Proceedings of the 1993 Winter Simulation Conference
G.W. Evans, M. Mollaghasemi, E.C. Russell, W.E. Biles (eds.)

HIERARCHICAL RELATIONS IN SIMULATION MODELS

Joel J. Luna

Dynamics Research Corporation
60 Frontage Road
Andover, Massachusetts 01810, U.S.A.

ABSTRACT

Hierarchical modeling is a powerful means to handle system simulation model complexity. It can often be loosely defined, however, creating ambiguities in its application to simulation modeling. In order to define hierarchical modeling, different aspects of simulation models are defined. Then four principal model hierarchical relations are defined and discussed: representation, composition, substitution, and specification. Hierarchical relations are then applied to the appropriate model aspects to provide a more complete definition of hierarchical modeling.

1. INTRODUCTION

The progress in the development of simulation tools in the last decade has made simulation an increasingly viable alternative for system analysis. A pressing need for currently available simulation tools, however, is making the tools easier to learn and use (Pegden and Davis 1992; Ulgen and Thomasma 1990). Certainly the increasing use of graphics-based model building tools (Ball 1992, Conrad, Sturrock and Poorte 1993; O'Reilly and Ryan 1992) helps meet this need. Another direction to pursue, hierarchical modeling, will also greatly assist in simulation modeling, particularly of large and complex systems. Concepts for hierarchical modeling have been documented for nearly a decade (Zeigler 1984; Oren and Zeigler 1986; Oren 1989; Zeigler 1990), which have been implemented in academic research (Kim and Zeigler 1987; Ulgen and Thomasma 1990; Fishwick 1988; Cellier 1991; Oren 1984) and commercial development (Silverman and Stelzner 1989; Pegden and Davis 1992). In general, hierarchical modeling provides a means for managing the system complexity by partitioning the system into usable chunks which can then be independently manipulated (created, changed, extended). Hierarchical modeling becomes particularly powerful when the model partitions can be saved and retrieved as independent units, thus facilitating model code reusability. This tutorial seeks to address what hierarchical modeling is, what it provides, and how it is implemented in current tools.

2. ASPECTS OF SIMULATION MODELS

First, a definition of what a model is needs to be provided. This may seem pedantic, but discrimination must be made between viewing a system model as an abstract concept (which has no documented representation), as a diagram (which has a graphic representation but little to no formal notation, from which the observer often infers system behavior based on preconceived concepts of the system), as a mathematical representation (such as differential equations, which without an accompanying graphic representation makes it difficult for the user to form concept of the system), and as an implementation (the executable simulation code, whether it is in source or object form). Merely mentioning the word "model" immediately conjures up mental pictures of one or some combination of the above. These views comprise different aspects of what a model is.

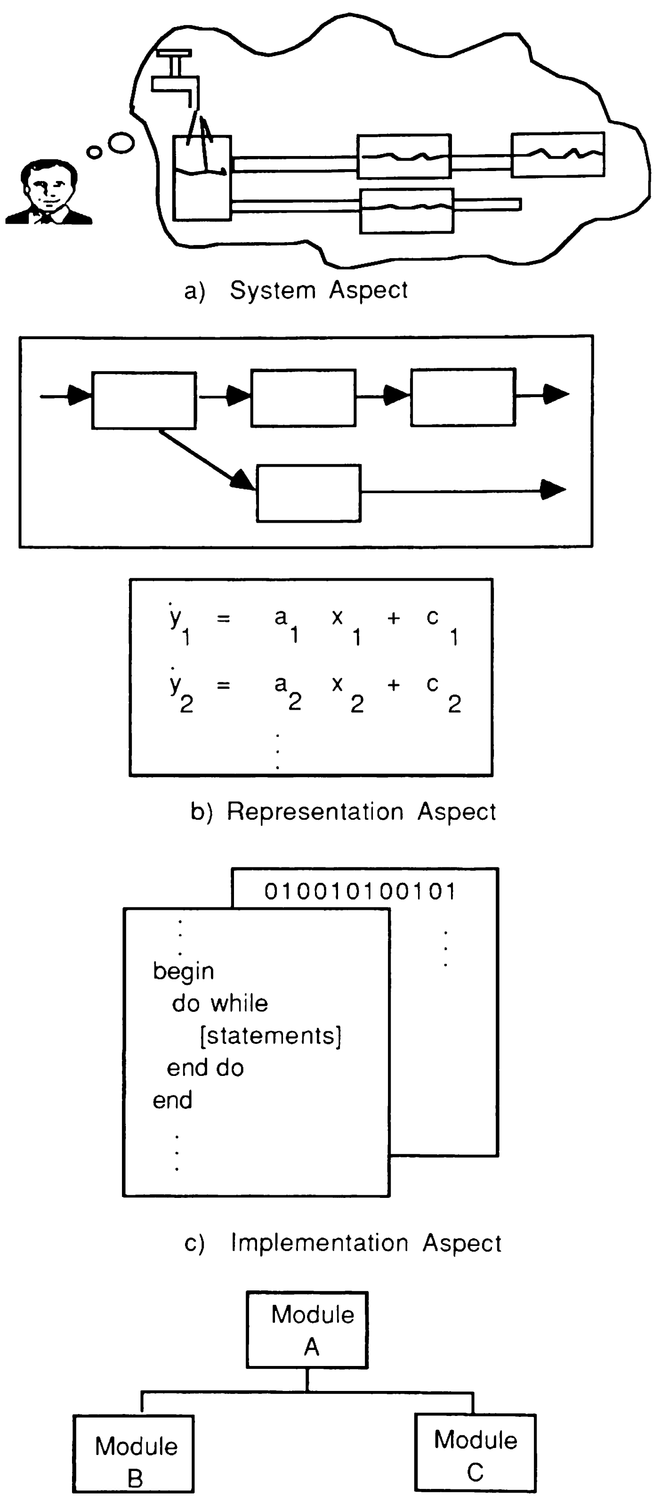
These aspects can be divided into four categories: system aspect, representation aspect, implementation aspect, and organization aspect. The system aspect corresponds to a perceived model of the system, typically based on preconceived ideas and observations, that exist as an abstract concept or mental model (Figure 1a). This model is typically biased by one's objectives for the model, interpretation through familiar formalisms (to a hammer everything looks like a nail), and other influences. In particular, one forms notions whether a system is regarded as hierarchical or not. For example, a military command and control system is perceived as being hierarchical because commands flow from the highest rank to the lowest rank. The representation aspect concerns the expression of the model in some form, whether graphical or mathematical or both (Figure 1b). The representation is the model aspect with which the user interacts most easily. The representation can be expressive but incomplete (such as in purely diagrammatic form), or it can be complete but lacking in expressive power (a system of equations). The implementation aspect concerns the model as executable code, whether in source or object code, which is implemented by a computing device (Figure 1c). It is at this level that many users typically dwell, but rather unsatisfactorily. This division of a model into different aspects can be mapped in part to Zeigler's (1984) view of a model. The organization aspect concerns the organization of the model implementation, particularly a model comprised of many parts which are related in various ways (Figure 1d). Zeigler defines the mathematical representation aspect of a model of a system as the system's model, and the implementation aspect of a model as the simulator (1984). This enumeration of the different aspects of a model will be used later to correspond to hierarchical relations.

3. HIERARCHICAL RELATIONS

A hierarchy is defined in Webster's as "a graded or ranked series" (1984, p. 569). A hierarchy essentially defines a type of relation in which the entities are grouped by levels. The relations exist only between adjacent levels (in which each level can have at most two). This lends to the familiar idea of there being an "up" and "down" relation at each level. As illustrated in Figure 2, a parent and child are hierarchically related in which the relation between a child and parent has both a "down" relation (from parent to child) and an "up" relation (from child to parent). A child is not hierarchically related to another child using the parent/child relation, although children are hierarchically related by the chronological (older/younger) relation. It is clear different types of model hierarchical relations can exist for the same entities. If a model is divided into parts, the parts can be hierarchically related. There are principally four different types of model hierarchical relations that can exist: representation, composition, substitution, and specification.

3.1 Representation

In the representation relation, the higher level does not have its own being per se, but is a representation of the lower level. For example, the United States Government is a balance of powers between the Executive, Congressional, and Judicial branches (Figure 3). The lower level, the branches of government with corresponding real entities, are represented as a whole in the higher level, the U.S. Government (although there is no entity which itself is the U.S. Government). Icons in graphic-based modeling tools, for example, are also representationally related to the underlying simulation model. The icon itself has no being as a model entity, but is used by the user (creating, connecting, deleting) as if it were.



$$\dot{y}_1 = a_1 x_1 + c_1$$
$$\dot{y}_2 = a_2 x_2 + c_2$$
$$\vdots$$

```
010010100101
:
begin
  do while
     [statements]
  end do
end
:
```

Figure 1. Model Aspects

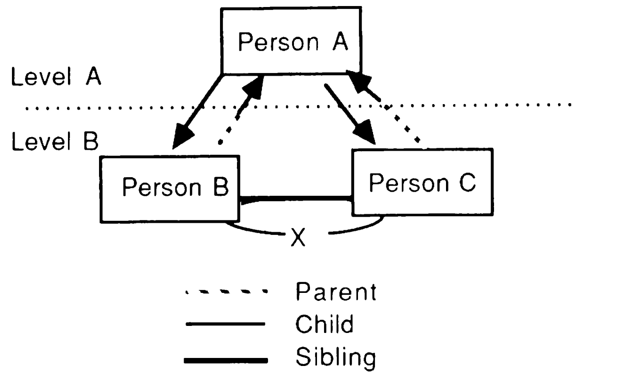


Figure 2. Parent and Child Hierachical Relations

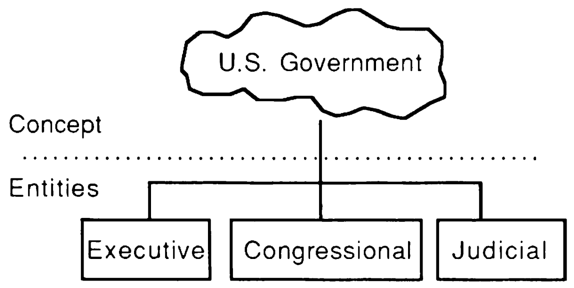
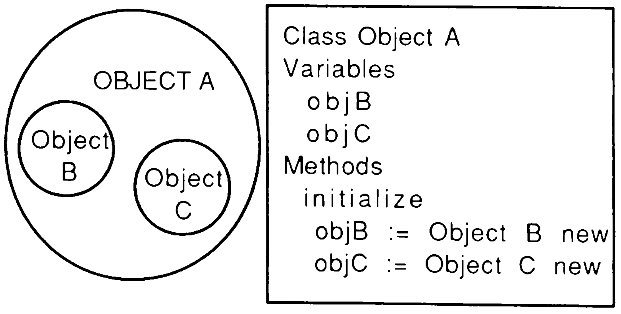


Figure 3. Representation Hierarchical Relation

3.2 Composition

The composition relation is one in which the higher level has its own being and employs the lower level in its behavior. For example, a military command has its own being at a higher level (the command post), but also employs the lower level (the commanded units). As another example (illustrated in Figure 4), an object in Smalltalk (Goldberg and Robson 1983; Digitalk 1988) will have its own variables and methods (the higher level), which themselves represent and manipulate other objects (the lower level).



```
Class Object A
Variables
  objB
  objC
Methods
  initialize
   objB := Object B new
   objC := Object C new
```

Figure 4. Composition Hierarchical Relation

3.3 Substitution

The substitution relation concerns the substitution for the lower level by the higher level in a given model. This relation encompasses both abstraction by reduction and morphic reduction as a means of simplifying the model. Abstraction by reduction (Figure 5) involves reducing the complexity of the lower level either by simplifying its systems specification in a process defined by Zeigler as association, or by moving to a simpler formalism defined as abstraction. The different levels in association correspond to the systems specification levels (i.e., Observation Frame, I/O Relation, I/O Function, I/O System, Structured System, and Coupling of Systems). The "up" relation in association is defined as realization while the "down" relation is defined as implementation. The different levels in abstraction correspond to formalisms which vary in complexity (e.g., FSM network to a Diagram, or FSM hierarchy to a Tree). For abstraction, the "up" relation bears the same name while the "down" relation is defined as concretization (Zeigler 1984). The term abstraction by reduction is borrowed from Fishwick, who gives a particularly well-grounded and thorough overview of abstraction (1988). Fishwick's abstraction by representation corresponds to the hierarchical representation relation, while his abstraction by total systems morphism corresponds to the substitution relation by morphic reduction to be described next.

Morphic reduction (Figure 6) is a simplification of a "larger" system into a "smaller" system in which the behavior observable at the specified systems specification level is preserved. The simplification is performed at the same systems specification level (as opposed to association, which moves between systems specification levels). The higher level in a morphic reduction then is "simpler" but homomorphic with a lower level.

Overall, the basic idea in the hierarchical substitution relation is that by substituting the higher level for the lower level, the model is simplified wit respect to some complexity measure. The lower and higher levels can be the same type of model (morphic reduction), different types (i.e., two different levels of the systems specification) but the same formalism (association), or different formalisms altogether (abstraction).

3.4 Specification

The specification relation is one in which the higher level and lower level are related by type, in which the lower level is a more specific type than the higher. For example, a car can be considered a type of object (as opposed to a truck or a motorcycle), which is a more specific type than "moving object with wheels". A car

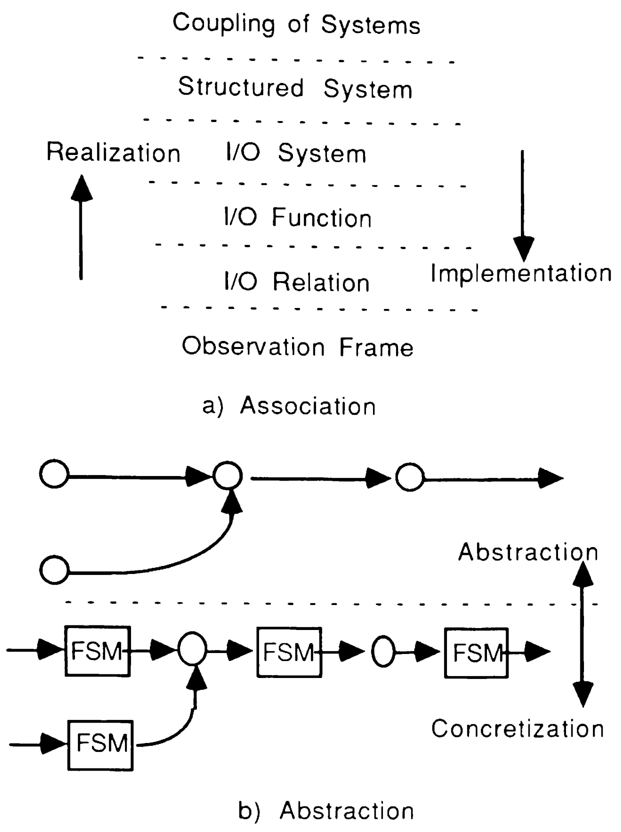


Figure 5. Substitution Hierarchical Relation: Abstraction by Reduction

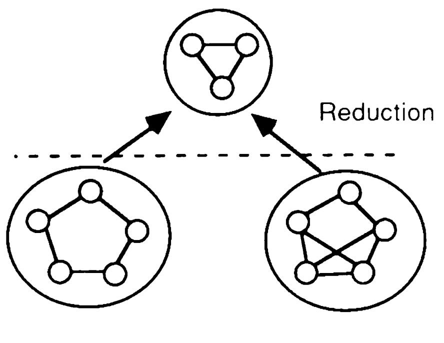


Figure 6. Substitution Hierarchical Relation: Morphic Reduction

has more specific characteristics by which we differentiate it from a truck or a motorcycle. Car, truck, and motorcycle then are hierarchically related by specification to "moving object with wheels" (Figure 7). The "up" relation is defined as generalization while the "down" relation is defined as specialization (Zeigler 1984). The class/subclass relations by which the inheritance mechanism is implemented in object-oriented languages are hierarchical specification relations. Odell (1992a) uses the same terminology to describe the class/subclass relation. Furthermore, Odell's use of the term composition (1992b) matches most closely the hierarchical representation relation, and his use of the term abstraction (1992a) matches Zeigler's composition relation in the System Entity Structure (Zeigler 1984; Rozenblit and Zeigler 1985; Zeigler 1990).

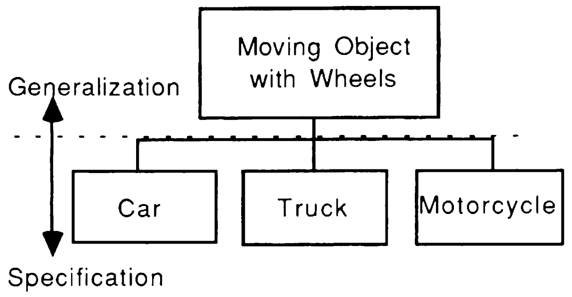


Figure 7. Specification Hierarchical Relation

4. APPLICATION OF HIERARCHICAL RELATIONS TO MODEL ASPECTS

Now that the different hierarchical relations (representation, composition, substitution, and specification) are defined, they can be applied to the model aspects described earlier, in particular, the aspects of representation, organization and implementation. The application of hierarchical relations to model aspects is summarized in Table 1.

	Hierarchical Relation			
Model Aspect	Rep.	Comp.	Sub.	Spec.
Representation	x	x		
Implementation		x		
Organization		x	x	x

Table 1. Application of Hierarchical Relations to Model Aspects

4.1 Representation Model Aspect

The hierarchical relations which pertain to the representation aspect of a model are representation and composition. The representation hierarchical relation is the most common, in which an icon at a higher level represents another set of interconnected icons at a lower level. The user typically changes level using a "point-and-click" approach with a mouse. Structured design tools (Meta Software 1992a) and some simulation tools (Meta Software 1992b, Silverman and Stelzner 1989; Pegden and Davis 1992; Ulgen and Thomasma 1990; Withers 1992) provide this approach. It is important to note that in the representation hierarchy of a model, only the base or root elements actually exhibit model behavior (e.g., SIMAN primitives in Arena - Pegden and Davis 1992). In this respect, the whole is the sum of the parts. Though model elements in the higher levels may exhibit behavior (e.g., the "Subsystem" class implemented by Ulgen and Thomasma 1990), it is behavior with respect to its representation rather than its implementation. On the other hand, model elements in higher levels do exhibit their own behavior in a composition hierarchical relation. We think of a person as being more than arms, legs, organs, etc. In this way, the whole is greater than the sum of the parts. The result is that because the elements in higher levels also exhibit model behavior, information (data and code) must be represented to the user for them as well.

4.2 Implementation Model Aspect

The implementation aspect of a model either utilizes the hierarchical composition relation or has no hierarchy at all. The implementation is hierarchical when it preserves the composition relations defined in the representation aspect. An example of hierarchical implementation is DEVS-Scheme (Kim and Zeigler 1987), in which the model is built up by successive layers of model and simulator modules. The model is executed by the successive upward and downward flow of messages to implement the timing of model events. It is important to note that a hierarchical representation does not require a hierarchical implementation. This is the approach taken by others, for example, Cellier (1991), Pegden and Davis (1992) and Luna (1992). The former emphasize implementation in simulation specific languages (which can include SmartSim's generation of SIMAN code in Ulgen and Thomasma 1990) while the latter is a more general object-oriented approach.

4.3 Organization Model Aspect

The organization aspect of a model can utilize the hierarchical relations of specification, composition, and substitution. It should be noted that representation of the organization of a model greatly facilitates model design and construction just as representation of the model itself does. The specification relation identifies which model elements are related by type. Variations between elements can be organized by using an object-oriented approach, even if the elements themselves are not implemented in an object-oriented language (although they are much easier to maintain that way). Smalltalk (Goldberg and Robson 1983; Digitalk 1988) provides a browser which arranges classes in a hierarchical specification relation, which makes the perusal of classes much easier. The composition relation identifies which model element types are aggregations of other types. While Smalltalk does not provide a browser for the composition relation, SimKit provides one for both the specification and composition relation (Silverman and Stelzner 1989). Note that composition relation in the organization aspect refers to model element types, while the composition relation in the representation aspect refers to model element instances.

Any discussion of hierarchical organization of model elements should include Zeigler's System Entity Structure (SES), which combines composition, specialization and instantiation relations in a consistent structure. It is not clear, however, how (if at all) entities are related by substitution (that is, which model element can be substituted by other model elements). Fishwick's HIRES (1988) specifically relates models in a hierarchy of abstractions. Thus, any model can be substituted by another model at a different level of abstraction (according to specified rules). It appears, however, that organization of model elements by substitution, so that a user may select one of several elements based on the desired simplicity or complexity, still needs to be implemented.

5. CONCLUSION

The claim that a model is hierarchical needs to be evaluated with respect to the model aspect and the hierarchical relation applied. In particular, the hierarchical relations of representation and composition apply to the representation model aspect; the relations of specification, composition, and substitution apply to the organization model aspect, and the composition relation applies to the implementation model aspect. While some of the hierarchical relations are currently implemented, the combination of hierarchical relations in the representation and organization model aspects would significantly ease and enhance the simulation model design and construction process.

REFERENCES

Ball, D. 1992. GPSS/VI. In Proceedings of the 1992 Winter Simulation Conference, pp. 426-430

Cellier. F.E. 1991. Continuous System Modeling. Borlin Springer-Verlag.

Conrad, S.A., Sturrock, D.T., and Poorte, J.P. 1992. Introduction to SINMAN/Cinema. In Proceedings of the 1992 Winter Simulation Conference, pp. 377-379.

Digitalk, Inc. 1988. Smalltalk/V286: Tutorial and Programming Handbook. Digital, Inc.

Fishwick, P.A. 1988. The Role of Process Abstraction in Simulation. IEEE Transactions on Systems, Man, and Cybernetics, 18 (1), pp. 18-39.

Goldberg, A. and D. Robson, 1983. SmallTalk-80: The Language and Its Implementation. Reading, MA: Addison-Wesley.

Kim, T.G., and Zeigler, B.P. 1987. The DEVS Formalism: Hierarchical, Modular system Specification in an Object-Oriented Framework. In

Proceedings of the 1992 Winter Simulation conference, pp. 559-566.

Luna, J. 1992. Hierarchical, Modular Concepts Applied to an Object-Oriented Simulation Model Development Environment. In Proceedings of the 1992 Winter Simulation Conference, 694-699.

Meta Software Corp. 1992a. Design/IDEF Version 2.0: Users' Manual. Meta Software Corp.

Meta Software Corp. 1992b. Design/IDEF Version 1.9: A Reference Manual. Meta Software Corp.

Odell, J. 1992a. Managing Object Complexity, Part I: Abstraction and Generalization. Journal of Object-Oriented Programming, 5 (5), pp. 19-21.

Odell, J. 1992b. Managing Object Complexity, Part II: Composition. Journal of Object-Oriented Programming, 5(6), 17-20.

Oren, T.I. 1989. Bases for Advanced Simulation: Paradigms for the Future. In Modeling and Simulation Methodology: Knowledge Systems' Paradigms, Ed. M.S. Elzas, T.I. Oren, and B.P. Zeigler, pp. 29-43.

Oren, T.I. and Zeigler, B.P. 1986. Multifaceted, Multiparadigm Modeling Perspectives: Tools for the 90;s. In Proceedings of the 1986 Winter Simulation Conference, pp. 708-712.

Pegden, C.D., and Davis, D.a. 1992. Arena: A SIMAN/Cinema-Based Hierarchical Modeling System. In Proceedings of the 1992 Winter Simulation Conference, 390-399.

Rozenblit, J.W. and Zeigler, B.P. 1985. Concepts for Knowledge-Based System Design Environments. In Proceedings of the 1985 Winter Simulation Conference, pp. 223-231.

Silverman, D. and Stelzner, M. 1989. SimKit Knowledge-Based System Design Environments. In Proceedings of the 1985 Winter Simulation Conference, pp. 223-231.

Ulgen, O.M. and Thomasma, T. 1990. SmartSim: An Object-Oriented Simulation Program Generator for Manufacturing Systems. International Journal of Production Research, 28 (9), 1713-1730.

Webster's Ninth New Collegiate Dictionary. 1984. Merriam-Webster.

Withers, D.H. 1992. Issues in Hierarchical Modeling with Procedural Languages. ORSA/TIMS Joint National Meeting, San Francisco, CA.

Zeigler, B.P. 1984. Multifaceted Modelling and Discrete Event Simulation. Orlando FL: Academic Press Inc.

Zeigler, B.P. 1990. Object-Oriented Simulation with Hierarchical, Modular Models. Orlando, FL: Academic Press.

Zeigler, B.P., Kim, T.G., Sevinc, S., and Zhang, G. 1989. Implementing Methodology-Based Tools in DEVS-Scheme. In modeling and Simulation Methodology: Knowledge Systems' Paradigms, Ed. M.S. Elzas, T.I. Oren, and G.P. Zeigler, pp. 431-450. North-Holland

AUTHOR BIOGRAPHY

JOEL J. LUNA is a Senior Analyst at Dynamics Research Corporation, He is primarily involved in application of systems analysis techniques, especially modeling and simulation, to a variety of projects. His current interests are in the areas of simulation, modeling, and object-oriented programming.