Journal Title: Modeling and Simulation Methodology in the Artificial Intelligence Era

Article Author: Rozenblit, J.W. and Zeigler, B.P.

Article Title: Entity-Based Structures for Model and Experimental Frame Construction

Volume:

Issue:

Month/Year: 1986

Pages: 78-100 **(scan notes and title/copyright pages for chapter requests)**

Imprint:

6/10/11
10:45 am

Paged by CR **(Initials)**

Reason Not Filled (check one):

☐ **NOS** ☐ **LACK VOL/ISSUE**

☐ **PAGES MISSING FROM VOLUME**

☐ **NFAC (GIVE REASON):**

Trans. #: 951841

Item #:

CUSTOMER INFORMATION:

Liana Napalkova
liananapalkova@email.arizona.edu

STATUS: Faculty
DEPT: Electrical/Computer Engr

University of Arizona Library
Document Delivery
1510 E. University Blvd.
Tucson, AZ 85721
(520) 621-6438
(520) 621-4619 (fax)
AskILL@u.library.arizona.edu

Modelling and Simulation Methodology
in the Artificial Intelligence Era
M.S. Elzas, T.I. Ören and B.P. Zeigler (Editors)

Chapter II.2

ENTITY-BASED STRUCTURES FOR MODEL AND EXPERIMENTAL FRAME CONSTRUCTION

Jerzy W. Rozenblit and Bernard P. Zeigler

Department of Electrical and Computer Engineering
The University of Arizona
Tucson, Arizona 85721
U.S.A.

This chapter sets up a conceptual framework for constructing knowledge-based environments to support a model and simulation program development process. The framework is based on the formal structures underlying the multifacetted modelling methodology, namely that of the system entity structure and experimental frame. It is argued that these structures represent the basic knowledge required to specify models and experimental conditions in simulation studies.

1. INTRODUCTION

The primary concerns of the modelling and simulation enterprise are the construction of models of real world systems, computer simulation of such models, and analysis of the simulation results. The ultimate benefit of modelling is to improve and increase decision making capabilities in engineering and business environments. The methodologies offered by the discipline should be an inherent component in computer-aided decision systems for management, control and design (Sprague and Carlson, 1982; Elzas, 1982). The tools and activities prescribed by such methodologies should enable the decision makers to evaluate the effects of decisions before they are actually carried out. The best - in terms of performance measures related to the system under evaluation - intervention alternatives should be chosen and deployed in the real system.

The choice of performance measures reflects the questions the decision maker (be it an economist, a designer of a car, or a technician supervising a manufacturing process) wants to ask about the system under consideration. The questions translate into the objectives of the simulation study that is undertaken to provide the answers about the system. Therefore, it is important to recognize the importance of the simulation objectives. They play the key role in orienting and driving both the model building and design of simulation experiments.

In this chapter we shall focus on the objectives-driven model development methodology (Zeigler, 1984) and its underlying formal structures. We propose a systematic model and experimental frame development methodology supported by adequate formal concepts. Our framework corresponds to the deducive modelling approach proposed by Elzas (1984). As defined by Elzas (1984) the deducive approach requires some a priori knowledge of the structure of a real system. It also assumes the existence of a methodology for the construction of experimental frames from proposed alternative system decompositions, obtained through the deductive process.

We first address the problem of representing the fundamental structures for the model and experimental frame development. Then, we proceed to define procedures that employ these structures and facilitate the model construction process.

2. THE SYSTEM ENTITY STRUCTURE

To appropriately represent a family of possible model structures we need a representation scheme that embodies knowledge about the following three relationships: decomposition, taxonomy, and coupling. By knowing about decomposition we mean that the scheme has a means for representing the manner in which an object is decomposed into components.

By taxonomic knowledge, we mean a representation for the kinds of variants that are possible for an object. Such variants are termed specializations. To construct a model, the components identified in decompositions and specializations must be coupled together. Thus, the third kind of knowledge that our scheme should have is that of coupling relationships.

A formal object that conforms to the above specification is the system entity structure (Zeigler, 1982, 1984, Belogus 1985).

The system entity structure is based on a tree-like graph encompassing the boundaries and decompositions that have been conceived for the system (Zeigler, 1982, 1984). An entity signifies a conceptual part of the system which has been identified as a component in one or more decompositions. Each such decomposition is called an aspect. Thus entities and aspects should be thought of as components and decompositions, respectively. The system entity structure organizes possibilities for a variety of system decompositions and model constructions.

Both entities and aspects can have attributes represented by the so called attached variable types. When a variable type V is attached to an item occurrence I, this signifies that a variable I.V may be used to describe the item occurrence I. Thus, while an unqualified variable type such as LENGTH may have multiple occurrences in the entity structure, a qualified variable e.g. QUEUE1.LENGTH belongs to one and only one item occurrence, QUEUE1.

Among many important features of the system entity structure we would like to emphasize the following three:

a.) coupling constraints on the possible ways in which components (represented by entities) identified in decompositions (represented by aspects) can be coupled together are attached to aspects. This plays a crucial role in the hierarchical model construction process (Zeigler, 1984);

b.) there is a special type of decomposition called a multiple decomposition that allows for a flexible representation of multiple entities whose number in a model may vary;

c.) there is a special relation termed specialization which allows for representation of objects with individual attributes, yet inheriting the variables of a general class to which they belong. The specialization is modelled after the class concept of SIMULA and more recent object oriented languages.

Figure 1., depicts a system entity structure for an automobile. The triple vertical lines denote multiple decompositions while the double vertical lines stand for specializations. We shall refer to this entity structure throughout the ensuing sections.

The representation of attached variables as item-name.variable-type pairs has profound implications for design of modelling support environments. A data base of generic variable types should be an indispensable component of a software package for the system entity structure and experimental frame specification. The user of such a system faced with a modelling problem that falls into a certain general class e.g. queuing systems, would refer to the generic variable base and choose variable types suitable for the class his problem belongs to. The same concerns the specification of the appropriate experimental frames via the concept of the generic frame type. At this point we proceed to define the other fundamental concept in our framework that is the experimental frame.

3. EXPERIMENTAL FRAME DEFINITION AND ITS STRUCTURAL REALIZATION

Zeigler (1984) has laid down the groundwork for the modelling methodology in which the statement of objectives is operationalized in a definition of experimental frames. The experimental frame as initially perceived by Oren and Zeigler (1979) defines a set of circumstances under which a model of the real system is to be observed and experimented with. We would like to emphasize the importance of the experimental frame concept in the following contexts.

First, a frame in the objectives-driven methodology directs the model building process. It also facilitates meaningful simplification and stating of relations the modeller seeks to establish between two models (Zeigler, 1984).

Secondly, in the context of computer assistance for simulation, tools and architectures for the multifacetted modelling, the frame concept allows for a clear separation of the model and experimentation specifications. This in turn results in modular simulation software designs. Such designs should incorporate the model/experimental frame separation in that separate modules for model, experiment and execution control specification are provided. This conceptual framework has in fact found its realization in the new simulation systems and software for both continuous and discrete event systems (Oren, 1982; Pegden, 1982; Crosby, 1983; Javor, 1982; Kettenis, 1983).

In view of the recent efforts leading towards knowledge-based simulation environments, the experimental frame concept is directly related to the selective model instrumentation framework postulated by Reddy, Fox and Husain (1985). The key facility in that framework is a rule-based system whose task is to generate experimental modules by consulting a domain specific knowledge base.

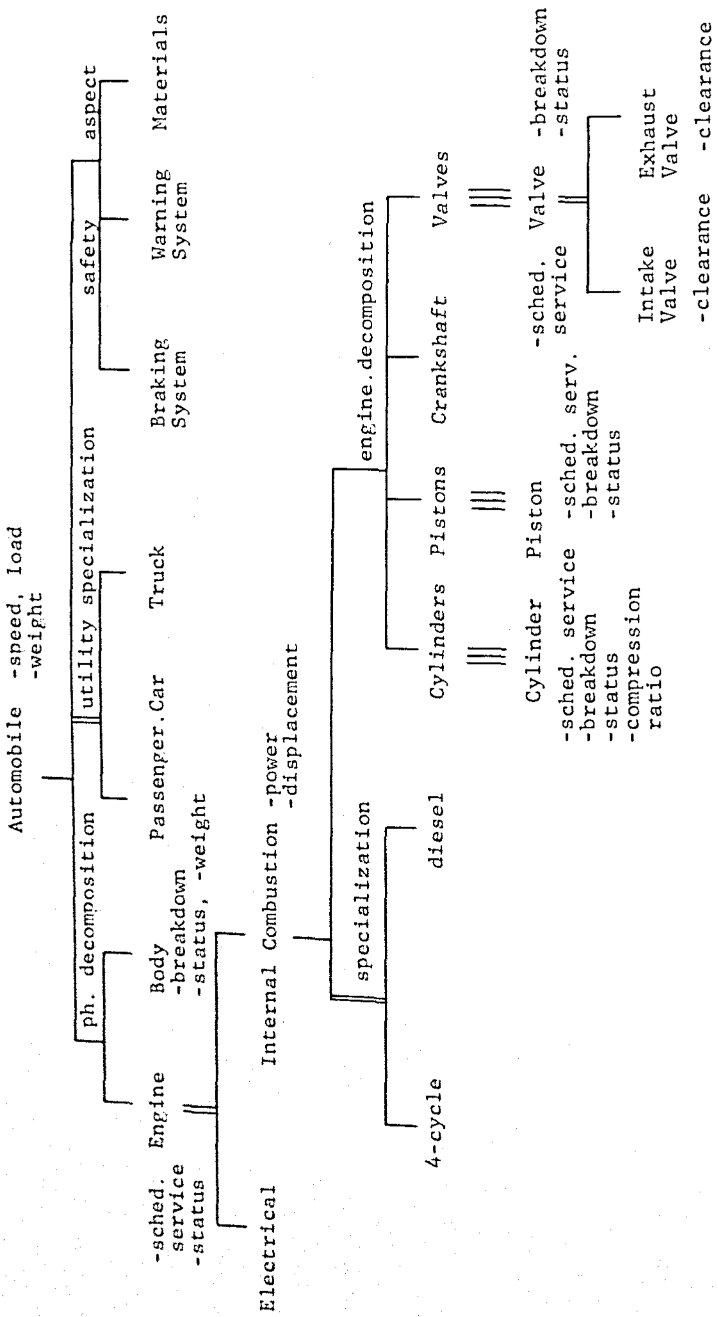

Figure 1. System Entity Structure for Automotive Modelling

Finally, the hierarchical frame specification (Rozenblit, 1985b) consolidates efforts to provide a unified framework for simulation of distributed, hierarchically specified systems.

Let us now briefly discuss the steps that lead to the specification of an experimental frame. The set of experimentation circumstances that we strive to define by means of a frame is perceived as consisting of four categories, namely: input, output, run control and summary variable sets. There are also constraints on the time segments of input and run control variables. Formally, the experimental frame is defined as follows:

$$EF=\langle T,I,O,C,W_I,W_C,SU,W_{SU}\rangle$$

where
- T is a time base
- I is the set of <u>input variables</u>
- O is the set of <u>output variables</u>
- C is the set of <u>run control variables</u>
- W_I is the set of <u>admissible input segments</u>, i.e. a subset of all time segments over the crossproduct of the input variable ranges
- W_C is the set of <u>run control segments</u>, i.e. a subset of all time segments over the crossproduct of the control variable ranges.
- SU is a set of summary variables
- $W_{su}=\{s \mid s: I \times O \rightarrow SU.range\}$ is the set of <u>summary mappings</u>

The I/O data space defined by the frame is the set of all pairs of I/O segments:

$$D=\{(w,r) \mid w \in (T,X)\ ,\ r \in (T,Y) \text{ and } dom(w)=dom(r)\}.$$

where X and Y are input and output value sets, respectively.

The reader is referred to (Zeigler, 1984) for a detailed exposition of the frame concept. Here we emphasize the meaning of the run control variables and segments. One should realize that they initialize the experiments and set up conditions for continuation as well as termination of simulation runs. The set of initialization conditions constitutes a subset of the control space called INITIAL. Similarly, the subset defined by the termination conditions is called TERMINAL. The run control segments can then be defined as follows:

$$W_C=\{m \mid m: \langle t_{initial}, t_{final}\rangle \rightarrow Z\}$$

where Z = crossproduct of the ranges of individual run control variables, and $m(t_{initial}) \in$ INITIAL, $m(t_{final}) \in$ TERMINAL.

We now relate the above definition of the experimental frame to the issue of simulation design. It is clear that a means of expressing a frame in the procedural form would greatly facilitate the generation of simulation programs. Other than the few simulation systems referenced above the state-of-the-art simulation languages do not capture the notion of experiment in a manner conducive to our description. However, prototypes for structural realization of experimental frames have been proposed by Zeigler (1984).

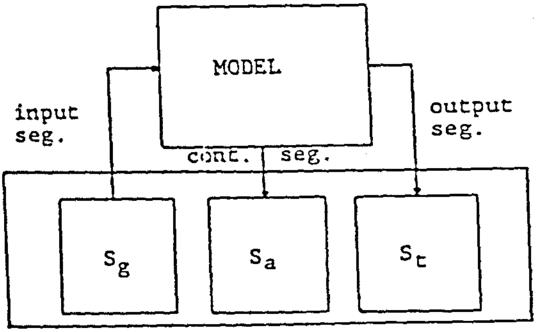

Figure 2. Structural Realization of an Experimental Frame

Employing the concepts of automata theory and the DEVS (discrete event system specification) formalism, Zeigler (1984) defines a DEVS generator, acceptor and transducer. The experimental frame E is then realized by a system S_E which is a parallel composition of systems S_I (an input segment generator or acceptor), S_C (a run control segments acceptor) and S_O (a composition of transducers, each of which realizes a summary mapping). Such a realization is depicted in Figure 2. Notice that the system S_E is coupled to the model of the system under study. Accordingly, the resulting simulation program should consist of procedures representing the model and frame specifications, respectively. Such tripartite realization of an experimental frame provides a very flexible means for user specification of the experimental circumstances in a simulation study. Moreover, basic DEVS devices for standard operations e.g. computation of the average of values, acceptance of a constant segments, can be used as elements of the computer-aider environment for the simulation program generation.

In the context of the above definition an experimental frame employed in a simulation program applies to a specific problem and answers the questions directly addressed to that problem. In other words, the variables, segments and initialization/termination conditions are defined in such a way that the frame applies to the specific model under study. However, we would like the frame generation to be supported by knowledge-based environments. It is thus natural to conceive any frame realization as a concretization of a general experimental frame type. Such a general frame type can be regarded as a generic experimental frame for certain classes of problems and types of performance evaluation criteria. Such standard criteria would include input/output performance indexes, utilization of resources measures, reliability assessments etc.

Also, recall that the experimental frame concept is discussed in this paper in the context of the objectives-driven simulation methodology. Thus, it is necessary to view the simulation design from a perspective in which the model and experimental frame development are complimentary and mutually supportive processes. In the following section we present the concept of the generic frame type and discuss its role in the model development methodology.

4. THE CONCEPT OF GENERIC EXPERIMENTAL FRAME

As we have indicated in Section 3., a generic frame should be a general class from which an experimental frame specification for a simulation study under consideration could be derived. The generic frame is defined by means of unqualified generic variable types that correspond to the objective for which the simulation is undertaken. Thus, a generic frame should be only a template whose instantiation takes place after the simulation model has been constructed. With an objective we associate a performance index that allows for a final judgement of the simulation model with respect to that objective. A generic experimental frame is defined as the following structure induced by a performance index pi.

$$GEF_{pi}=\{IG,OG,W_{GI},SU,W_{SU}\}$$

where: GEF_{pi} denotes a generic experimental frame for performance index pi and

IG is the set of generic input variable types for pi
OG is the set of generic output variable types for pi
W_{IG} is the set of generic input segment types for pi
SU is the set of summary variables
W_{SU} is the set of standard summary mappings

Notice that we have decided not to include the set of run control variables and segments in the above definition. We feel that the execution control conditions for a simulation run should be specified after the relevant generic frame has been instantiated and the model is ready to be experimented with.

To illustrate how a generic frame type may be specified let us consider a simple example. In many classes of problems one of the standard simulation objectives is to obtain measurements concerning utilization of system's components. A common measure associated with this objective is often called utilization and is expressed as follows:

Utilization = (total time a component is active/total observation time)

Let us now define a generic frame type that corresponds to this performance measure. It is easy to notice that in order to record the utilization of a component we must monitor its status, i.e. whether it is active or idle. We also need to define input variables and segments in order to observe how the component responds to a sequence of tasks arriving at the system. Then, a generic frame Utilization can have the following form:

Generic Frame Type: UTILIZATION
{comment: specifies a class of experimental frames for evaluation of component utilization in discrete event systems}

Generic Input Variables:
Arrival with range {0,1} where 1 denotes an event of arrival, 0 is an empty event

Generic Output Variables:
Status with range {0,1} where Status=0 denotes idle, Status=1 denotes active component

Generic Input Segment Type:
    InSeg_Arrival
        class: DEVS segment
        parameters: inter-arrival distribution type

{comment: after the generic frame is instantiated the segment description is matched with the specification of standard experimental frame input generators, so the input segment can be realized as a DEVS generator;

Generic Summary Variable
    Utilization

{comment: to obtain values for Utilization a standard DEVS transducer should be employed. Such a transducer will monitor the variable Status and record the ratio of time(Status=1)/ Total.Elapsed.Time}

Other examples of generic experimental frame types for various performance criteria are presented in (Rozenblit,1985).

In the context of the experimental frame generation, the generic frame constitutes a skeleton from which the experimental modules are constructed. We shall discuss this process, called instantiation, in Section 7. In the model development aspect, the key role of generic frames is to provide a means for selecting substructures of the system entity structure which accommodate the modelling objectives i.e., contain all the attached variable types present in the generic frames. This process is called pruning and will be explained in detail in the next section.

We are now ready to incorporate the system entity structure and generic frame types into the model and experimental frame development process. Recall that the entity structure represents a family of model structures (Zeigler, 1984) that are used in the model construction process. Each such structure has attributes expressed by means of variables and associated coupling constraints that restrict the way in which the components of the structure can be connected together. Also, it is important to remember that the simulation objectives are the driving mechanism in the model construction process. We are simply interested in obtaining the "simplest" (minimal, least complex) model that is capable of answering our questions about the real system. Thus, we need a means of extracting from the system entity structure all the model structures that meet the simulation objective, or in more specific terms, that accommodate the generic experimental frame expressing that particular objective. Consequently, the extracted model structures should support the instantiation of the generic frame in which they have been obtained.

In the following section we propose a framework for the entity structure-based model and experimental frame development.

5. ENTITY STRUCTURE PRUNING FOR GENERATION OF MODEL STRUCTURES

Given the system entity structure we are offered a spectrum of model alternatives due to the multiplicity of aspects and specializations. The question arises: "how can we meaningfully use the structure to support the model development process?"

Assume that an entity structure has been transformed into a structure with no specializations. Then, imagine that we traverse the structure selecting a single aspect for each entity and zero or more entities for each aspect. All selected entities carry their attributes with them. The coupling constraint of the selected aspect is attached to the entity to which this aspect belongs. The above process results in decomposition trees (Zeigler 1982, 1984) that represent hierarchical decompositions of models into components. We term such decomposition trees model structures. The process that extracts the model structures from the system entity structure is called pruning.

In what follows we present definition of the pruning process based on the generic experimental frame concept. By pruning the system entity structure with respect to generic frames we derive the following benefits:

a.) a generic frame extracts only those substructures which conform to the modelling objectives. Thus, a number of model alternatives may be disregarded as not applicable or not realizable for a given problem.

b.) partial models can be formulated and evaluated. This may significantly reduce the complexity which would arise if we had to deal with the overall model. The generic frame concept may thus be viewed as an object that partitions the system entity structure into modelling objectives related classes.

c.) the evaluation of the models constructed from the pruned substructures is performed in corresponding experimental frames. Such frames are generated by instantiating the generic frames used to prune the system entity structure. Hence, automatic evaluation procedures could be employed in the simulation design process.

In terms of facilitating the pruning process itself, generic frames automatically determine:

a.) the aspects that are selected for each entity

b.) the depth of the pruning process

c.) the descriptive variables of components

Having discussed the benefits afforded by the generic frame concept we now proceed to define the pruning procedure.

The pruning procedure presented here is defined for pure system entity structures, i.e., structures in which no specializations are present. Rozenblit (1895) presents a suite of algorithms that transform entity structures with specializations into pure system entity structures.

For the pruning process it is enough to restrict the generic experimental frame to the generic observation frame i.e.:

$$GOF = \{IG, OG\}$$

where IG denotes the set of generic input variable types, and OG is the set of generic output variable types. By defining the observation frame as above, we restrict its role to representing the behavioral aspects of modelling objectives. As we have already indicated, there are also objectives that constrain the structural aspects of the project under consideration. Therefore, as we shall see in the next section, in order to realize the structural constraints it will be necessary to augment the model development with a process that we term synthesis rule generation.

The pruning procedure is based on the depth first tree traversal. In this procedure every entity in each aspect is searched for occurrences of variable types that are present in the generic observation frame. The entities are attached to the model decomposition tree as the search progresses. At the same time the algorithm calls itself recursively for each entity being searched. The complete pruning procedure is given below:

Procedure Prune(E_j, CV_{GOF}, V_{GOF});
{ This procedure prunes the pure system entity structure and returns the model structures that accommodate the generic observation frame GOF. Multiple occurrences of a frame variable type are permitted in the model structures }

E_j - root of the pure entity structure
CV_{GOF} - set of variables of the generic frame GOF
this set is used to check if all the frame variables are present in the pruned substructure
initially $CV_{GOF} = V_{GOF}$
V_{GOF} set of input and output variable types of GOF

begin
for each aspect $A_i \in E_j$ do
begin

for each entity $E_k \in A_i$ do
begin

attach E_k with all its variables as a child of TE_j;
{ TE_j denotes the root of the model structure being currently built }

$CV_{GOF} := CV_{GOF} - v_k$;

{ update the current set CV_{GOF} by subtracting the variable types v_k such that v_k V_{GOF} and v_k is attached to E_k }

if E_k has at least one variable type present in V_{GOF}
then mark this level in the model structure as the last level at which variable types present in the frame have been found;

end; {of for each entity ... }

attach the coupling constraint of the aspect A_i to TE_j;

```
for each E_k ∈ A_i such that E_k has aspects do
  Prune(E_k, CV_GOF, V_GOF);


if C_GOF is empty   { i.e. the frame is accommodated }
then

   begin

     create a copy of the current model structure
     rooted by TE_j;

     { this copy will serve as a basis for model
       structure construction in the next aspect A_i+1 }

     output the current model structure rooted by TE_j
     without the entities that appear below the level
     marked as the last level with frame variable type
     occurrence;

   end; {of if}

update the current structure TE_j by cutting
off the last level entities;

{ thus prepare the structure for pruning in the
  next aspect}

end; {of for each aspect ... }

end. {of Prune}
```

To initialize the pruning process we follow the steps given below:

a.) in the system entity structure choose the entity E_i that represents the model you intend to evaluate (this entity will label the root of the model structure TE_i).

b.) create a dummy entity DE (with no variables) with a dummy aspect DA in which E_i is a subentity of DE.

c.) call Prune(DE, CV_{GOF}, V_{GOF});

After the procedure has been executed we have to eliminate DE from all the model structures.

We have already indicated that the procedure Prune generates a set of model structures in the form of decomposition trees. Each such structure accommodates the generic observation frame GOF and constitutes a skeleton for a hierarchical model construction. Figure 3 illustrates the results of pruning of the system entity structure with respect to frame GOF.

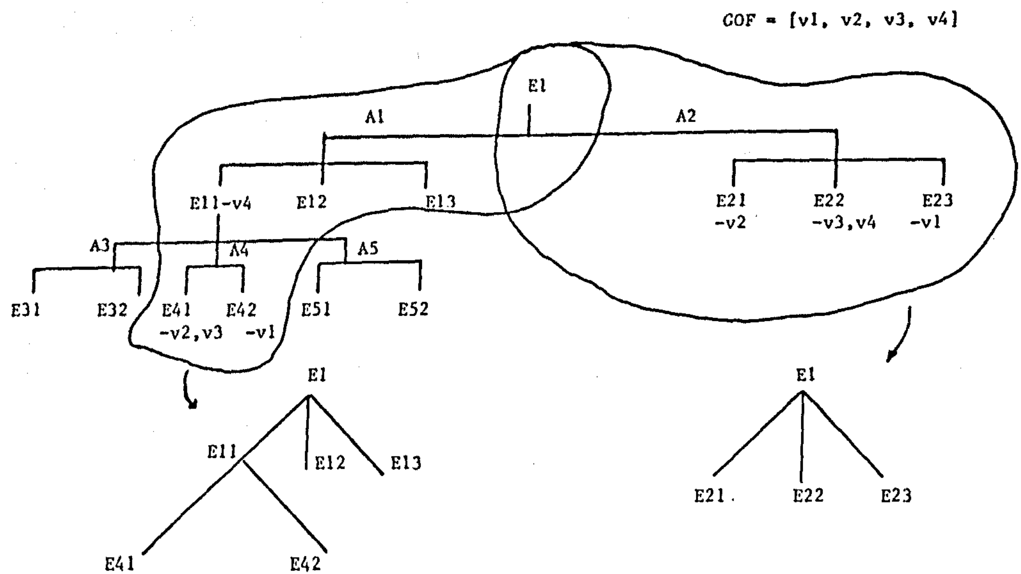

Figure 3. Pruning the System Entity Structure in the Generic Frame GOF and Resulting Structures for Model Construction

In conclusion, the pruning process plays the major role in the selection of model alternatives that conform to the objectives. Thus the modelling space is meaningfully restricted to behaviorally feasible model structures. Having presented the framework for behavioral pruning, in the next chapter we proceed to establish rules for expressing the structural aspects of modelling objectives.

6. ENTITY STRUCTURE-BASED SYNTHESIS RULE SPECIFICATION

The pruning process described in the foregoing chapter restricts the space of possibilities for selection of components and couplings that can be used to realize a model. Thus we can assume that model development process may now be reduced to the synthesis problem (Zeigler, 1986). Synthesis involves putting together a system from a known and fixed set of components in a fairly well-prescribed manner. In the synthesis problem, we are modelling a rather restricted design process, one amenable to automation by extracting concepts and procedures from experts' knowledge and experience, augmenting them and molding them into a coherent set of rules. The rule development methodology that we propose for such a modelling enterprise is as follows:

*.) Restrict the modelling domain by pruning the system entity structure in respective generic observation frames.

*.) Examine the the resulting substructure and their constraints. Try to convert as many constraint relations as possible into the active from, i.e. into rules that can satisfy them. For those that cannot be converted into such rules write rules that will test them for satisfaction.

*.) Write additional rules, modify existing ones, to coordinate the actions of the rules (done in conjunction with the selected conflict resolution strategy).

In a synthesis problem, several kinds of constraints may come into play. Here, we focus on two types of constraints that influence the manner in which synthesis rules are specified. Assuming that a synthesis problem is appropriate for expert system design, there are known actions that can be taken to try to satisfy the performance constraints derived from the objectives and imposed standards. Indeed, an expert's procedural knowledge represents efficient procedures that are likely to achieve the goals and subgoals that arise in attempting to meet the performance requirements. The pruning process described in the previous section is an example of such an action. We see the following constraint classification emerging: some constraints are convertible to active form, i.e., they can be converted into actions intended to satisfy them. Other constraints are inherently passive, they do not motivate or guide action, they sit there demanding satisfaction. The question that now begs to be addressed is: assuming that it is possible, how can we convert a constraint to active form? We conceive of the synthesis problem as a search through the search space, the set of all pruned model structures. These are candidates for a solution to the problem. Our set of rules will take us from an initial state in this space to a goal state. The search should proceed by generating successive candidate structures in an efficient manner.

We can assume that for each active constraint we have a means of generating such candidates to test against the constraint. Call such an operator NEXT_IN_Ci.

The passive constraints have no corresponding operators and thus we can only test for their satisfaction. Failure causes backtracking if a state has been reached for which none of the operators can be applied. Instead of applying an operator and then testing if it has consumed more than what remains of an available resource, we can try to inhibit the application of operators that would bring about the resource depletion.

Let Con be a constraint that we wish to pretest. An operator, NEXT_IN_Ci will map a state s into the region satisfying Con if, and only if, Con(NEXT_IN_Ci(s)). To allow the operator to be applied safely we need to define applicability predicate, Ai such that:

Ai(s) if, and only if, Con(NEXT_IN_Ci(s))

Thus a canonical rule scheme for a synthesis problem takes the following form:

```
RC  If C satisfied on (state)
       then Output (state) as the solution

R1  If C1 is not satisfied
       A1 is satisfied
       then state:=NEXT_IN_C1(state)

 .......

Ri  If Ci is not satisfied
       Ai is satisfied
       then state:=NEXT_IN_Ci(state)

 .......

Rn  If Cn is not satisfied
       An is satisfied
       then state:=NEXT_IN_Ci(state)
```

The structures generated as results of behavioral pruning and structural synthesis should be used to construct models employing the hierarchical model construction methodology . We shall briefly describe the underlying concept of this framework and refer the reader for details to (Zeigler, 1984).

Recall that the pruned entity structures generated by the objectives-driven pruning represent minimal structures that have all the variables required by the generic observation frame. Many more variables may have to be employed by a model to fully express the nature of the system being modelled. Thus, we must assume that an expansion of the set of attached variables is possible to incorporate all the attributes requires by the model.

The next step in the model construction process is the so-called orientation and role designation i.e., selection of input, output, state variables and model parameters. Having established input/output orientation, we are in a position to couple components together in accordance with the coupling constraint associated with internal nodes of the structure. The formalism that enables us to uniquely specify models based in the hierarchies pruned from the system entity structure is called <u>composition tree</u> (Zeigler, 1984a).

Let us now gather the strands up and propose an environment to support the model and experimental frame development process. An example explicating the use of such an environment and its formal tools will follow in Section 8.

7. <u>ENVIRONMENT FOR SUPPORT OF MODEL AND FRAME DEVELOPMENT</u>

It has been our contention throughout the foregoing sections that the system entity structure and the concept of the generic frame type constitute the knowledge that can support the automatic specification of models and experimental frames. To discuss this argument we propose the following architecture to support such an automatic process.

As illustrated in Figure 4, the data base of simulation objectives specification is one of the major components of the system. It has to be well understood that the modelling objectives drive three processes in our methodology. First, the retrieval and/or construction of the system entity structure. Naturally, the modeller desires to obtain a family of model representations rather then a single model structure. A classic example would be the area of system design where a spectrum of design alternatives is sought for evaluation before the final design is chosen (Rozenblit, 1984b). Secondly, the objectives serve as a basis for definition of the generic frame types. Finally, the objectives understood in a somewhat broader context (e.g. as design requirements), imply a set of rules for the model synthesis and constraints on how the model components may be coupled. Therefore in the proposed architecture we introduce the base of synthesis rules and coupling constraints.

The ultimate purpose of the system represented in Figure 4, is to analyze and integrate the relationships concerning the objectives specification base, the generic frame, and system entity structure base to form an appropriate model and simulation experiment for the problem at hand. As Shannon, Mayer and Adelsberger (1985) point out, this presents an ideal problem for the application of expert systems technology.

Let us propose how such a system should operate given the knowledge represented by the aforementioned bases. First, we augment the system entity structure extraction with a synthesis rule-based pruning. The pruning procedure presented in Section 5 extracts the substructures that accommodate the simulation objectives from the behavioral standpoint. Actually, the nature of the generic frame concepts is intrinsically behavioral. Pruning the entity structure in a generic template results in models whose behavioral properties enable us to answer the questions of the simulation study. We feel however, that the class of models generated by pruning should be further restricted in order to account for the constraints imposed by the rules of synthesis (Rozenblit and Zeigler, 1985).

Both, structural and behavioral pruning applied to the system entity structure should result in model structures that we term candidates for hierarchical model construction. The term candidates implies that some checks for consistency and admissibility (in the sense of conformance to the objectives) should be performed at this stage. If the candidate is inadmissible or no candidates can be obtained by pruning, the process should be reiterated with possible user intervention. The

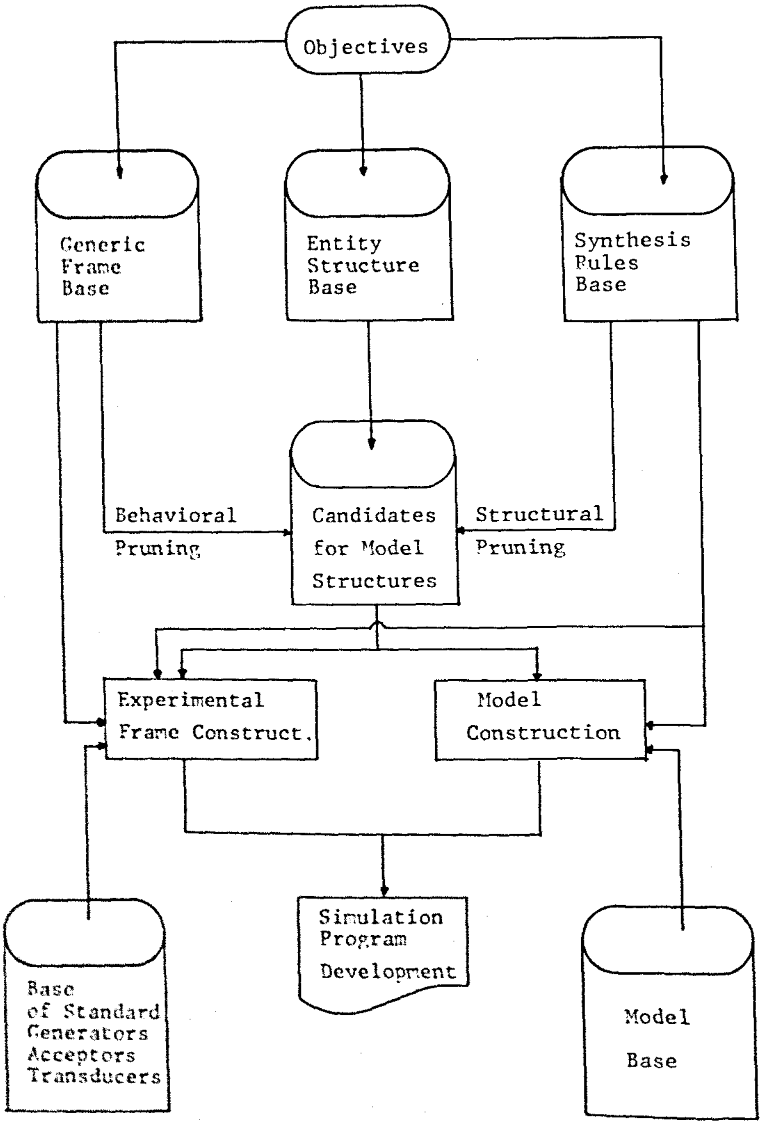

Figure 4. Environment for Support of Model and Frame Development

kinds of interventions we suggest are modifications or retrieval of the new system entity structure, enhancement of the generic experimental frame or modification of synthesis rules.

The system should construct models for the skeletons generated as a result of the structural and behavioral pruning. Such model construction is based on the composition tree formalism presented in (Zeigler, 1984). Again, the base of synthesis rules should be consulted for proper implementation of coupling constraints. At the same time the generic experimental template should be instantiated. We envision the instantiation as a three phase process.

In the first stage the variable types present in the template are assigned component names i.e. the names of entities to which these types are attached. However, since the pruning procedure proposed in Section 5., generates all occurrences of a given type, it is necessary to eliminate from the experimental frame those variables which are internally controlled in the model. Recall, that the pruning is guided only by the input and output generic types, therefore we have to be concerned with that type of variables. An appropriate scheme for elimination is to consult the synthesis rules and coupling constraints associated with the entities of the model candidate structures and proceed to filter out the internally controlled variable by applying a scheme similar to that of the <u>coupling recipe</u> defined by Wymore (1980).

Following Wymore's terminology we assert that the experimental frame induced by a generic frame type contains only those (input and output) variables that are free input and output variables. An input/output variable is free if it does not appear in any of the links of the coupling scheme. As this may be somewhat restrictive in the sense of limiting the observation space we can relax this constraint by allowing the frame to collect the data from some of the internally controlled output variables.

The third stage in the frame generation is to choose appropriate variables to serve as run control variables. We feel that this should be done by the user just before the model is ready to be run within the frame under consideration. The same concerns setting up the INITIAL and TERMINAL sets. The summary variables of the generic frame are directly applied in the experimental frame. They are simply instantiated with the names of the model components to which the modeller wishes them to apply.

Finally, in the context of the frame realization, a base of standard generators, acceptors and transducers should be available in the system. The retrieval of the appropriate modules from that base would be guided by the obtained experimental frame definition.

To illustrate the concepts discussed in the foregoing sections we now provide a simple example form the area of automotive design.

8. <u>EXAMPLE - DEVELOPMENT OF DESIGN MODELS</u>

Assume that an automotive company is designing a new model of a truck. To satisfy prospective customers who, among other things, require that a truck should be operational above 95% over its life cycle, the company has placed a very strong emphasis on the reliability aspect of the new model. Factors like: the number of scheduled inspections, the time it takes to complete an inspection, the time it takes to repair or replace a malfunctioning part etc., will play a major role in evaluating the new design. In general, the generic frame Utilization can be chosen to represent this particular behavioral design objective. A corresponding observation frame is given below:

<u>Generic</u> <u>Observation</u> <u>Frame</u>: Utilization.

Input variables:
   Scheduled.Service
   Breakdown

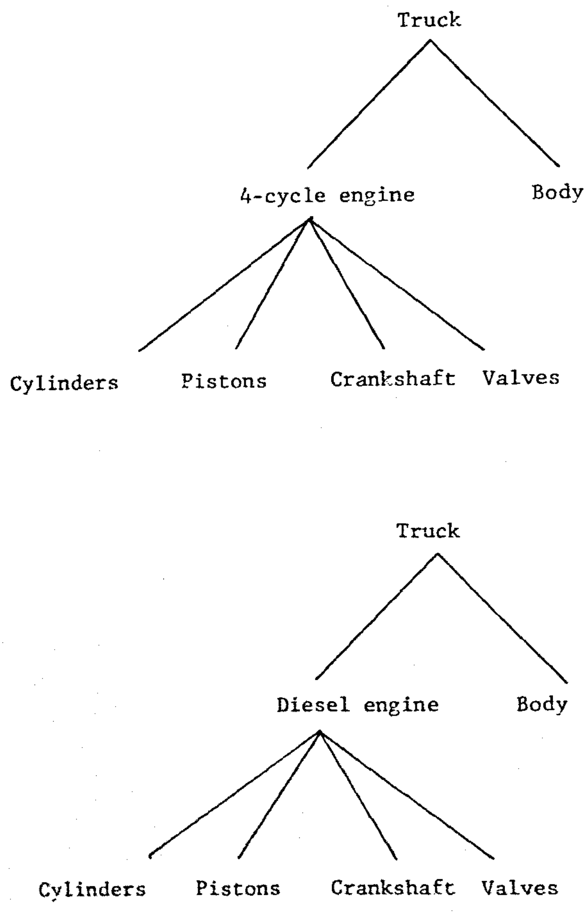

Figure 5. Model Structures Resulting from Pruning the Sytem Entity Structure of Figure 1.

Output Variables:
Status (with range {In.Repair, In.Operation})

We should also list some other requirements and constraints concerning the truck design. For example, the company may be restricted by the technological standards to use only internal combustion, four cycle or diesel engines.

In the first stage of the design process, a system entity structure representing an automobile is proposed. Such an entity structure can have the form depicted in Figure 1. As we can see there are several aspects and specializations in the structure. Certainly, for the sake of brevity, our automotive design is rather simple.

Given the design entity structure we first prune it with respect to the observation frame Utilization. The safety aspect will be pruned out as it does does not have the generic variable types present in our observation frame. Pruning results in a total of six design model structures with Passenger.Car and Truck as the root entities and Body, 4-cycle/diesel/electrical engine, as the entities representing the components of decomposition.

As the design objectives and constraints dictate, the structures for a passenger car are eliminated first. Further, the structure for a truck with an electrical engine is disregarded. Finally, only the configurations with a 4-cycle and diesel engines are deemed admissible. They are shown in Figure 5.

The general truck design problem is now reduced to the synthesis of a truck with a 4-cycle, internal combustion or diesel engine. We are now ready to formulate the structural constraints and convert them into a production rule scheme consistent with the canonical form presented in Section 6.

In our formulation we shall define synthesis rules for a very coarse model of a truck. We shall assume that the following factors play a major role in the synthesis process: first, we should restrict the maximum capacity of the truck. Secondly, we assert that it is necessary to synthesize an engine with enough power to set the truck (with maximum load) in motion. We assume that in order to increase the engine's power we can add cylinders in pairs. However, the number of cylinders cannot be less than 4 and cannot exceed 16. Adding a pair of cylinders also increases the volume of the engine i.e.:

ENGINE.VOLUME = VOLUME.FACTOR * CYLINDER.VOLUME * CYLINDERS.NUMBER

The constraints associated with the physical decomposition of the entity TRUCK can be formulated as follows:

1.) BODY.VOLUME >= ENGINE.VOLUME + LOAD.VOLUME

2.) ENGINE.POWER >= BODY.WEIGHT + MAXIMUM.LOAD

3.) BODY.VOLUME <= MAXIMUM.VOLUME (capacity)

where MAXIMUM.VOLUME can be interpreted as a standard constraint imposed by government regulations (due to restrictions on maximum axis load on interstate highways) and the measures of load and weight are given by the relations below:

LOAD = LOAD.VOLUME * WEIGHT.OF.VOLUME.UNIT

BODY.WEIGHT = BODY.VOLUME * WEIGHT.OF.VOLUME.UNIT

The constraints associated with the ENGINE synthesis have the form:

4.) CYLINDERS must be coupled in pairs
either in line or across from one another

5.) CYLINDERS.NUMBER $\in$ [4, 16]

To convert the constraint to production rules we implement the canonical scheme given in Section 6. As in the general approach rule RC is the global constraint checker. Rules RC1 and RC2 are implemented as local constraint satisfiers for constraints 1 and 2. Note, that the resource constraints 3 and 5 have been formulated as pretests for applicability of the rules. The production rule scheme is presented below:

```
RC  if ENGINE.POWER >= BODY.WEIGHT + MAXIMUM.LOAD
       BODY.VOLUME >= ENGINE.VOLUME + LOAD.VOLUME

    then
       Print "Truck Completed"

RC1 if BODY.VOLUME <= MAXIMUM.VOLUME - 1 VOLUME.UNIT
       BODY.VOLUME < ENGINE.VOLUME + LOAD.VOLUME

    THEN
       expand BODY.VOLUME by 1 UNIT
       update BODY.WEIGHT

RC2 if a pair of CYLINDERS is available
       ENGINE.POWER < BODY.WEIGHT + MAXIMUM.LOAD
    then
      add this pair of CYLINDERS to the ENGINE
      update ENGINE.VOLUME
```

After candidate structures that satisfy all the constraints have been found, design models of the truck should be constructed and the observation frame "Utilization" should be refined to an experimental frame. Then, the resulting models can be evaluated via simulation experiments.

9. CONCLUDING REMARKS

We have presented an approach to generate the model and experimental modules for simulations in the objectives-driven modelling environments. As we have shown, the model and experimental frame development should be mutually supportive in the following sense: while the basic objects representing the knowledge about the model and experimental frame (i.e. the system entity structure and generic frame type) can be conceived and developed separately, the construction of the model and experiment specification for a given problem is a process in which inferences from both objects should be drawn at the same time.

We hope that our approach will contribute to the ongoing discussion concerning the interfaces between modelling methodologies and expert system techniques, and that our subsequent research efforts will result in the fruition of the presented concepts in the form of expert simulation software support.

REFERENCES

Belogus D. (1985) "Multifacetted Modelling and Simulation: A Software Engineering Implementation", Doct. Diss., Weitzman Institute of Science, Rehovot, Israel.

Crosbie R.E., Hay J.L. (1983) "ISIM - A Simulation Language for CP/M Systems", in: Proc. of the Summer Computer Science Conference, Vancouver, July 1983.

Dekker L., Elzas M.S., (1984) "Methodology-Based Interactive Systems Modelling and Implementation", in Proc. of the 1984 Summer Simulation Conference, Boston, July 1984.

Elzas M.S., (1982) "The Use of Structured Design Methodology to Improve Realism in National Economic Planning", in Model Adequacy (ed. H. Wedde), Pergamon Press, London.

Elzas, M.S., (1984) "System Paradigms as Reality Mappings", in Simulation and Model-Based Methodologies: An Integrative View, (ed Oren, T.I., et al), Springer-Verlag, New York.

Kettenis D., (1983) "The COSMOS Modelling and Simulation Language", in Proc. of The First European Simulation Congress, Sept. 1983

Oren, T.I., Zeigler B.P. (1979) "Concepts for Advanced Simulation Systems", Simulation, 32,3 pp. 69-82.

Oren, T.I. (1982) "GEST - A Modelling and Simulation Language Based on Systems Theory Concepts", in: Simulation and Model Based Methodologies: An Integrative View, (ed. Oren T.I., et.al), Springer-Verlag .

Pegden D., (1982) "Introduction to Siman", Systems Modeling Corp., State College, Pensylvania.

Reddy Y.V., Fox M.S., Husain N., (1985) "Automating the Analysis of Simulations in KBS", in Proc. of the SCS Multiconference, San Diego, January 1985.

Rozenblit J.W., (1985a) "A Conceptual Basis for Integrated, Model Based System Design", Doct. Diss. Dept. of Computer Sci., Wayne State University, Detroit, Mi

Rozenblit J.W., (1985b) "Experimental Frames for Distributed Simulation Architectures", Proc. of the 1985 SCS Multiconference, San Diego, January 1985.

Rozenblit J.W., (1984) "Structures for a Model-Based System Design Environment", Technical Report, Siemens AG, West Germany (internal distribution).

Rozenblit J.W. and Zeigler B.P., (1985) "Concepts for Knowledge-Based System Design Environments", Proc. of the 1985 Winter Simulation Conference, San Francisco, December 1985.

Shannon R.E., Mayer R., Adelsberger H.H., (1985) "Expert Systems and Simulation", Simulation vol. 44, number 6, June 1985.

Sprague R.H., Carlson E.D. (1982) "Building Effective Decision Support Systems", Prentice Hall, N.J.

Wymore, W. (1980) "A Mathematical Theory of Systems Design", Tech. Report, College of Engineering, U. of Arizona.

Zeigler B.P. (1984a) "Structures for Model Based Simulation Systems", in: Simulation and Model-Based Methodology: An Integrative View, Springer-Verlag

Zeigler B.P. (1984) "Multifacetted Modelling and Discrete Event Simulation", Academic Press, London.

Zeigler B.P., (1986) "Expert Systems: A Modelling Framework", (in preparation)