Proceedings of the 1988 Winter Simulation Conference
M. Abrams, P. Haigh, and J. Comfort (eds.)

Systems theory instrumented simulation modeling

Jerzy W. Rozenblit
Dept. of Electrical and Computer Engineering
University of Arizona
Tucson, Arizona 85721

Abstract

The paper reviews the impact of systems theory–based representation methods in the context of discrete event simulation modeling. Tools for formal system specification and methods for model static and dynamic structure description are presented. Recent efforts in implementing the systems theory instrumented multifacetted methodology are also discussed.

1. Introduction

In the past decade methodological research in the area of simulation and modeling has significantly changed and influenced the way in which we carry simulation studies today. The power of conventional simulation techniques has been extended in directions that enable us to structure data better, help in building hierarchical models, and assist us in designing simulation experiments. Two related trends have significantly contributed to this progress in the simulation field. One was the development of conceptual theories for modeling practice and for designing software tools for supporting such practice (Zeigler et. al. 1980). The other was the emergence of simulation languages based on theoretical frameworks (Ören 1971; Pegden 1982; Livny 1987; Zeigler 1987). Clearly, the latter trend demonstrates the functional and expressive power of theory–based modeling methods.

Systems Theory is a scientific discipline whose primary concern is to provide problem solving methods and tools. Although Systems Theory has been a subject of intensive studies for over two decades and an abundant literature on the subject is available, its problems solving methods have been often ill–understood and ignored by engineers, system designers, and simulation practitioners. Pichler (1988) points out the major reasons for this problem and suggests that with the advent of powerful engineering workstations, systems theory will eventually find its way to applied sciences and engineering.

There are notable exceptions to the skepticism about the applicability and usefulness of system–theoretic methods. They include work of Zeigler (1984a, 1987) whose multifacetted modeling we shall explore here in more detail. Given the space limitation, it is impossible to examine closely the work of others in this paper. However, to provide the reader with points of reference, we briefly describe research efforts bridging systems theory, system design, and simulation modeling.

First attempts to unify these disciplines date back to the early seventies and work of Ören (1971). He designed a general systems theory implementor language (GEST) that facilitated expressing both continuous and discrete simulation formalisms. A more advanced version named GEST81 and its high level development shell MAGEST (Aytac and Ören 1986) provide a means for specifying hierarchically coupled models.

Wymore's work (1967, 1976, 1980) is primarily concerned with applying mathematical systems theory to systems engineering problems. The tricotyledon theory of system design (T3SD) has been applied to a number of engineering design problems. In T3SD, a designer specifies in mathematically rigorous terms an I/O specification of the designed system and a merit ordering over the set of I/O specifications. Similarly, a class of buildable systems and a merit ordering is defined with respect to available technology. A feasible design is selected from the systems that are buildable in the technology and satisfy the I/O specification.

Klir has developed a general systems problem solver (GSPS) for inductive modeling (1984). The system is based on the hierarchy of systems descriptions called epistemological levels related to Zeigler's taxonomy of system specification levels (1976).

Fishwick (1988a, 1988b) has categorized process abstraction in simulation modeling and presented formalisms for valid methods of abstraction. Recently, he has developed procedures for a reverse process, i.e., refinement, used to automate the transition from abstract lumped models to base models.

A new development in realizing systems concepts has been initiated by Pichler (1988). This new research, termed Computer Aided Systems Theory (CAST) and conducted at the University of Linz in Austria, is aimed at providing method banks for computer aided problem solving. CAST is derived from Pichler's work on Systems Theory Instrumented Problem Solver (STIPS). STIPS has been realized in an object-oriented environment for finite state machines (Pichler 1986).

To illustrate the power and viability of a theory–based approach, in this paper we shall present the basic concepts of a modeling framework termed multifacetted methodology (Zeigler 1984a). We then illustrate how the theoretically motivated concepts have been realized in an advanced simulation and system design environment. The software environment is discussed in detail in a companion paper in this volume (Rozenblit et.al. 1988).

2. System Theoretic Concepts for Model Representation

In this section we present formal concepts that underlie the specification of simulation models. We provide a formal definition of a system, discuss the distinction between the system's structure and behavior and present methods for the specification of model static and dynamic properties.

Hierarchy of System Specifications

The term model is viewed here as a specification for a system. In general, the term system refers to a description (often mathematical) which captures some of the essential features concerning the system or problem being modelled. Since there are many characteristics of real systems, there are several concepts of the system and thus it is useful to organize the specifications into a coherent whole. In this way we arrive at a stratification of system specifications that starts with intuitive black box concepts at the lover levels and adds more and more constructs for the description of system's internal structure as the levels increase. Klir's epistemological classification of systems description of one such example (Klir 1984).

In this paper, we shall adopt a notion that a system is a collection of interacting component systems (Zeigler 1984b). This recursive definition implies that we should have a means for expressing a decomposition process. In this process, a system can be represented by a collection of subsystems whose composition exhibits the same properties as the system subject to the decomposition. This implies that a system may be a component of a larger system.

Zeigler provides a hierarchy of system specifications with morphism concepts that enable comparisons between systems specified at any level of abstraction. The hierarchy is defined as follows:

1. System Specification IORO. This type of description is an Input/Output observation relation. It is a classical example of a black box.
2. System Specification IOFO. For each input function of a given IORO there exists exactly one output function. This specification is called an I/O function observation.
3. System Specification S (often referred to as an I/O system). In addition to input and output sets, a set of states and state-transformation mechanisms have to be defined in this specification. We shall shortly discuss this specification in more detail and show how it is used as a basis from which a model description can be derived.
4. Structured System Specification. The specification at this level is the same as the one at level 3 except that each set and function is structured.
5. System Specification NET (Coupled System). NET denotes a network of system specifications consisting of a family of systems and a coupling mechanism. This specification is a basis for a hierarchical form of model construction.

The above stratification of system specifications is independent of any particular modeling formalism. In other words, any formalism may be employed to specify a system at any level. A good review of major modeling formalisms (differential equations, discrete time systems, and discrete event specification) and their associated translation mappings into system specifications are given in (Zeigler 1984b).

At each level in the above hierarchy there is a morphism that enables comparisons between systems specified at that level. Transitions between levels of system specifications and their associated morphisms are straightforward when we want to derive a specification assuming that one is given at a higher level (we ascertain behavior from the structure). The opposite transformation, i.e., recovering a structure from a given behavior is not possible unless certain requirements called justifying conditions are satisfied. Both types of transitions are well described in (Zeigler 1976, 1984a; Pichler 1984).

Formal System Definition

We provide a formal system definition (System Specification S—Level 3) as a basis for describing the model's structure and behavior. Subsequently, we present a modeling formalism called Discrete Event System Specification (DEVS) and show how it specifies an I/O system.

A system is a structure:

$$S =< T, X, \Omega, Q, Y, \delta, \lambda >$$

where:

T is the time base
X is the input value set
Ω is the input segment set
Q is the internal state set
Y is the output set
$\delta: Q \times \Omega \rightarrow Q$ is the state transition function
$\lambda: Q \rightarrow Y$ is the output function

The input segments of the system S have to be closed with respect to concatenation. In addition, δ must have the semi-group property, i.e., for all $\omega, \omega' \in \Omega$ and for all $q \in Q$ the following equation must hold: $\delta(q, \omega \cdot \omega') = \delta(\delta(q, \omega), \omega')$

The input, state, output sets, and the output function constitute the static structure of the system. The time base, input segment set, and the state transition function are referred to as the system's dynamic structure. In Section 3, we shall demonstrate how the formal concepts of the multifacetted methodology support the specification of both structures.

In systems theory instrumented simulation, models are expressed in special formalisms depending on the problem at hand. Typical specifications include differential equations, finite state machine, or discrete event. Each formal model description specifies a system and selects a class of subsystems by placing constraints on the possible static and dynamic structures it encompasses. A characterization of such constraints is given in (Zeigler 1984b). The model construction process involves specification of the static and the dynamic structure. We shall explain this process in the ensuing section.

3. Model Specification in Multifacetted Methodology and Discrete Event Simulation

The concepts of model development presented here are derived from multifacetted modeling methodology. Multifacetted methodology denotes a modeling approach which recognizes the existence of multiplicities of objectives and models in a simulation project. It provides formal representation schemes that support the modeller in organizing the model construction process, aggregating partial models, and in specifying simulation experiments (Zeigler 1984a; Rozenblit and Zeigler 1986, 1988).

System Entity Structure

The key concept underlying structuring of models, their organization, and specification of simulation experiments (experimental frames) is the *system entity structure* (Zeigler,

1984a). The system entity structure is based on a tree-like graph that encompasses the boundaries, decompositions and taxonomic relationships that have been perceived for the system being modelled. 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 are thought of as components and decompositions, respectively. In addition to decompositions, there are relations termed specializations. A specialization relation facilitates representation of variants for an entity. Called specialized entities, such variants inherit properties of an entity to which they are related by the specialization relation.

Entities have attributes represented by the attached variable types. When a variable type V is attached to an entity E, this signifies that a variable I.E may be used to describe a property of the entity E. Aspects can have coupling constraints attached to them. Coupling constraints restrict the way in which components (represented by entities) identified in decompositions (represented by aspects) can be joined together.

In addition to coupling constraints, there are selection constraints in the system entity structure. Selection constraints are associated with specializations of an entity. They restrict the way in which its subentities may replace it in the process of model construction (Rozenblit and Huang 1987).

Given the system entity structure the modeller has a choice of a number of model alternatives. This is due to the multiplicity of aspects and specializations. Therefore, we require that the modeller have procedures for generating model structures pertaining to the modeling objectives. Such structures should be selected from the system entity structure.

In our previous work (Rozenblit and Huang 1987) we have developed procedures for generating structures underlying the coupled model specification. The process, called *constraints–driven pruning*, employs the production rule formalism to support automatic selection of entities from taxonomic relationships and synthesis of structures underlying simulation models.

This approach to pruning consists in specifying the system entity structure for a modeling problem. Then, a knowledge base that contains rules for selection and configuration of the entities is constructed. The modeller invokes an inference engine which, through a series of queries based on the constraint rules, allows him/her to consult for an appropriate structure for the modeling problem at hand. The result is a recommendation for a model composition tree (Zeigler 1984a) which is a basis for the hierarchical model development. The model composition tree is a tree whose leaf nodes are system specifications. These are the atomic components which will be coupled in a hierarchical manner. The interior nodes n have the following specification attached to them: a system specification S_n, a coupling scheme C_n, and a morphism H_n. The coupling scheme C_n is used to interface the system specifications assigned to the children of the interior node. H_n establishes a correspondence between S_n and the resultant of the coupling process using C_n. The leaf nodes are assigned only system specifications which are atomic and are not subject to decomposition.

DEVS Formalism

The modeling formalism used to specify a system in the multifacetted methodology is Discrete Even System Specification (DEVS) (Zeigler 1976, 1984a, 1984b). DEVS provides a formal representation of discrete event systems. Formally, DEVS is a structure:

$$M =< X, S, Y, \delta, \lambda, ta >$$

where:

X	is the external event set
S	is the sequential state set
Y	is the output set
δ	is the transition function
λ	is the output function
ta	is the time advance function

DEVS specifies an I/O system:

$$S =< T, X, \Omega, Q, Y, \delta, \lambda >$$

where:

$T = Reals$
$X = X_{DEVS} \cup \{\phi\}$ (an empty event)
Ω = set of discrete event segment over X

The state set is defined as follows:

$$Q = \{(s, e) \mid s \in S, 0 \leq e \leq ta(s)\}$$

where:

$$ta: S \rightarrow R_{0,\infty}$$

and (s, e) is a total state pair, where s is a sequential state and e is elapsed time in state s.
The transition function consists of two pairs, namely:

$\delta_\phi: S \rightarrow S$ is the internal transition function

and

$\delta_{ex}: Q \times X \rightarrow S$ is the external transition function

The formal construction of the system's transition function δ is given in (Zeigler 1976). DEVS is closed under coupling. This property enables us to construct hierarchical DEVS network specifications. A detailed formal treatment of DEVS at the coupled system level is presented in (Zeigler 1984a).

The formal model specification in multifacetted methodology consists in specifying the system entity structure, attached variable types (called descriptive variables), pruning and then specifying a discrete event model for the components identified by the pruned entity structure. Selection of input, output, and state variables results in the model's static structure. Definition of transition and output functions add the dynamic components to the DEVS specification. A full scale demonstration of this process is presented in (Zeigler 1984b).

Clearly, a formal set theoretical description of a large scale system would be a tedious and impractical process. In fact, this may well have been a reason why theory-based approaches have been shunned by simulation practitioners, and a primary motivation for the development of software implementing the above formal modeling concepts. In what follows, we shall briefly describe a simulation environment realizing the DEVS specification and the system entity structure.

4. Implementing Theoretical Concepts

The Entity Structuring Program (ESP) was first developed by Zeigler, Belogus and Bolshoi (1980) and subsequently modified to include graphic facilities. The system entity structure specification has been incorporated in the DEVS-Scheme environment (to be discussed shortly) under the name ESP-Scheme.

The program helps the modeller conceptualize and record the decompositions underlying a model, or family of models, before, during, and after development. To the extent that ESP- Scheme is used before beginning model development, it is a tool for assisting in top down model design. However, when additions and changes are made as the development proceeds, ESP serves as a recorder of progress. At the end of the development phase, the record constitutes de facto documentation of the system structure arrived at. Pruned entity structures serve as a basis for the retrieval from a model base of model components specified in DEVS-Scheme.

The DEVS formalism has been implemented in TI–Scheme, a LISP–based programming environment. The shell, called DEVS-Scheme, supports modular, hierarchical specification of discrete event models (Zeigler 1987). Component models, called atomic models are specified using Scheme's semantics that corresponds closely to the formal definition of DEVS. The elements of DEVS formalism take the following form in the shell: The input and output sets consist of pairs (port, value). Thus, x = (p,v) signals the receipt of a value v at an input port p. The functions take the following forms:

```
internal transition function (define (int s) ...)
external transition function (define (ext s e x)...)
output function (define (out s) ...)
advance function (define (ta s)...)
```

where "..." represent function body definitions expressed in Scheme. The atomic models may be coupled together to form a model at the coupled specification level by using the composition tree concept introduced in Section 3. DEVS-Scheme is still under development. Recently, new features for testing model morphisms and model simplification mappings have been incorporated in the shell (Kim 1988, Sevinc 1988). We refer the reader to (Rozenblit et. al. 1988) and (Zeigler 1987) for a full description of DEVS-Scheme.

Other implementations of DEVS specification include the work of Melman and Livny (1984) and Praehofer and Spalt (1988). Livny has been developing tools for distributed processing systems using DEVS as an underlying model description mechanism. His first realization of DEVS resulted in a SIMSCRIPTII.5–based software system called Distributed System Simulator (DISS). Recently, he has implemented DEVS concepts in Modula–2. The simulator called DeNET has been successfully applied in simulation of large scale distributed systems (Livny 1987)

Praehofer and Spalt have implemented DEVS in an object-oriented environment INTERLISP/LOOPS as part of the CAST project mentioned in Section 1. Their implementation is based on the original concept of DEVS-Scheme.

5. Summary

We have summarized a methodological development in simulation modeling that bases itself on systems theory concepts. We have focused on multifacetted modeling. The work

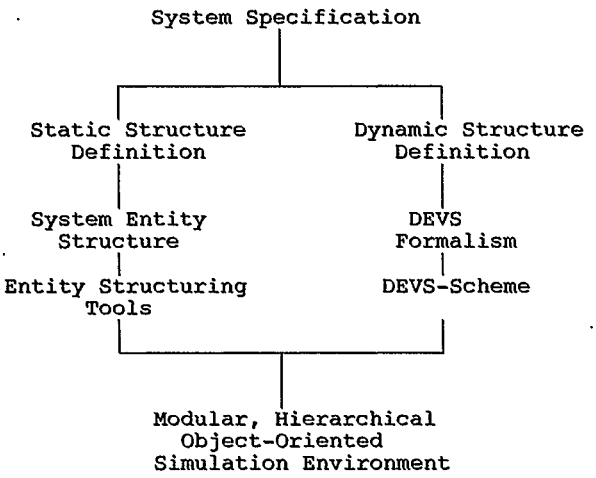

Figure 1. Basic Concepts and Tools in Multifacetted Methodology

of others employing the theory–based concepts has been briefly outlined and referenced. Figure 1 retraces the major concepts of this paper. The system specification hierarchy is a basis for the model description at different levels of abstraction. The model development process requires that both model's static and dynamic structures be defined. We have presented tools for deriving such specifications, namely, the system entity structure and the DEVS formalism. We have also described the current efforts in implementing the tools in software environments.

We feel that the latest developments in implementing systems theory concepts will ultimately result in better simulation management. This will be due to the abilities provided by the state-of-the-art software to employ systematic methods for the model development, automatic construction of models at the coupled system level, reusability of atomic model components, and automatic management of model bases.

References

Aytac, Z. K. and T. I. Ören (1986) MAGEST: A Model Based Advisor and Certifier for GEST Programs, in *Modelling and Simulation Methodology in the Artificial Intelligence Era*, (eds. Elzas, M. et. al.), North Holland, Amsterdam, pp. 299-307

Fishwick, P. A., (1988a) The Role of Process Abstraction in Simulation. *IEEE Transactions on Systems, Man and Cybernetics*, 18(1), January.

Fishwick, P. A., (1988b) Automating the Transition from Lumped Models to Base Models. *Proc. of the*

1988 Eastern Simulation Conference, Orlando, Florida, April. pp. 57-63.

Kim, T. G., (1988) A Knowledge–Based Environment for Hierarchical Modelling and Simulation, Doctoral Dissertation, University of Arizona, Tucson.

Klir, G. J., (1984) General Systems Framework for Inductive Modelling, in: *Simulation and Model-Based Methodologies: an Integrative View* (eds. Oren, T.I. et. al.) Springer-Verlag, New York, pp. 69-90

Livny, M. (1987) DeLab–A Simulation Laboratory, *Proc. of the 1987 Winter Simulation Conference*, Atlanta, December, pp. 486-494

Melman, M. and M. Livny (1984) The DISS Methodology of Distributed System Simulation, *Simulation Journal*, April, pp. 163-176

Pichler, F. (1984) Symbolic Manipulation of Systems Models, in: *Simulation and Model-Based Methodologies: an Integrative View* (eds. Oren, T.I. et. al.) Springer-Verlag, New York, pp. 217-234.

Pichler, F. (1986) Model Components for Symbolic Processing by Knowledge-Based Systems: The STIPS Framework, in: *Modelling and Simulation Methodology in the Artificial Intelligence Era* (eds. Elzas, M. et. al.), North-Holland, Amsterdam pp. 133-142

Pichler, F. (1988) CAST–Computer Aided Sytems Theory: A Framework for Interactive Method Banks, in: *Cybernetics and Systems '88*, (ed. Trapl, R.) Kluwer Academic Publishers, pp. 731- 736

Praehofer, H. and A. Spalt (1988) An Interactive Simulation Environment Based on Systems Theory Concepts and Object Oriented Programming", in: *Cybernetics and Systems '88*, (ed. Trapl, R.) Kluwer Academic Publishers.

Ören, T. I., (1971) General Systems Theory Implementor. Doctoral Dissertation, University of Arizona, Tucson.

Rozenblit, J. W. and Zeigler, B. P., (1988) Design and Modelling Concepts, in: *International Encyclopedia of Robotics*, (ed. Dorf, R.) John Wiley and Sons, New York.

Rozenblit, J. W. and Zeigler, B. P., (1986) Entity Based Structures for Experimental Frame and Model Construction, in: *Modelling and Simulation in the Artificial Intelligence Era*, (ed. M. S. Elzas, et. al.) North Holland, Amsterdam, pp. 79-100.

Rozenblit, J. W. and Y. M. Huang (1987) Constraint-Driven Generation of Model Structures. *Proc. of the 1987 Winter Simulation Conference*, Atlanta, December, pp. 604-611

Rozenblit, J. W., Kim, T. G. and B. P. Zeigler (1988) Towards the Implementation of a Knowledge–Based System Design and Simulation Environment. *Proc. of the 1988 Winter Simulation Conference*, San Diego. December.

Sevinc, S. (1988) Automatic Simplification of Simulation Models in a Hierarchical, Modular Simulation Environment, Doct. Dissertation, University of Arizona, Tucson.

Wymore, W. A., (1980) A Mathematical Theory of Systems Design. Technical Report, University of Arizona, Tucson, Arizona.

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

Wymore, W. A., (1976) *Systems Engineering Methodology for Interdisciplinary Teams*, John Wiley and Sons, New York.

Zeigler, B. P. (1976) *Theory of Modelling and Simulation*, John Wiley and Sons, New York.

Zeigler, B. P., Belogus, D., Boshoi, A. (1980) ESP - An Interactive Tool for System Structuring. *Proc. of the 1980 European Meeting on Cybernetics and Research*, Hemisphere Press.

Zeigler, B. P. (1984a) *Multifaceted Modelling and Discrete Event Simulation*, Academic Press, London.

Zeigler, B. P. (1984b) System–Theoretic Representation of Simulation Models. *IIE Transactions*, March, pp. 19-34

Zeigler, B. P. (1987) Hierarchical, Modular Discrete Event Modelling in an Object Oriented Environment. *Simulation Journal*, vol 49:5, pp. 219-230.

Jerzy W. Rozenblit is an assistant professor in the Electrical and Computer Engineering Department at The University of Arizona. He received his Ph.D. in Computer Science from Wayne State University in Detroit, in 1985. His research interests are in the areas of modelling and simulation, system design, and artificial intelligence. He is a member of ACM, IEEE Computer Society, and The Society for Computer Simulation.

Jerzy W. Rozenblit
Dept. of Electrical and Computer Engineering
The University of Arizona
Tucson, Arizona 85721
(602) 621-6177