Proceedings of the 1992 Winter Simulation Conference
ed. J. J. Swain, D. Goldsman, R. C. Crain, and J. R. Wilson

DISCRETE EVENT SIMULATION MODELING: DIRECTIONS FOR THE '90s

Chair
Ashvin Radiya
Computer Science Department
The Wichita State University
Wichita, KS 67260, U.S.A.

Panelists
Paul A. Fishwick
Richard E. Nance
Jeff Rothenberg
Robert G. Sargent

Fresco's Discovery: If you knew what you were doing you'd probably be bored.

1 INTRODUCTION

This panel considers some of the important unresolved topics ("Grand Challenges"?) in discrete event modeling and simulation that need to be addressed in the '90s. In a panel like this, there are uncompromising tradeoffs in the breadth versus depth of the topics that can be covered. We have opted to cover more depth and yet not overly compromise on breadth with the assumption that there is little benefit from a "watered-down" panel and with the hope that future panels will cover the topics not considered here. Our approach for striking a good balance has been to ask each of the five panelists to describe one or two topics of his own choice which are different from others and to require that the topics be directly related to his research or expertise. The description of each topic essentially contains motivation, brief introduction, major research issues, and references for those who desire to explore the topic further. As expected, all the issues within a particular topic and their interrelationships have not been completely identified. Hopefully through the interaction among panelists and audience participation, a better understanding of the topics will emerge.

The panel may seem to be a cacophony because each panelist is independently playing his own little favorite tune. Surprisingly, there has been considerable commonality in the issues which gives the panel its own melody. Some of the common issues are (1) integration with other fields including artificial intelligence, software engineering, distributed computing and real-time systems, (2) representational issues — multilevel abstractions, symbolic knowledge representation, and formalization of temporal and causal relationships, and (3) reasoning issues — more ways to reason with models than just by predictive queries which are answered by simulation. These and other issues such as dialogue driven model specification, multimodeling, and automated discovery of discrete event models are considered in detail by the panel. Many other important topics such as stochastic aspects of modeling and simulation, parallel and distributed simulation, and graphical/visual approaches to modeling are only briefly touched upon, if at all, and can be a part of the future panels.

In the following, topics are listed in the alphabetical order of the panelists' names.

2 INTEGRATING MODELING IN SIMULATION, SOFTWARE ENGINEERING AND AI by PAUL A. FISHWICK

All of a sudden, everyone is doing simulation! This quote may be a slight exaggeration; however, there are a number of system modeling efforts underway in other disciplines that closely resemble efforts in simulation. These other areas can benefit greatly from the simulation literature, and conversely, the simulation discipline can benefit from increased interaction with these other disciplines. Consider software engineering as an example. One of the key thrusts of the next decade is toward object-oriented design and programming. The object approach stresses the creation and reuse of objects containing both code and data. In this approach, one creates an object model and proceeds to capture the semantics through the use of state transition and data flow diagrams. Software engineers realize that it is not enough to simply "dive into the code" when constructing large programs; one must proceed more gradually through the practice of "modeling" [Nance 1988, Balci and Nance 1988]. Why are software engineers suddenly interested in modeling? A brief scan of some recent software engineering texts [Rumbaugh, et al 1991, Booch 1991] shows that many of the examples given are real world examples such as automated teller machines, air conditioners and digital watches. Isn't this the *very stuff* of simulation? There are, perhaps, several reasons why this convergence has taken place; however, one reason stands out in the form of a general area — distributed computing and real-time systems. Computers are no longer in the form of huge mainframes; instead, computing elements are scattered literally everywhere, and this means that software must be designed, tested and executed in many different locations. The world of software is gradually becoming a mirror for the physical world. Also, it has become clear in software engineering that more effort must be placed in the early phases of the software process where one takes elements from the problem domain and matches them to the world of software. This is where the emphasis on object oriented techniques comes into play — objects in the real world (and object interactions) are most naturally mapped to software that incorporates "object-like" constructs [Zeigler 1990].

Artificial intelligence (AI) can be defined as "formalizing common sense knowledge." From a simulation perspective, AI comes into play when our simulation models include autonomous agents or "intelligent objects." While discrete event simulations are processing agents in queues, we must use AI methods to formulate mental models of how these agents will reason about their environments and react to external stimuli. It is not always sufficient to model humans using simple statistical approximations; often the success of a complex model incorporating human interaction will be predicated on the validation of the mental (not physical) models. Intelligent objects not only carry out preordained sequences, they also dynamically formulate plans and maintain knowledge bases of their beliefs. It may not be necessary for future simulations to include detailed, unique models of every individual; however, generic mental models for a class of intelligent object (teams, groups, crowds) will be useful.

A common thread between the work of software engineering and AI, in terms of modeling, is the distinction between declarative models and functional models. In AI, there has been considerable work in declarative representations —especially with regard to knowledge representations using logic— while functional approaches using actors have also been explored. In software engineering, the object oriented approach involves two fundamentally different types of model technique: functional (data flow) and declarative (state transition). We believe that a similar dichotomy of models can be expressed within simulation [Fishwick 1992]. That is, many simulation modeling methods fall into one of two basic categories: functional or declarative. In the past, there has been much emphasis in simulation on model execution and analysis of simulation results. These two activities are essential to any good simulation task; however, we must not forget the creative act of "modeling." When the literature of systems theory/science and simulation is studied, we find that simulation has much to offer with regard to modeling techniques —especially with regard to clear, formal specifications for dynamical systems. However, if simulation modeling is to progress substantially, we will need to better integrate with modeling approaches in software engineering and AI.

References

Nance, R. E. 1988. "Modeling and Programming: An Evolutionary Convergence", (Unpublished overheads requested from author), April.

Balci and Nance 1992, see section 4.

Rumbaugh J., M. Blaha, W. Premerlani, F. Eddy and W. Lorenson. 1991. "Object-Oriented Modeling and Design", Prentice Hall.

Booch, G. 1991. "Object Oriented Design", Benjamin Cummings.

Zeigler, B. P. 1990. "Object Oriented Simulation with Hierarchical, Modular Models: Intelligent Agents and Endomorphic Systems", Academic Press.

Fishwick, P. A. 1992. "An Integrated Approach to System Modelling using a Synthesis of Artificial Intelligence, Software Engineering and Simulation Methodologies." Submitted for review to ACM TOMACS, March.

3 MULTIMODELING: PROVIDING ANSWERS TO MORE QUESTIONS by PAUL A. FISHWICK

When we create a model, we are interested in answering a class or set of questions at some level of abstraction. For instance, the classic discrete event simulation of a barbershop queuing model (along with the associated probability distributions describing arrival and service rates) exists to answer questions such as "How frequently is the barber busy cutting hair?" This model, however, is inadequate for answering the question "How long does it take the chair to fall back to the ground position when the barber activates the chair control?" Even though these two questions are quite different, and may be asked by different sorts of people, there is a growing need to simulate large, complex systems for use by lots of people. Just as product design and manufacturing phases are becoming better integrated in CAD/CAM technology, system models of physical scenarios must also become more integrated. If a model answers a very select group of questions, then it is brittle in the same sense that a physical tool is limited if it performs only one function. Analysts and users of future simulation models want to be able to ask more types of questions of their systems. Not only do models need to answer questions at different abstraction levels, but future simulation models should be able to answer more than simply predictive questions [Rothenberg 1989a]. Specifically, diagnostic inquiries and sophisticated explanation facilities must also be supported. For example, it should be possible to answer barbershop system questions such as "What was the reason for Joe (the 5th customer) to take so long in the barber chair?" whose answer might be "Because he was preempted by a phone call to the barber which occurs, on average, once every 30 minutes."

When considering models that require discrete event simulation, we are choosing a fairly high level of abstraction at which to simulate a model. The use of the term "discrete event" implies that there is an underlying, more complex process whose state space is continuous – whether or not we choose to explicitly represent this lower level model. Models that are explicit about having discrete and continuous sub-models have been traditionally called "combined models" in the literature [Pritsker 1974, Kreutzer 1986]. Combined modeling is the process of taking two different model types and combining them together to form a more comprehensive model. Most often, combined models have referred to a combination of discrete event and continuous models to form a hybrid model where two distinct forms of simulation are necessary. For instance, consider a grocery store cashier (the discrete event part) combined with a block model of the grocery package belt dynamics (the continuous part). Some recent work [Fishwick 1991, Fishwick and Zeigler 1992] suggests that traditional combined models are just one type of a larger class of model which we term a "multimodel" (after Oren's [Oren 1987] definition of the term). Specifically, we specify that a multimodel is a tightly coupled network of models that are connected together via behavior-preserving homomorphic mappings. The key to successful multimodeling is to partition system components (such as state, event and input spaces) and then to create higher level models by labeling these partitions as new, lumped components at a more abstract level. The resulting hierarchy of models is available for simulation at each level, making it possible to "reason" about system behavior from a variety of abstraction perspectives. Moreover, greater quantities and types of questions may be asked of a system when a multimodel is employed.

References

Rothenberg 1989a, see section 7.

Pritsker, A. 1974. "The GASP IV Simulation Language", John Wiley and Sons.

Kreutzer, W. 1986. "System Simulation: Programming Styles and Languages", Addison Wesley.

Fishwick, P. A. 1991. "Heterogeneous Decomposition and Inter-Level Coupling for Combined Modeling" In 1991 Winter Simulation Conference, Phoenix, AZ, pp. 1120 - 1128.

Fishwick , P. A. and B. P. Zeigler. 1992. "A Multimodel Methodology for Qualitative Model Engineering". ACM Transactions on Modelling and Computer Simulation, Volume 2, Number 1.

Oren, T. I. 1987. "Simulation: Taxonomy", Systems and Control Encyclopedia, Ed. M. G. Singh, Pergammon Press, 1987, pp. 4411 - 4414.

4 DIALOG DRIVEN MODEL SPECIFICATION by RICHARD E. NANCE

An Introductory Lament

The challenge of expressing concepts in some representational form has attracted the attention of scientists from many different disciplinary domains (e.g. philosophy, linguistics, computer science) and for many years. Conceptual expression or transformation is the central issue in programming languages. Within discrete event simulation, the early recognition of the importance of "world view" differences [Kiviat 1969] contributed to the concern for easing the task of the modeler, who strived to represent the system of study in conformance with language constraints both of semantic and syntactic forms. Clearly, the modeler holds, or should hold, the "expert knowledge" about the system and the objectives in simulating it. But, how can this expert knowledge be extracted without error and efficiently?

For over nine years, the correct, efficient representation of simulation models has been a nagging, if not consuming, problem for me, Osman Balci, Mike Overstreet, Joe Derrick and others in the MDE Research Project at Virginia Tech (see [Balci and Nance, 1992] for a complete bibliography). Thus, I must tell you that the problem is difficult; else, I admit not only to my inadequacies but to those of valued colleagues and friends. Why is the problem so difficult? Answers to this question consume the remainder of this short narrative.

A quick and dirty answer is that representing a three or more dimensional problem with two dimensional media is not possible. The complexities of capturing the behavior of objects, both spatially and temporally, especially in their mutual interactions is simply not achievable. That answer must be rejected because it: (1) admits defeat, (2) ignores technological leaps experienced in the past, (3) casts doubt on the claims of many vendors, or (4) spells the end of my research project. Nevertheless, we must keep before us a constant image of what we are trying to do – represent temporal relationships in a simple, clearly understood manner irrespective of the:

(1) number of objects being described,
(2) degree of interaction among objects,
(3) differences in timing granularity (e.g. one object's clock works in days, another in microseconds).

Most importantly, the criteria of "simple" and "clearly understood" are applied by different persons, with different backgrounds and from vastly different technical cultures.

An Inviting Approach

Years ago I described something called the Simulation Model Specification and Documentation Language (SMSDL) [Nance 1977], noting some of the characteristics of such a language. Almost a decade later, our research group examined the SMSDLs offered as potential candidates and found them all wanting (including the Condition Specification of our own creation [Overstreet and Nance 1985]). At this point we assumed a very objective posture and reached two conclusions:

(1) Loud claims to the contrary, AI techniques were not going to solve the model specification problem in the foreseeable future; after all, knowledge acquisition was the most challenging hurdle for expert systems development.

(2) While the support environment could employ the simulation technique "expert" knowledge, a general modeling task, as opposed to a specific problem domain, must rely on the modeler for the application domain knowledge.

We then phrased our problem as extracting the application domain knowledge required in the modeling task from the modeler. Continuing with our pragmatic approach, we proposed an IA (Intelligence Amplification) approach that was characterized as "dialogue driven specification" [Barger 1986, 36-46].

Dialogue driven specification recognizes five levels of dialogue for model specification, shown in Table 1 [Barger 1986, p. 38]. Note that the levels represent progressively less abstract representations, as the model description moves from level 1 to level 5. An internal model is used in a comparative fashion so that the dialogue-based specification proceeds through resolution of abstraction to incorporate domain-dependent representation and replace that which is devoid of application information. Figure 1 illustrates this process as it is utilized in a prototype model generator.

Dialogue level	Function	Expected form of user's response	Model representation
1	Direction	• One key response to a menu • or Text with some limited syntactic rule enforcement	----
2	Description	English text, no syntactic rule enforcement	MR1
3	Naming	Text with some limited syntactic rule enforcement	MR2
4	Typing	• One key response to a menu • or Text in a precise syntax if a response is needed	MR3
5	Conditions and Actions	Text in a precise syntax if a response is needed	Condition

Table 1. Dialogue Levels

A Realistic Quasi-Conclusion

While the dialogue driven specification offers the attractive feature of responding to the modeler's perceptions of important characteristics, it has uncovered, or perhaps emphasized, difficulties that cannot be ignored:

(1) Humans differ so much in their approach to describing a system representation that the dialogue-driven model generator must be prepared to add description at any of the five levels and with little consistency in order. The same modeler could bounce from level 5 to level 3 then back to level 4.

(2) While a major emphasis lies in representational capability for describing the system under study, often more difficulty is found in characterizing the influence of the objectives of the study, which are crucial in the abstraction resolution process.

(3) The properties needed for model specification remain to be adequately or successfully embodied in an SMSDL. Some are there, e.g. object-oriented with inheritance, but some are not, e.g. identification of data collection techniques.

Consequently, we are left with a half-hearted

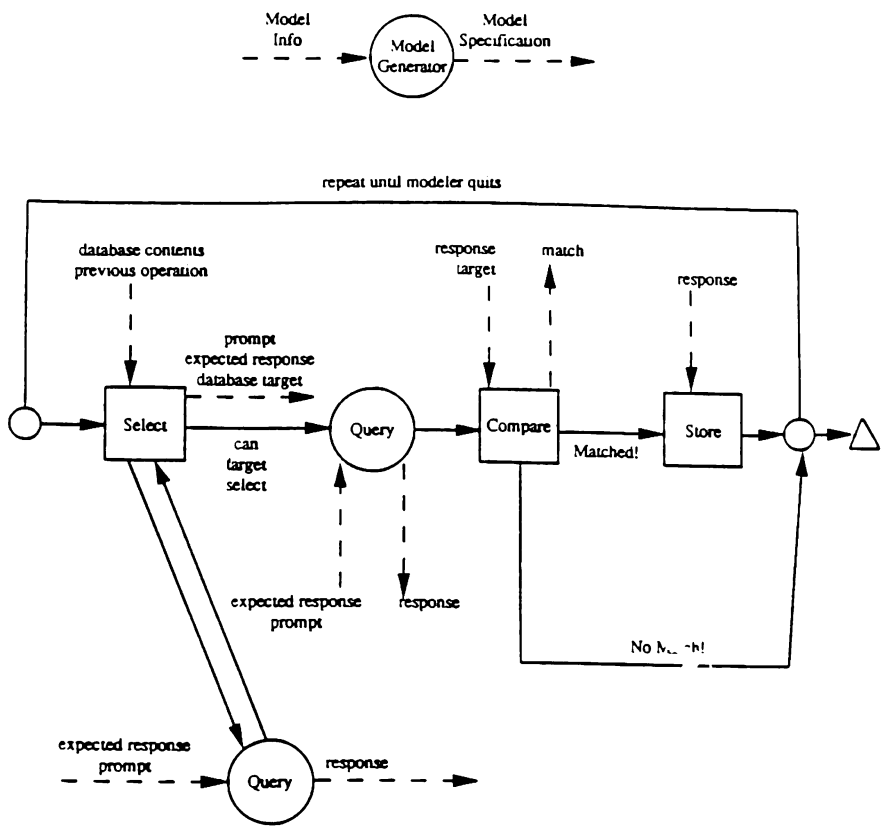

Figure 1. System Function

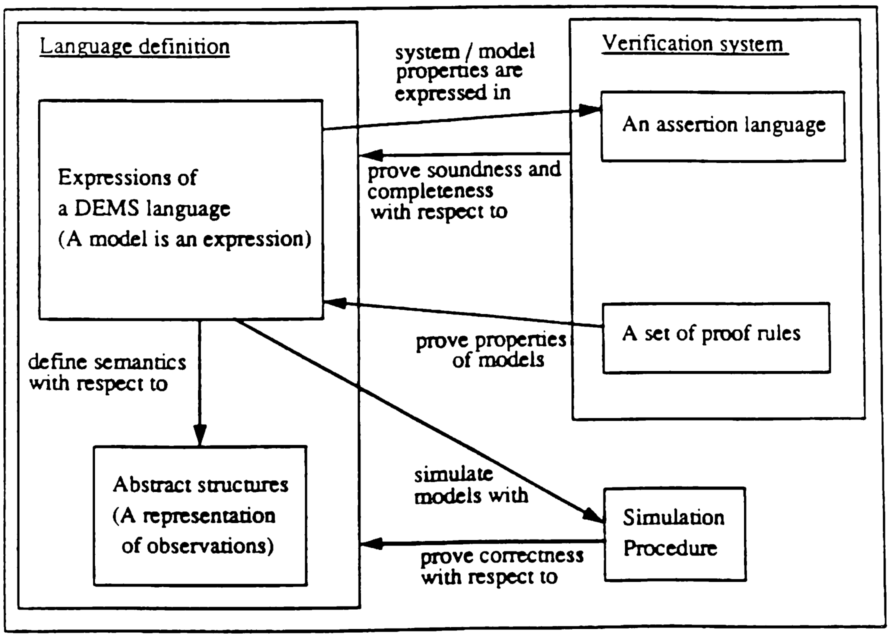
Language definition
Expressions of a DEMS language (A model is an expression)
define semantics with respect to
Abstract structures (A representation of observations)
system / model properties are expressed in
prove soundness and completeness with respect to
prove properties of models
simulate models with
prove correctness with respect to
Verification system
An assertion language
A set of proof rules
Simulation Procedure

Fig. 2. Relationship among expressions of a DEMS language, abstract structures, a simulation procedure, assertion language, and proof rules.

(quasi-) conclusion that dialogue driven specification looks "neat" but, as yet, we lack the understanding to demonstrate its utility. Perhaps some breakthrough awaits. Maybe this conference and even this panel session can identify it.

References

Balci, O. and R. E. Nance. 1992. The simulation model development environment: An overview, Proceedings of the Winter Simulation Conference, Washington, DC, 13-16 December.

Barger, L. F. 1986. The Model Generator: A Tool for Simulation Model Definition, Specification, and Documentation, MS Thesis, Virginia Polytechnic Institute and State University, 9 August.

Kiviat, P. J. 1969. Digital computer simulation: Computer programming languages, RAND Memo RM-5883-PR, RAND Corporation, Santa Monica, CA, January.

Nance, R. E. 1977. The feasibility of and methodology for developing federal documentation standards for simulation models, Final Report to the National Bureau of Standards, Department of Computer Science, Virginia Tech, June.

Overstreet, C. M. and R. E. Nance. 1985. A specification language to assist in analysis of discrete event simulation models, Communications ACM, Vol. 28, No. 2, February, 190-201.

5 A LOGIC-BASED SYSTEM FOR SPECIFYING, EXPRESSING, AND VERIFYING DISCRETE EVENT MODELS by ASHVIN RADIYA

The methodology of Discrete Event Modeling and Simulation (DEMS) can be successfully applied in reasoning about systems only if models correctly represent the intended systems. Currently, intuitive arguments are used to verify that a model correctly represents the intended system. However, there are many critical systems such as nuclear reactors for which it must be "almost" guaranteed that a system is accurately modeled. Hence, it is essential to develop a verification system for proving model properties, e.g., (i) the order of the occurrences of events satisfies certain condition and (ii) certain scenario defined by the values of state variables and event occurrences over an interval lead to an occurrence or nonoccurrence of an event. It is also desirable to prove or disprove a vendor's claim of superior expressivity of a new language, to prove the equivalence of two simulation procedures of the same language, or to ensure that a customer and a simulationist have the same understanding of a system being modeled. Such assertions can be proven in a logic-based system for specifying, expressing, and verifying discrete event models.

The elements of such a logic-based system and their interrelationships as defined in [Radiya 1990] are shown in Figure 2. A DEMS language is formally defined, independent of its simulation procedures, by relating the expressions of the language to abstract structures. Abstract structures represent of behaviors of a system over time intervals. A model is an *expression* in a DEMS language. A simulation procedure simulates a model. A verification system consists of a language, called an assertion language, for *specifying* properties of models/systems and a set of proof rules. Properties of a model expressed in the assertion language are to be *proven (verified)* by applying proof rules. The correctness and scope of the verification system and simulation procedures are established with respect to the definition of the DEMS language.

A logic-based system has been partially developed in [Radiya 1990] by defining the Propositional Discrete Event Logic L_{PDE} [Radiya and Sargent 1992] and Generalized Temporal Logic L_{GT} for expressing models and properties, respectively. L_{PDE} and L_{GT} contain a vast variety of temporal operators (program connectives) including **next**, **if**, **when**, **whenever**, **until**, **while**, and **at**; many of which are either not available in the existing languages or the existing languages permit their usage in a restricted manner. These operators allow one to express temporal and causal relationships and laws of a system in a direct, succinct, and accurate way.

For the logic L_{PDE}, some of the possible research directions are the design, implementation, and computational efficiency of simulation procedures for larger sublogics, and proof systems for proving properties of models. L_{PDE} can also be extended to develop a first order discrete event logic. The logic-based approach can be applied to design DEMS languages based upon other conceptual entities such as objects, inheritance, and rules. It would be mutually beneficial to the field of DEMS and other fields such as artificial intelligence [Galton 1987] and real-time systems [Ostroff 1989, Proceeding of IEEE 1989] to relate L_{PDE} and L_{GT} to other relevant logics. The similarities and differences among these logics and eventual synergism may lead to more expressive languages for DEMS as well as for other fields.

References

Galton, A. (ed). 1987. *Temporal Logics and their*

Applications, Academic Press.

Ostroff, J. S. 1989. *Temporal Logic for Real-time Systems.* John Wiley & Sons, Inc.

Proceedings of the IEEE. 1989. *Special Issue on Dynamics of Discrete Event Systems*, Vol. 77, No. 1, January.

Radiya, A. 1990. A Logical Approach to Discrete Event Modeling and Simulation, PhD Dissertation. School of Computer and Information Science, Syracuse University.

Radiya, A. and R. G. Sargent. 1992. A Logic-based Foundation of Discrete Event Modeling and Simulation, Submitted for publication.

6 AUTOMATED DISCOVERY OF DISCRETE EVENT MODELS by ASHVIN RADIYA

Autonomous agents are playing an increasing important role in many critical activities in space explorations, manufacturing systems, dangerous civilian and defense missions, and traffic control systems. In order for an autonomous agent to function effectively, it is necessary for the agent to discover causality and models of its environments. One important methodology for representing and reasoning with temporal and causal knowledge is Discrete Event (DE) modeling and simulation which allows one to represent certain kind of temporal and causal knowledge of a system - usually as a model - and to reason with this knowledge - usually by performing simulations. Hence, a theory and a system for discovering DE models will (1) make autonomous agents more powerful, (2) create a new testbed for demonstrating the versatility of the methodology of DE modeling, and (3) take us one step closer to making a science of DE modeling by enhancing our understanding of the ways in which DE models are developed.

There are two basic approaches for automated discovery of models. One is a "database discovery method (DDM)" and another is "experimentative discovery method (EDM)". In DDM, the discovery process utilizes a given database of representative histories denoting behaviors of a system over time intervals. In EDM, a model is discovered by performing experiments on a system. Each experiment consists of observing a system for some finite time interval $[t_i, t_f]$ and requires the system to be in a specific state at t_i and some events to occur in $[t_i, t_f]$.

A framework has been developed in [Radiya and Zytkow 1992] for discovering DE models. This includes specification of histories of a system necessary for discovery process, the form of causal and temporal rules that define DE models, and a DDM for discovering a DE model. A history is a record of events, state-changes, and activities that occur over a time interval. The causal and temporal rules of a DE model specify the effects of event occurrences. The effect of an event occurrence at an instant t is to cause state-changes at t and other future event occurrences. The relationship among event occurrences and their effects are expressed in terms of three temporal operators **whenever**, **after**, and **unless** [Radiya 1990]. The formulae to fit these operators are the task of the discovery mechanism of the theory. These temporal relationships suggests a gradual way of partitioning histories. The discovery process utilizes the capabilities of FAHRENHEIT - an empirical discovery system [Zytkow 1987] - to find a set of partial functional relationships among state variables and events, separated by boundaries. The above DDM can be easily modified into an EDM.

With respect to the above framework, some future research directions are methods for discovering more complex temporal relationships, handling symbolic variables, scope and complexity of the proposed methods, introduction of theoretical terms to generate new events and variables, abstractions of propositions, and devices for making observations. Much remains to be done for discovering DE models in the alternative frameworks of flow of entities, progress of processes and activities, and interaction among objects in a system. ([Kodatroff and Michalski 1990] is a good collection of papers on machine learning.)

References

Kodatroff, Y. and R. Michalski. 1990. *Machine Learning*, Vol. I-III, Morgan Kaufman Publications.

Radiya, A. and J. M. Zytkow. 1992. A framework for discovering discrete event models. *Proceedings of the Ninth International Machine Learning Conference*, July 1992, Aberdeen, UK, Morgan Kaufmann Publications.

Radiya 1990, see section 5.

Zytkow, J.M. 1987. Combining many searches in the FAHRENHEIT discovery system, in: *Proceedings the Fourth International Workshop on Machine Learning*, Irvine CA, 281-287.

7 CAUSAL MODELING TO ANSWER BEYOND "WHAT IF...?" QUESTIONS by JEFF ROTHENBERG

Simulationists often view modeling quite narrowly: as a way of making predictions by running an encoded behavioral model. This "toy duck" approach to modeling ("wind it up and let it run") can only answer "What if...?" questions (i.e., "What would happen if...?"). Yet in order to understand and improve real-world systems, it is necessary to do more than simply predict how they will behave: it is equally necessary to understand why things happen the way they do, when (i.e., under what conditions) certain things may happen, and how to produce desired results [Davis et al. 1982, Erickson 1985, Rothenberg 1986]. This requires models that can answer questions such as "Can some result ever occur?", "Under what conditions will some result occur?", "Why would some result occur in a certain situation?", or "How can a desired result be achieved?" Such questions go far beyond the capabilities of traditional simulation. For a number of years, I have referred to these as Beyond "What if...?" questions [Kameny et al 1987, Rothenberg 1988, 1989a, 1989b]. In order to answer such questions, a model must be able to reason (i.e., perform inferences) about itself; this is all but impossible using traditional techniques.

To answer Beyond "What if...?" questions, a model must have a semantics that is rich enough to support the required kinds of inferences [Rothenberg 1989a, 1992, Klahr 1986]. In particular, it is not enough for a model to represent sequences or times of events or activities, as most simulations do; it must represent relationships among occurrences (beyond their temporal order), to support inferences about how and when certain occurrences lead to other occurrences. Furthermore, it is not enough to represent just any relationships among occurrences: the relationships that are represented must be meaningful in ways that support the inferencing necessary to answer questions such as "why", "when", and "how" things happen in the real world (beyond simply answering "what" happens).

Of the possible relationships that can satisfy this requirement, causality has a number of theoretical and pragmatic advantages [Rothenberg 1992]. The implications of causal modeling are subtle but profound. Building models based on causality improves their design, use and interpretation, by: (1) providing a valuable decision criterion for making modeling choices and eliminating artifacts; (2) facilitating the inferencing necessary to answer Beyond "What if...?" questions; and (3) endowing a model with a unique combination of transparency and intuitive cogency.

References

Davis, M., S. Rosenschein, and N. Shapiro. 1982. Prospects and Problems for a General Modeling Methodology, The RAND Corporation, N-1801-RC, June.

Erickson, S. A. 1985. "Fusing AI and Simulation in Military Modeling", in AI Applied to Simulation, Proceedings of the European Conference at the University of Ghent, pp. 140-150.

Rothenberg, J. 1986. "Object-oriented Simulation: Where Do We Go from Here?" In Proc. of the 1986 Winter Simulation Conference, pp. 464-469.

Kameny, I., S. Cammarata, and J. Rothenberg. 1987. "Concept for an Integrated Development Environment for Knowledge-Based Simulations", Proceedings of the IEEE Conference on Systems, Man, and Cybernetics, October.

Rothenberg, J.. 1988. "Knowledge-Based Simulation at RAND", Simuletter, Vol. 19, No. 2, pp. 54-59.

Rothenberg, J., et al. 1989a. Knowledge-Based Simulation: An Interim Report, The RAND Corporation, N-2897-DARPA, July.

Rothenberg, J. 1989b. "The Nature of Modeling", in Artificial Intelligence, Simulation, and Modeling, L. Widman, K. Loparo, and N. Nielsen (eds.), John Wiley & Sons, Inc., August, pp. 75-92.

Klahr, P. 1986. "Expressibility in ROSS: An Object-oriented Simulation System", AI APPLIED TO SIMULATION: Proceedings of the European Conference at the University of Ghent, February 1985, SCS, San Diego, pp. 136-139.

Rothenberg, J. 1992. "Using Causality as the Basis for Dynamic Models", Proceedings of the Third International Conference on Dynamic Modeling of Information Systems, June 9-10, pp. 277-292.

8 REQUIREMENTS OF A MODELLING PARADIGM by ROBERT G. SARGENT

Discrete event simulation modelling is an intellectually intensive and time consuming task and is primarily a creative technical art. There has been modest research to move modelling towards a science; however, considerable research is yet needed. The purpose of this paper is to identify a set of requirements that this author believes a modeling paradigm for discrete event simulation should have.

Prior to presenting the requirements, a few comments will be made. There has been only limited work on theoretical foundations for discrete event simulation modelling. Zeigler, e.g., has developed DEVS based on systems theory. Limited research has been done on modelling environments. There is almost no hierarchical modelling capability. We note that to discrete event simulations on parallel and distributed computers, the models usually have to be specifically developed to include specific informa-tion, e.g. "lookahead" information. Visual/graphic modelling has only received limited attention. Modelling using object-oriented languages has started to receive attention which will perhaps lead to a different world view than the classical world views of process-interaction, activity-scanning, and event-scheduling. We also note that computing cost is rapidly decreasing while people cost is increasing. Perhaps the computer should be doing part of what people are doing even if from the computation point of view it is not efficient.

The requirements that we believe a discrete event simulation model paradigm should have are the following:

GENERAL PURPOSE: the modelling paradigm should allow the modeller to model a wide variety of problem types and domains; it should not be primarily for one type of system, e.g. queueing systems or transaction oriented systems.

THEORETICAL FOUNDATION: A theoretical foundation should underlie a model paradigm if modelling is to be moved towards a science.

HIERARCHICAL CAPABILITY: A modelling paradigm should allow hierarchical modelling so that complex systems can be more easily modelled.

COMPUTER ARCHITECTURE INDEPENDENCE: The model paradigm should be such that a model can be executed on different computer architectures (e.g. sequential, distributed, or parallel, and be able to take advantage of the architecture that it is executing on) and be transparent to the modeller. This requires that such items as "lookahead" information be available from the model itself and not have to be specially added by the modeller.

STRUCTURED: A structured approach that guides the user in model development, including hierarchical modelling, should be part of the model paradigm.

MODEL REUSE: the model paradigm should allow models and submodels to be easily reused and support a model database.

SEPARATION OF MODEL AND EXPERIMENTAL FRAME: Both the model's input and the model's output should be able to be separated from the model itself in the model paradigm.

GRAPHICAL/VISUAL MODELLING: The model paradigm should allow the capability to have graphical/visual modelling.

EASE OF MODELLING: The model paradigm should allow a world view(s) of modelling to be used that is easy to model with.

EASE OF COMMUNICATION: The conceptual model(s) allowed by the model paradigm should be easy to communicate to other parties.

EASE OF MODEL VALIDATION: The model paradigm should support both conceptual and operational validity.

ANIMATION: The model paradigm should allow animation to be accomplished without difficulty.

MODEL DEVELOPMENT ENVIRONMENT: A model development environment can aid in the steps of model development and a model paradigm should allow the use of such an environment.

EFFICIENT TRANSLATION TO EXECUTABLE FORM: The model paradigm should be capable of allowing efficient model translation to executable code. The paradigm should allow the model to automatically be converted to computer code or allow ease of program verification if it does not.

This author believes that some form of encapsulation is required (whether as in object oriented languages or as proposed by Cota and Sargent), that some type of model representation (which allows analysis to be performed on the representation) will be required to provide the information needed for algorithms to execute on different computer architectures (such as in Control Flow Graphs developed by Cota and Sargent), that a modelling language is desirable, and that the implementation will probably be best done in some object-oriented language. No "artificial intelligent" capability was stated as a requirement, although this may be useful, it is not believed by this author to be a requirement for the type of model paradigm he visualizes.

References

Cota, B.A. and R.G. Sargent. Automatic Lookahead Computation for Conservative Distributed Simulation. CASE Center Technical Report 8916, CASE Center, Syracuse University, December 1989.

Cota, B.A. and R.G. Sargent. A New Version of the Process World View for Simulation Modeling. CASE Center Technical Report 9003, CASE Center, Syracuse University, February 1990. (Also the revised version.)

Cota, B.A. and R.G. Sargent. A Framework for Automatic Lookahead Computation in conservative distributed simulations. In Distributed Simulation, ed. D. Nicol, the Society for Computer Simulation, 1990.

Cota, B.A. and R.G. Sargent. Simulation Algorithms for Control Flow Graphs. CASE Center Technical Report 9023, CASE Center, Syracuse University, November 1990.

Cota, B.A. and R.G. Sargent. Control Flow Graphs: A Method of Model Representation for Parallel

Discrete Event Simulation. CASE Center Technical Report 9026, CASE Center, December 1990.

Zeigler 1990, see section 2.

AUTHOR BIOGRAPHIES

PAUL A. FISHWICK is an associate professor in the Department of Computer and Information Sciences at the University of Florida. He received the BS in Mathematics from the Pennsylvania State University, MS in Applied Science from the College of William and Mary, and PhD in Computer and Information Science from the University of Pennsylvania in 1986. He also has six years of industrial/government production and research experience working at Newport News Shipbuilding and Dry Dock Co. (doing CAD/CAM parts definition research) and at NASA Langley Research Center (studying engineering data base models for structural engineering). His research interests are in computer simulation modeling and analysis methods for complex systems. He is a member of IEEE, IEEE Society for Systems, Man and Cybernetics, IEEE Computer Society, The Society for Computer Simulation, ACM and AAAI. Dr. Fishwick was chairman of the IEEE Computer Society technical committee on simulation (TCSIM) for two years (1988-1990) and he is on the editorial boards of several journals including the ACM Transactions on Modeling and Computer Simulation, The Transactions of the Society for Computer Simulation, International Journal of Computer Simulation, and the Journal of Systems Engineering.

RICHARD E. NANCE is the RADM John Adolphus Dahlgren Professor of Computer Science and the Director of the Systems Research Center at Virginia Polytechnic Institute and State University. He received B.S. and M.S. degrees from N.C. State University in 1962 and 1966, and Ph.D. degree from Purdue University in 1968. He has served on the faculties of Southern Methodist University and Virginia Tech, where he was Department Head of Computer Science, 1973-1979. He has held research appointments at the Naval Surface Weapons Center and at the Imperial College of Science and Technology (UK). Within ACM and chaired two special interest groups: Information Retrieval (SIGIR), 1970-71 and Simulation (SIGSIM), 1983-85. He has served as Chair of the External Activities Board, the Outstanding Service Awards Subcommittee, the ad hoc Conference Procedures Committee, and the ad hoc Film Committee that produced *Computers In Your Life.* The author of over 80 papers on discrete event simulation, performance modeling and evaluation, computer networks, and software engineering, Dr. Nance has served on the Editorial Panels of *Communications of the ACM* for research contributions in simulation and statistical computing, 1985-89, as Area Editor for Computational Structures and Techniques of *Operations Research*, 1978-82, and as Department Editor for Simulation, Automation, and Information Systems of *IIE Transactions,*, 1976-81. He served as Area Editor for Simulation, 1987-89 and as a member of the Advisory Board, 1989-92, *ORSA Journal on Computing.* He is the founding Editor-in-Chief of the *ACM Transactions on Modeling and Computer Simulation.* He served as Program Chair for the 1990 Winter Simulation Conference. Dr. Nance received an Exceptional Service Award from the TIMS College on Simulation in 1987. He is a member of Sigma Xi, Alpha Pi Mu, Upsilon Pi Epsilon, ACM, IIE, ORSA, and TIMS.

ASHVIN RADIYA is an assistant professor in the Computer Science Department at the Wichita State University. He received the B. Tech. in Mechanical Engg from Indian Institute of Technology, Bombay and MS and PhD in Computer and Information Science from Syracuse University. His research interests are discrete event modeling and simulation, logical foundation of procedural programming languages, visual programming, animation, and machine learning.

JEFF ROTHENBERG is a Senior Computer Scientist in The RAND Corporation's Social Policy Department. He has worked extensively in knowledge-based simulation and user-oriented design. He performed his graduate work in AI at the University of Wisconsin in the area of semantic nets for robotic applications. His most recent work has been in the development of new modeling formalisms for integrating simulation and planning in order to answer analytic questions that go Beyond "What-if...?".

ROBERT G. SARGENT is a Professor at Syracuse University. He received his education at The University of Michigan. Dr. Sargent has served and continues to serve his profession in numerous ways and has been awarded the TIMS College on Simulation Distinguished Service Award for long-standing exceptional service to the Simulation Community. His research interests include the methodology areas of modelling and discrete event simulation, model validation, and system performance evaluation. Professor Sargent has published widely, is a member of Alpha Pi Mu, New York Academy of Sciences, Sigma Xi, ACM, IIE, ORSA, SCS, and TIMS, and is listed in *Who's Who in America.*