.

Üren, T.I. (1971a)
Combined Digital Simulation
of Arizona, TIA

Üren, T.I. (1971b)
Large Scale Simulation International Congress
International Congress
B-11/4.
LIST OF REFERENCES
(And BIBLIOGRAPHY OF GEST)


APPENDIX: THE METALANGUAGE USED TO DEFINE GEST 81

- Terminal symbols of the GEST 81 language are enclosed within quotation marks and are written in UPPER CASE CHARACTERS.

- Non-terminal symbols of the GEST 81 language are written in lower case characters.

- When a non-terminal symbol requires more than one word, hyphen "-" is used between-the-words.

- A blank separates two syntactic units.

- A period "." indicates end of a rule.

- Left and right sides of any rule of the grammar are separated by the equivalence symbol "=" which can be read "is" or "can be formed from."

- Exclusive or is represented by the symbol "|" which can be read "or"

- Parentheses "( )" indicate grouping without repetition.

- Square brackets "[ ]" indicate option, that is, zero or one occurrence.

- Curly brackets "{ }" indicate repetition, that is, one or more occurrences.

- Zero or more occurrences of a syntactic unit is represented by a combination of square and curly brackets, i.e., by "[{{ }}]."
Simulation and Model-Based Methodologies: An Integrative View

Edited by
Tuncer I. Ören
Computer Science Department, University of Ottawa
Ottawa, Ontario, Canada

Bernard P. Zeigler
Computer Science Department, Wayne State University
Detroit, Illinois, USA

Maurice S. Elzas
Computer Science Department, Dutch Agricultural University
Wageningen, The Netherlands

Springer-Verlag Berlin Heidelberg New York Tokyo 1984
Published in cooperation with NATO Scientific Affairs Division
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Author(s)</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Conceptual Bases for System Modelling and Design</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Model-Based Activities: A Paradigm Shift</td>
<td>Tuncer I. Oren</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>System Paradigms as Reality Mappings</td>
<td>Maurice S. Elzas</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>General Systems Framework for Inductive Modelling</td>
<td>George J. Klir</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td>System Theoretic Foundations of Modelling and Simulation</td>
<td>Bernard P. Zeigler</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td>The Tricotyledon Theory of System Design</td>
<td>A. Wayne Wymore</td>
<td>119</td>
</tr>
<tr>
<td></td>
<td>Concepts for Model-Based Policy Construction</td>
<td>Maurice S. Elzas</td>
<td>133</td>
</tr>
<tr>
<td>2</td>
<td>Model-Based Simulation Architecture</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Structures for Model-Based Simulation Systems</td>
<td>Bernard P. Zeigler</td>
<td>185</td>
</tr>
<tr>
<td></td>
<td>Symbolic Manipulation of System Models</td>
<td>Franz Pichler</td>
<td>217</td>
</tr>
<tr>
<td></td>
<td>Concepts for an Advanced Parallel Simulation Architecture</td>
<td>Len Dekker</td>
<td>235</td>
</tr>
<tr>
<td>3</td>
<td>Impact of Formalisms on Model Specification</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>GEST - A Modelling and Simulation Language Based on System Theoretic Concepts</td>
<td>Tuncer I. Oren</td>
<td>281</td>
</tr>
<tr>
<td></td>
<td>Continuous and Discontinuous-Change Models: Concepts for Simulation Languages</td>
<td>Roy E. Crosbie</td>
<td>337</td>
</tr>
<tr>
<td></td>
<td>Discrete Event Formalism and Simulation Model Development</td>
<td>Sudhir Aggarwal</td>
<td>357</td>
</tr>
<tr>
<td>4</td>
<td>Model Identification, Reconstruction, and Optimization</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Structure Characterization for Ill-Defined Systems</td>
<td>Jan A. Spriet and Ghislain C. Vansteenkiste</td>
<td>383</td>
</tr>
</tbody>
</table>