Proceedings of the 1995 Winter Simulation Conference
ed. C. Alexopoulos, K. Kang, W. R. Lilegdon, and D. Goldsman

A TAXONOMY OF MODEL ABSTRACTION TECHNIQUES

Frederick K. Frantz
Computer Sciences Corporation
One MONY Plaza, Mail Drop 37-2
Syracuse, New York 13202, U.S.A.

ABSTRACT

Model abstraction is a method for reducing the complexity of a simulation model while maintaining the validity of the simulation results with respect to the question that the simulation is being used to address.

Model developers have traditionally used a number of abstraction techniques, and simulation researchers have conducted formal research to build a theoretical foundation for model manipulation. More recently, researchers in the artificial intelligence (AI) subfield of qualitative simulation have also been developing techniques for simplifying models, determining whether models results are valid, and developing tools for automatic model selection and manipulation. Metamodeling also can be considered as an abstraction technique.

The purpose of this paper is to provide a taxonomy of abstraction techniques drawn from these fields. This taxonomy provides a framework for comparing and contrasting various abstraction techniques.

1 CONCEPT OF MODEL ABSTRACTION

We begin by describing our concept of model abstraction. Figure 1 illustrates the modeling and simulation process. There exists some **real world system** (either actual or hypothesized), of which we want to create a model to use to answer some question(s), such as how a real-world system would respond to particular stimuli.

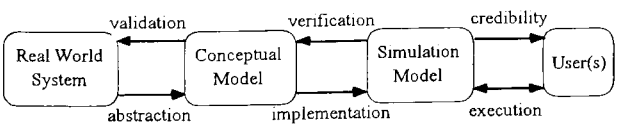

Figure 1: Model Abstraction in the Simulation Process

Development of a simulation model, as shown in Figure 1, involves two steps. The first step is development of a **conceptual model**, a way of "thinking about" and representing the real world system. A crucial design decision is determination of what factors influence system behavior, the system behaviors to be incorporated into the model, and the representation of those behaviors. This determination is based on what questions the model will be used to answer; the computational and development resources available to build, validate, and use the model; and the data available which describes the real-world system and its interfaces. We refer to this determination as **abstraction**, since the conceptual model is a simpler representation of the more complex real-world system. The process of determining whether the conceptual model is an adequately accurate representation of the real-world system is **validation**.

A conceptual model is then **implemented** in the form of a computer-executable set of instructions known as the **simulation model**. **Verification** is the process of determining whether the implemented version of the model is an accurate representation of the conceptual model.

Modeling and simulation initially focused on the development of single conceptual and simulation models for a real-world system. However, as new questions arise after the model is in use, and as models are decomposed and/or combined to address new questions, a single conceptual model becomes inadequate. Zeigler (1984) provides a more effective approach using the concept of multifaceted modeling. A multifaceted model is a set of related conceptual models for a real-world system, emphasizing different aspects of the system required to address different questions about the real-world system.

Just as abstractions are a mapping from a real-world system to a conceptual model, they can map one conceptual model to another conceptual model in a multifaceted model. The abstraction techniques discussed in this paper focus on mappings among conceptual models, although the principles can also be applied to development of conceptual models from real-world systems.

We define abstraction techniques as: *techniques that derive simpler conceptual models while maintaining the validity of the simulation results with respect to the*

question being addressed by the simulation. Abstractions are valid only if they preserve the validity of the simulation results, which are dependent on the question being asked. We refer to the question(s) for which the simulation is being executed as the **simulation requirement**.

This definition bounds our taxonomy. Because we are dealing with the conceptual model, we do not include code optimization techniques that change only the implementation of a conceptual model.

2 ILLUSTRATION OF MODEL ABSTRACTION

To illustrate the concept of a model abstraction, consider the following simple example of a model of a grocery store adapted from Zeigler (1976). In the real world system, there is a grocery store with one checkout lane. Customers enter the store, spend some time shopping, then get into the checkout line, wait (if necessary) until they reach the front of the line, then check out and exit the store. The model input set consists of the time and identity of the customer when a customer enters the store. A shopping time duration is assigned to each entering customer from a probability distribution function. At the end of the shopping time duration, if the checkout line is empty (i.e., the checker is not busy), then a checkout duration time is assigned from a probability distribution function. At the end of the checkout duration time, the customer exits the store. If the checkout line is not empty, then the customer is inserted into a first in, first out queue. When the customer reaches the front of the queue, a checkout duration time is assigned from a probability distribution function; at the end of the checkout duration time, the customer exits the store.

This simulation model could be used to determine the average amount of time a customer is in the store, or the average amount of time that a customer waited in line to check out. Suppose, however, that the simulation requirement is to determine the average amount of time that the checker was busy, and the average length of time the checker must work without a break. There are certain aspects of the model that are not relevant to the problem at hand. We need not be concerned with specific customers; in fact, we need not represent the individual customers in line. We abstract the model by first eliminating the identity of individual customers. Then when a customer's shopping time is finished, the checkout time is immediately derived and added to a model variable which expresses the amount of time until the checker is no longer busy. The times when the checker's state (Busy or Idle) changes are the only required model outputs. The queue, and the order of customers in the queue, is no longer relevant, and can be ignored. The validity of the results is preserved because the model output of checker state changes is identical for both the original model and the abstracted model.

3 FOUNDATIONS

The abstraction techniques described in this paper are drawn from both the modeling/simulation community and the artificial intelligence (qualitative reasoning) community. Techniques for model abstraction have been in existence since the first model was coded, though they have generally been applied in an ad hoc fashion. The first attempt at formalizing discrete event simulation, providing a formal basis for definition and discussion of abstraction techniques, is found in Zeigler (1976). Zeigler includes an informal discussion of abstraction techniques, then provides a formal basis, called the Discrete Event Simulation (DEVS), for reasoning about discrete event simulation systems. Other formalisms have been advanced since then, such as Radiya and Sargent (1994).

We also draw from model-based reasoning, a field of study in artificial intelligence (AI). AI researchers, led by Kuipers (1994), attempted to develop systems which would automatically reason about common physical systems. Researchers observed that humans reason about such systems in a qualitative manner. Thus much research was focused on qualitative, rather than quantitative, models of real world systems. These models tended to generate complete envisionments (all possible state trajectories). This is feasible since qualitative models can have a finite number of states. However, reasoning about real world situations requires models of sufficient complexity that generation of complete envisionments rapidly became computationally demanding. The next step was to develop a hierarchy of models with varying degrees of complexity, and execute the simplest model required for the reasoning to be performed.

Fishwick and Zeigler (1992) note substantial parallels between their work and the ongoing research in qualitative simulation. This realization of the potential synergism of qualitative and traditional simulation techniques has been reiterated. Miller et al. (1992), Fishwick (1992), and Fishwick et al. (1994) provide general rationale and approaches for synergizing traditional simulation and AI modeling/simulation techniques. The remainder of this paper is an exploration of a specific focus of that synergy: model abstraction techniques.

4 A TAXONOMY OF MODEL ABSTRACTION APPROACHES

The taxonomy of model abstraction techniques is shown in Figure 2. The first level reflects the distinction between abstraction of behaviors within a model and abstraction of the model boundary. Model boundary abstraction changes the real-world factors accounted for in the model by changing the exogenous variables of the model. Behavior abstraction eliminates the transitions through system states whose distinction is irrelevant to the simulation requirement.

It is also possible to abstract a model without changing either the system boundary or the behaviors, by determining input-output relationships in a different way. This category is identified as changes to model form, the third category in our taxonomy.

The discussion in this section is organized into three subsections, each focused on one of these three major abstraction classes. We then conclude this section with a discussion of the relationship of this taxonomy to other taxonomies of abstraction techniques.

4.1 Model Boundary Modification

The first class of model abstraction techniques is based on modification of the input variable space. For example, a particular model of a physical system could be abstracted by ignoring the effects of friction. Within this category, we define two subcategories: explicit abstractions and derived abstractions. Explicit abstraction require the modeler to specifically identify the exogenous variables to be eliminated in the abstraction as part of the model definition. Derived abstractions are determined by analyzing a particular model or a particular set of input variables, and determining the minimum set of those variables which are required to meet the simulation requirement.

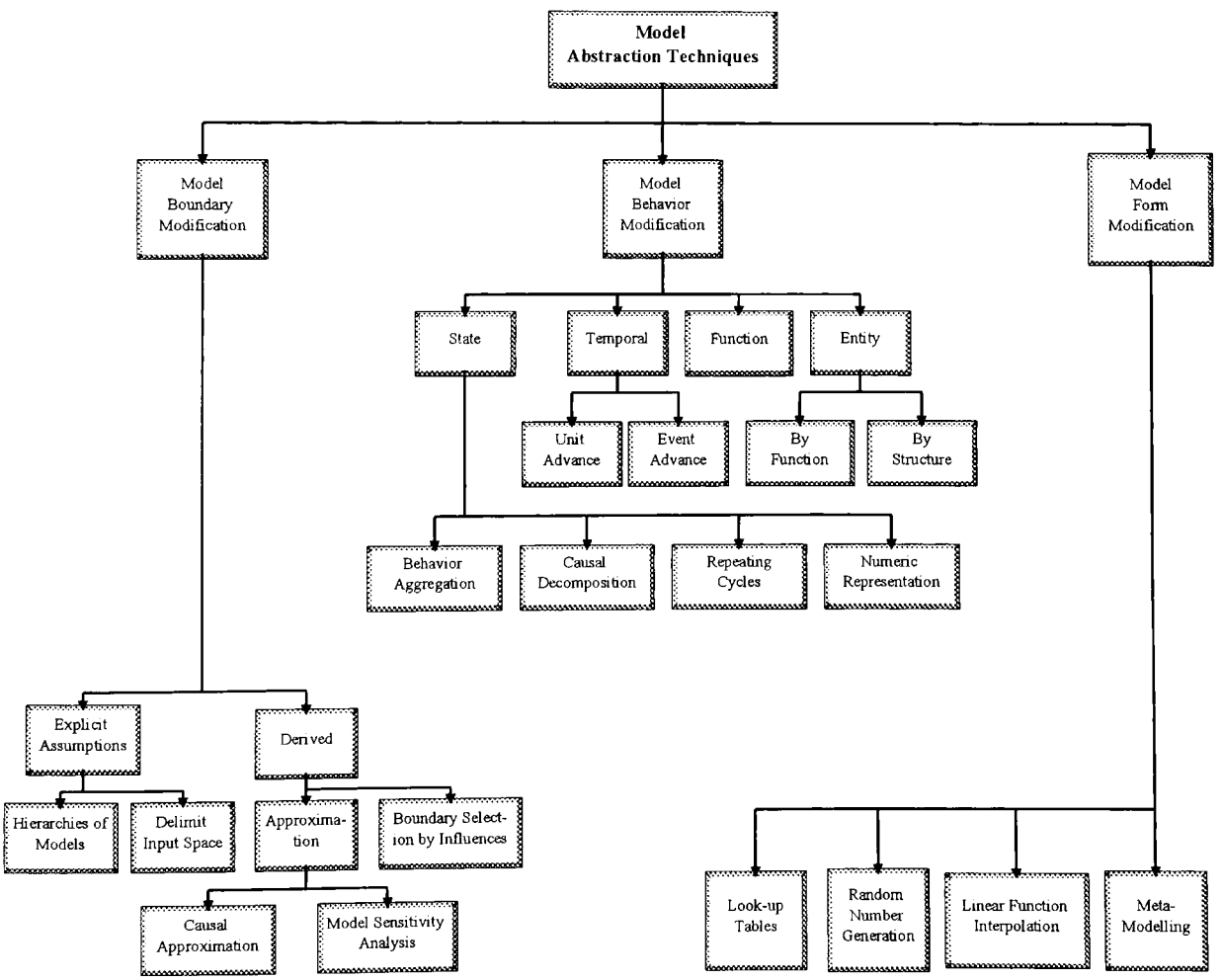

Figure 2: Taxonomy of Model Abstraction Techniques

4.1.1 Explicit Assumptions

The first category of assumptions deals with those that are expressed explicitly by the model developer. Within this category, we include two subcategories: hierarchies of models, in which the abstractions are expressed as specific relationships between models; and limitations of input ranges, in which assumptions are expressed as constraints on the ranges of input variables.

4.1.1.1 Hierarchies of Models

A pre-defined hierarchy of models or model components can be created, in which the relationships between models or model components in a hierarchy are defined as explicit assumptions. For example, a pre-defined hierarchy of models could include one simplified frictionless model as well as a more complex model in which friction is explicitly addressed. The assumptions drive the models; the abstraction of friction is the explicit assumption, and an appropriate model which represents the more simplistic view of the world must then be developed/incorporated.

Model hierarchies have seen extensive use in the simulation domain (see Zeigler (1984) for a formal discussion), although assumptions are not always explicit. This approach is also popular in the field of qualitative reasoning about physical system. There are numerous "ideal" laws in physics which "abstract" away some complicating factor to allow a simpler expression of the interaction. Since such laws are the basis of systems used for qualitative reasoning, hierarchies with explicit abstraction by parameter elimination are used. Examples of pre-defined model hierarchies with explicit assumptions include the Graph of Models concept of Addanki, Cremonini, and Penberthy (1991) and Compositional Modeling of Falkenhainer and Forbus (1991).

4.1.1.2 Delimit Input Space

Another explicit abstraction technique involves limiting the domain of the inputs provided to the model. In Zeigler's (1976) terminology, this technique is equivalent to defining the experimental frame of reference within which the model is valid. For example, a model can be limited to a range subset, such as a model which only applies if the water temperature is in the range of 0 to 100 degrees Celsius. Computation associated with transition of water to solid or gas can be eliminated, thereby simplifying the model.

4.1.2 Derived Abstractions

The second category of model boundary abstractions is derived abstractions. These techniques take an existing model and/or input variable set and determine whether and what simplifications can be made. They do not depend on a priori decisions made by a model developer, so they can be applied to existing models. We have identified two types of derived abstraction techniques: approximation, and a technique based on the influences of input variables.

4.1.2.1 Approximation

An alternative to the pre-defined model hierarchy is parameter elimination based on the requirements of the simulation to be executed, rather than defined externally. Analysis determines which parameters in a model can be eliminated, yielding a new and computationally simpler model that provides approximate results that are within a tolerance necessary to meet the requirements of the simulation.

Model sensitivity analysis is one example of a traditional approach to approximation abstraction. Weld (1992) applies such analysis to qualitative models to formalize an approximation approach. Nayak (1992) describes an alternative approximation approach based on the causal relationships of model parametetrs.

4.1.2.2 Selection by Influences

Selection by influences uses causal relationships among variables to find the simplest model that meets simulation requirements. Model abstractions are defined by dividing the variables in a model into three classes: exogenous, dependent, and irrelevant. The model abstraction space is then searched for the abstraction with the minimum number of exogenous variables. This approach using qualitative models is described by Rickel and Porter (1994).

4.2 Modification of Behaviors

The second category of abstractions involves modification of behaviors within a model rather than the inputs to a model. These abstractions involve aggregating some aspect of the model, which form the next level of the taxonomy: states, time, entities, and functions. Such aggregations are not mutually exclusive, and aggregation of one aspect may require concomitant aggregation of another aspect.

4.2.1 State Aggregation

We define a model state as an n-tuple of system state variables. One way to create a simpler model is to define a model with fewer possible states.

This notion of aggregation (in a qualitative simulation context) is illustrated in Figure 3. At the more abstract level, there is a single state in which the value of X is greater than the value of Y. This aggregate state is comprised of five states, in which the addition of 0 as a landmark value provides the basis of the decomposition. Conversely, the five "lower level" states are subsets of, and can be aggregated into, the higher level state.

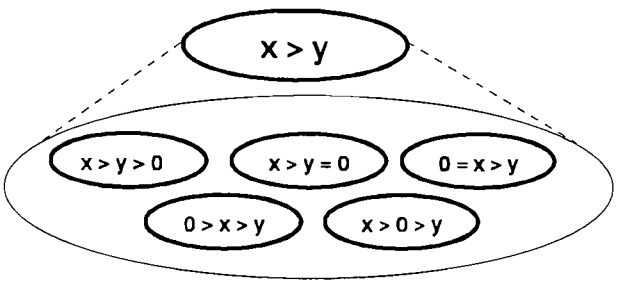

Figure 3: State Aggregation Example

Zeigler's (1976) definition of coarsening of range sets fits our concept of state aggregation. In Zeigler's terms "the lumped model variables can describe relatively few of the conditions that the base model components can be found in." In our terminology, replacing lumped model with abstracted model, base model with original model, and conditions with state, this statement is as follows: the abstracted model has fewer states than the original model, i.e. the states have been aggregated.

There are a number of approaches to determining how to aggregate states. Four specific approaches identified in the literature include the following: behavior aggregation, causal decomposition, aggregation of cycles, and numeric representation. Each of these approaches is addressed in a subsection which follows.

4.2.1.1 Behavior Aggregation

The principle of behavior aggregation is to combine states whose distinctions are irrelevant to the simulation requirement. One such situation is represented by a transition graph representation of simulation history, as shown in Figure 4. Each system state is represented as a node in the graph, with arcs representing transitions among states. Where there is a single input single output subgraph, that subgraph is a candidate for abstraction to a single system state, removing the necessity to compute the separate states of the subgraph. Consider the example of a model of military unit movement. State S2 represents the initial location of the unit; states S3 and S4 represent intermediate states following different routes of march. State S5 represents the unit at the destination location and time. If the distinction between S3 and S4 is irrelevant to S5 (meaning that the simulation results are only dependent on whether the unit arrives at the destination location by a certain time), then the model can be abstracted to eliminate the decision logic to determine the route of march.

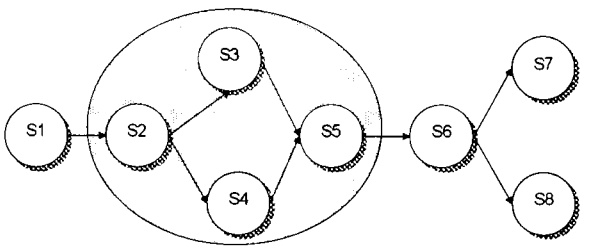

Figure 4: Notional Behavior Aggregation Example

The abstraction described in Section 2 is a behavior aggregation. All system states in the original model where the sums of the customer checkout times are equal (regardless of how the checkout times are distributed among individual customers) are aggregated to a single state.

4.2.1.2 Causal Decomposition

The philosophy of causal decomposition (Clancy and Kuipers 1994) is to divide a model into loosely connected components, then execute each component separately while specifically addressing the component interactions. The causal relations among components are analyzed such that execution of one component can constrain the behavior of subsequently executed components. Also, where there is no causal relationship the components can be executed in parallel. Causal decomposition is implemented as three major steps: variable partitioning, creation of submodels, and then executing the partitioned simulation.

4.2.1.3 Aggregation of Cycles

The aggregation of cycles concept is to combine system states that reflect similar sequences in which distinctions among the individual sequences are irrelevant to the final outcome. For example, an iterative determination of a situation (such as checking the distance between two closing targets) could be abstracted to determine the point when the targets are

within range. Weld (1986) describes an approach to this type of abstraction in qualitative simulation.

4.2.1.4 Numeric Representation

Even the representation of values in a model forms the basis for potential abstraction. Consider the example of a model that is abstracted merely by representing all numeric values as integer values rather than double precision floating point values. Since the number of bits used to represent the values in the computer is larger in double precision representation, converting the representation to single precision reduces the number of potential system states, which is abstraction by state aggregation. An integer state encompasses all double precision states that round to the integer value, e.g. the state i = 1 is an aggregation of the states whose double precision representation of the value of $i \in [0.5, 1.5)$.

4.2.2 Temporal Aggregation

The second type of aggregation in our taxonomy is defined as temporal abstraction. This type of abstraction reduces computational complexity by reducing the granularity of the time advance. This abstraction can be applied in either unit advance or event advance discrete event simulation. With unit advance, the abstraction technique is implemented by merely changing the time step. With event advance, the abstraction involves assuming that events which occur "very closely" in time are considered simultaneous.

Time warp time management schemes for parallel simulations (Jefferson 1985) are another example of temporal aggregation. The principle of time warp is to execute events in parallel, maintaining sufficient information to roll back the event sequences if the temporal ordering of events is significant to simulation outcome. In our terminology, this amounts to abstracting a set of states that differ only due to execution sequence into an aggregate state in which execution order is indeterminate. Rather than determining abstractions a priori, the time warp approach assumes that states can be aggregated, then reverts to the disaggregated state definition when it is determined that the disaggregated states contain significant distinctions.

4.2.3 Entity Aggregation

Abstraction by entity aggregation refers to the use of a "higher level" entity to represent a collection of "lower level" entities. This abstraction is particularly useful in models whose entities have a natural hierarchical structure. Entity aggregation is often accompanied by other aggregation as well. The appropriate time cycle for an entity representing an entire army is not the same as the appropriate time cycle for an individual tank. We note two different criteria by which entities are typically aggregated: structure and function.

4.2.3.1 By Function

One approach is to aggregate entities which perform a common or similar function. The resulting entity performs a function that is the aggregate of the individual functions. Consider a model of a bank in which there are multiple servers (tellers) but a single queue (line). A model could have each teller represented by a separate entity, or by a single aggregate entity with correspondingly shorter service times. In this case the lower level entities are performing identical functions, and the aggregated entity is modeled as also performing that identical function.

4.2.3.2 By Structure

Entity aggregation can also be based on a hierarchical structure of entities, in which abstraction is accomplished by replacing lower level entities with higher level entities. For example, a model which represents the individual activities of a collection of tanks could be replaced by a model of a tank regiment. Behavior in these examples is substantially different as a result of the abstraction, even if the overall output is the same. The information required to model individual tanks, and the behavior generated by such a model, is much different from the information required by and generated by a model of a tank regiment.

We distinguish this situation from entity aggregation by function. In this case the aggregated entity is performing functions different from the lower level entities, rather than a composite or union of lower level entity functions.

4.2.4 Function

The final technique for entity aggregation is that of function aggregation. Function aggregation is different from entity aggregation by function. Entity aggregation by function involves changing the entities in the model. Function aggregation does not change the entities that are represented, but aggregates the functions that those entities perform.

For example, consider an air warfare model that includes the activities required to prepare an aircraft for

a new mission upon completion of a mission. Specific functions could be represented such as refueling, reloading weapons, damage repair, and so on. These functions could also be aggregated as simply a mission turn-around function. The aircraft and base entities still exist, but the details of some functions are abstracted.

4.3 Modification of Model Form

The third category of model abstraction techniques is modification of model form, characterized by a simplification of the input-output transformation within a model or model component. The same set of inputs is used in both the original and abstracted models, distinguishing this category from model boundary modifications. The entities, states, and time frames are the same, distinguishing this category from behavior modifications. The difference in the abstracted model is the manner in which parameter values are determined. These abstraction techniques are based on deriving the input/output relationships and then using a more computationally efficient means to compute the parameter value.

4.3.1 Look-Up Table

The use of a look-up table is a venerable tool among simulation model developers. In this abstraction, a transformation function is represented by a set of values; the input value is transformed to an index value into a data table, and the output value is then determined by retrieving the indexed value. Look-up tables provide an efficient abstraction since they eliminate a substantial amount of computation, but at a storage cost.

4.3.2 Probability Distribution

The notion of replacing the computation of a parameter value with a pseudorandomly-generated number has long been recognized as an abstraction technique. Zeigler (1976) identified it as a technique for creating lumped models. He identifies this technique as "replacing deterministic variables with random variables."

4.3.3 Linear Function Interpolation

Modeling by Linear Function Interpolation (LFI) provides a level of abstraction between that of a simple look-up table and a full polynomial representation. To increase resolution the classic look-up table requires additional data points. Doubling resolution requires doubling the number of data points and so on. To model any given polynomial more accurately without increasing the number of breakpoints, a typical LFI system uses a look-up table whose entries are points (breakpoints) on the polynomial curve. The LFI model implementation serves to reduce a polynomial function to a series of straight line approximations.

4.3.4 Metamodeling

A metamodel is a mathematical approximation of a more complex process or model. It is derived from analysis of input-output pairs, generally treating the more detailed model as a black box. Early metamodeling work focused on developing a best polynomial fit for input/output data generated by the model. There are several metamodeling techniques that have been developed within the simulation community. Burton (1994) identifies the following categories of techniques in an overview of metamodeling techniques:

- parametric polynomial response surface approximations,
- splines,
- radial basis functions,
- kernel smoothing,
- spatial correlation models, and
- frequency-domain approximations.

More recently, Caughlin (1994) describes a metamodeling approach to identify the underlying processes that define the system.

These techniques share in common simpler approximations of the input-output transformations of a model. Because the objective of a metamodel is a simpler expression of the model, metamodeling can be considered as an abstraction technique which changes the form of the model.

4.4 Comparison with Other Taxonomies

Fishwick (1988) describes a taxonomy of abstraction techniques, which covers a broader scope. He includes items such as cerebral abstractions, which is beyond our specific focus on simulation models. However, his taxonomy does not address specific techniques; rather, his taxonomy is oriented towards the application of the abstraction (e.g., sensory presentation). Most of the techniques described in this paper would most likely fit his broad definition of partial systems morphism.

5 FUTURE WORK

The taxonomy of the preceding section provides a framework for comparing abstraction approaches from various sources. It provides an initial step towards

better understanding and utilization of abstraction techniques, and provides the baseline for several continuing thrusts:

1. Formally define the categories of abstraction techniques (e.g., in DEVS).
2. Characterize simulation models in terms of criteria that can be used to select the best abstraction approach for a particular model and simulation requirement.
3. Identify synergy among different types of abstraction techniques.

The ultimate goal is to exploit abstraction techniques to create more flexible, responsive simulation models.

ACKNOWLEDGMENTS

This research has been funded by the U.S. Air Force under contract F30602-94-C-0272. The ideas in this paper evolved with the collaboration of Mr. John Ellor of Computer Sciences Corporation, under the guidance and direction of Mr. Al Sisti, Rome Laboratory.

REFERENCES

Addanki, S., R. Cremonini, and J. Penberthy. 1991. Graphs of models. *Artificial Intelligence* 51:145-177.

Burton, R. 1994. Metamodeling: a state of the art review. In *Proceedings of the 1994 Winter Simulation Conference*, ed. J. D. Tew, S. Manivannan, D. A. Sadowski, and A. F. Seila, 237-244. Institute of Electrical and Electronics Engineering, San Francisco, California.

Caughlin, D. 1994. A metamodeling approach to model abstraction. In *Proceedings of the Fourth Dual Use and Technologies Conference*, 387-396. Institute of Electrical and Electronics Engineering, San Francisco, California.

Clancy, D., and B. Kuipers. 1994. Model decomposition and simulation. In *Working Papers of the Eighth International Workshop on Qualitative Reasoning About Physical Systems (QR-94)*. Nara, Japan.

Falkenhainer, B. and K. Forbus. 1991. Compositional modeling: finding the right model for the job. *Artificial Intelligence* 51:95-143.

Fishwick, P. 1988. The role of process abstraction in simulation. *IEEE Transactions on Systems, Man, and Cybernetics* 18:18-39.

Fishwick, P. 1992. An integrated approach to system modeling using a synthesis of artificial intelligence, software engineering, and simulation methodologies. *ACM Transactions on Modeling and Computer Simulation* 2:307-330.

Fishwick, P. and B. Zeigler. 1992. A multimodel methodology for qualitative model engineering. *ACM Transactions on Modeling and Computer Simulation* 2:52-81.

Fishwick, P., N. Narayanan, J. Sticklen, and A. Bonarini. 1994. A multimodel approach to reasoning and simulation. *IEEE Transactions on Systems, Man, and Cybernetics* 24:1433-1449.

Jefferson, D. 1985. Virtual time. *ACM Transactions on Programming Languages and Systems* 7:404-425.

Kuipers, B. 1994. *Qualitative reasoning: modeling and simulation with incomplete knowledge.* Cambridge, MA: MIT Press.

Miller, D., J. Firby, P. Fishwick, D. Franke, and J. Rothenberg. 1994. AI: what simulationists need to know. *ACM Transactions on Modeling and Computer Simulation* 2:269-284.

Nayak, P. 1992. Causal approximations. In *Proceedings of the Tenth National Conference on Artificial Intelligence,* 703-709. Cambridge, MA: AAAI/MIT Press.

Radiya, A. and R. Sargent, 1994. A logic-based foundation of discrete event modeling and simulation. *ACM Transactions on Modeling and Computer Simulation* 4:3-51.

Rickel, J., and P. Porter. 1994. Automated modeling for answering prediction questions: selecting the time scale and system boundary. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*. Cambridge, MA: AAAI/MIT Press.

Weld, D. 1986. The use of aggregation in causal simulation. *Artificial Intelligence* 30:1-34.

Weld, D. 1992. Reasoning about model accuracy. *Artificial Intelligence* 56:255-300.

Zeigler, B. 1976. *Theory of modeling and simulation.* New York, New York: Wiley and Sons.

Zeigler, B. 1984. *Multifaceted modeling and discrete event simulation.* San Diego, California: Academic Press.

AUTHOR BIOGRAPHY

FREDERICK FRANTZ is a Department Manager with Computer Sciences Corporation's Integrated Systems Division. He received his M.S. degree in Computer and Information Sciences from Syracuse University, and a B.S. in Mathematics from Bucknell University. His research interests include the process of model development and reuse and the synergism of modeling and simulation, artificial intelligence, and software engineering. He is a member of ACM and ACM Special Interest Groups in Simulation, AI, and Software Engineering.