PARALLEL DISCRETE EVENT SIMULATION: A MODELING METHODOLOGICAL PERSPECTIVE

Ernest H. Page
Richard E. Nance

Systems Research Center
and
Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061-0106

ABSTRACT

The field of parallel discrete event simulation is entering a period of self-assessment. Fifteen years of investigation has witnessed great strides in techniques for efficiently executing discrete event simulations on parallel and distributed machines. Still, the discrete event simulation community at large has failed to recognize many of these results. The central question is, why has this occurred? One possible reason is an apparent disagreement in both the focus and objectives of the parallel discrete event simulation research community (primarily computer scientists) and the discrete event simulation community (a widely diverse group including, operations researchers, management scientists, mathematicians, and statisticians, as well as computer scientists). An examination of parallel discrete event simulation from a modeling methodological perspective illustrates some of these differences and reveals potentials for their resolution.

1 INTRODUCTION

In a recent series of articles (Abrams 1993; Bagrodia 1993; Fujimoto 1993a,1993b; Lin 1993; Reynolds 1993; Unger and Cleary 1993) some of the leading researchers in parallel discrete event simulation (PDES) provide their assessment of the field. In the lead article, Fujimoto observes that a central problem for PDES is its failure to "make a significant impact in the general simulation community" during the last 15 years of its development (Fujimoto 1993a, p. 228). Several factors are identified as contributing to this lack of impact, and various approaches are proposed to make parallel simulation "more accessible to the simulation community." The companion articles each identify and bemoan the *acceptance* problem, and while perhaps not reaching a consensus pertaining to the correct resolution of this (and related) problems, a unanimous call is given that discussion and evaluation of the place, role and future of parallel discrete event simulation as a problem solving technique needs to become an important part of PDES literature.

Abrams (1993) suggests that the precipitate cause of *the problem* is a dichotomy in the perspectives and priorities of modeling methodologists – the primary purveyors of discrete event simulation, and parallel programmers – the leaders in parallel discrete event simulation, noting that until these two camps come to terms it is unlikely that PDES will thrive. Fujimoto (1993b, p. 247) acknowledges the correctness of this assertion and labels the potential collaborative attack as a "worthy goal."

This paper is a response to that call, and presents a view of the parallel discrete event simulation problem from a modeling methodological perspective. Of course, such a premise borders on self-aggrandizement; let us stipulate that we make no claim to be the standard-bearers of modeling methodology for discrete event simulation. However, the views expressed herein do come from a background of over 25 years of thinking about the fundamental issues of *how* to do discrete event simulation. Consequently, we hope to provide a somewhat different view of the problem – lending new insights into the current status of PDES, and perhaps shedding some light on the far more important question of where should PDES go from here?

A context for answering this question requires answers to even more fundamental questions. These questions form much of the remainder of the paper

2 WHAT IS SIMULATION?

First and foremost, the question must be asked, what is a simulation? Any number of definitions can be gleaned from a variety of distinguished texts (see Fishman (1973), Franta (1977), Law and Kelton (1991), Shannon (1975)) but for purposes of this discussion we may regard simulation as:

> The use of a mathematical/logical model as an experimental vehicle to answer questions about a referent system.

This definition seems to be efficient in the use of words and careful not to presume certain conditions or implicit purposes. For example, computer simulation is not mandated; the model could follow either discrete event or continuous forms; the answers might not be correct; and the system could exist or be envisioned.

Essentially, a simulation is the basis for making some decision – this decision being based on the "answers" provided by the simulation. The relative importance of the decision, once made, and the subsequent action (or inaction) taken as a result of the decision are myriad. Oftentimes, the simulation provides an assessment of some system which is not readily amenable to other types of analysis, thus the simulation provides the only means by which to assess a given situation. So, the ramifications of making an incorrect simulation-based decision can range from a mere nuisance, to loss of investment, to more catastrophic consequences such as the loss of lives. Accordingly, *arriving at the correct decision is the singular overriding objective of simulation.* One may want a simulation to provide a variety of behaviors and possess a multitude of characteristics, but none of these can be achieved at the expense of a correct decision.

The importance of a correct decision seems to be a fact often overlooked – or at least regarded as unimportant or somehow tangential – in PDES research. Perhaps this perception explains why the big payoff of PDES, the heralds of improved performance, fall largely on deaf ears in the general discrete event simulation (DES) community. To evaluate the benefits of applying parallelism to simulation solely on the basis of typical evaluative measures for parallel programs, such as *speedup* of the runtime, is specious – at best. Such a view casts simulation as merely a piece of code which is run to produce *the* answer. This view is overly simplistic. To gain "general" acceptance, PDES research must examine the methods used to attain execution speed *explicitly* in terms of their relationship to the central objective: arriving at a correct decision.

3 WHAT FACTORS INFLUENCE A CORRECT DECISION?

While computer architecture and compiler design technology have often driven model development in terms of how a simulation model *can* be constructed, modeling methodology has focused on the question of how a simulation model *should* be constructed. Investigation in modeling methodology has persisted some 35 years, beginning with the General Simulation Program of Tocher in 1958 (see Tocher and Owen 1960; Tocher 1979), and continuing in the writings of Lackner (1962, 1964), Kiviat (1963, 1967), Nance (1971, 1981, 1986, 1994), and Zeigler (1976, 1984), to cite the most prominent. The lessons of this history identify several factors that are positively correlated with the probability of making a correct decision. These factors include (but are not limited to):

1. *An adequate understanding of the problem to be solved.* If the problem to be solved is not well-defined and manageable, then little hope exists that a solution to the problem is readily forthcoming. (This is fundamental to every known problem-solving technique and certainly not unique to simulation.)

2. *A correct model.* The correctness of the model is paramount to a cost-effective solution in light of the overall objective. Errors induced in the model, if never detected, could lead to the acceptance of results based on an invalid model. The cost of making this type of error is typically very large. Even if the error *is* detected, if its detection comes late in the development stream, the cost of correction involves the cost of correcting the model and repeating the development steps. To be cost-effective, the methods for model development should foster the initial development of correct models.

3. *A correct program.* Recognizing that the program is but one representation of a model – usually the last in a line of development, a correct program can only be generated from a correct model. The arguments for program correctness mirror those for model correctness.

4. *Experimental design.* Construction of the model and program must reflect the objectives in carrying out the simulation; the right *questions* must be asked of the program in order that the appropriate answers can be derived. The problem understanding must be sufficient and the model and program designed to facilitate the experimental design process.

5. *Interpretation of results.* A key recognition here is that no simulation program ever built produced *the* answer to anything. Typically, simulation output measures are observations of random variables, and a proper understanding of statistical methods, including variance reduction and multivariate analyses, are required to successfully – and correctly – interpret the results provided by a simulation.

These observations confirm that *simulation* involves more than merely a program. Only with careful attention to all of the factors identified above can the overall objective of simulation, a correct decision, be consistently achieved. With this recognition, considerable effort has been made to impose a management structure onto the framework of using simulation as a problem-solving technique.

These efforts are made manifest through life-cycle models for simulation. One such model is illustrated in Figure 1. This life-cycle model of a simulation study is described in detail in (Nance and Balci 1987). Note that the entire structure serves to support one thing: the decision process. Note also the very limited role of the program within the life-cycle model. While programmatical decisions *may* necessarily have impact outside of the program "phase" of the life-cycle model, the impact of program design should not be so pervasive that it is allowed to encumber the other phases of the life-cycle. To allow such an encumbrance, is to ignore 35 years of modeling methodological research, and to place at risk the correctness of the decision that is at the heart of the endeavor.

4 WHAT ROLE SHOULD THE MODEL PLAY?

According to Nance (1983), beginning in the late 1970's a shift in the focus of the discrete event simulation community from a *program*-centric view of the simulation process to a *model*-centric view occurred. Motivating this shift was an evolving recognition that programming language representations of simulation models necessarily contain many implementation-related details that obscure the clear enunciation of model behavior. Furthermore, and potentially the most damaging, the use of a particular language has direct, often hidden, influences on the structure of the model formulation itself. To illustrate this point, consider the classical machine interfence problem in which a technician monitors and repairs a set of machines that fail intermittently. A SIMULA implementation of this model may contain a description of the *lifetime* of a machine class object and a technician class object – a machine operates for some time, fails, waits for repair, and then repeats the cycle; a technician detects a machine failure, travels to the machine and repairs it. A GPSS implementation of this model is very different: defining the technician as a static facility and defining machine failures as transactions which queue for the technician facility. And a SIMSCRIPT implementation of this same model might adopt an entirely different view by describing the model behavior that corresponds to identifiable *events* in the model, such as a machine failure or an end of repair.

This example raises the question: which implementation provides the most "natural" description of the machine interference problem? Obviously no *definitive* answer to this question exists; different people often view the same thing in different ways. Clearly though, a representation (such as GPSS) that characterizes machine failures as "moving" objects does not conform with the physical reality, and so *may* become problematic for model verification and validation.

4.1 How Critical is a "Natural" Model Description?

While no etiology of software errors exists, a consensus among many researchers is that most arise as the result of a poor mesh between the models of problems as they form in the mind (or minds) of a modeler (modeling team), and the representational capabilities provided by extant programming languages and techniques.

The developers of simulation programming languages (SPLs) sought to close this conceptual distance through the provision of a *conceptual framework* (or "world view") within the language. The conceptual framework provides a modeler with a means to construct a *mental picture* of the model. Theoretically, if the model in the

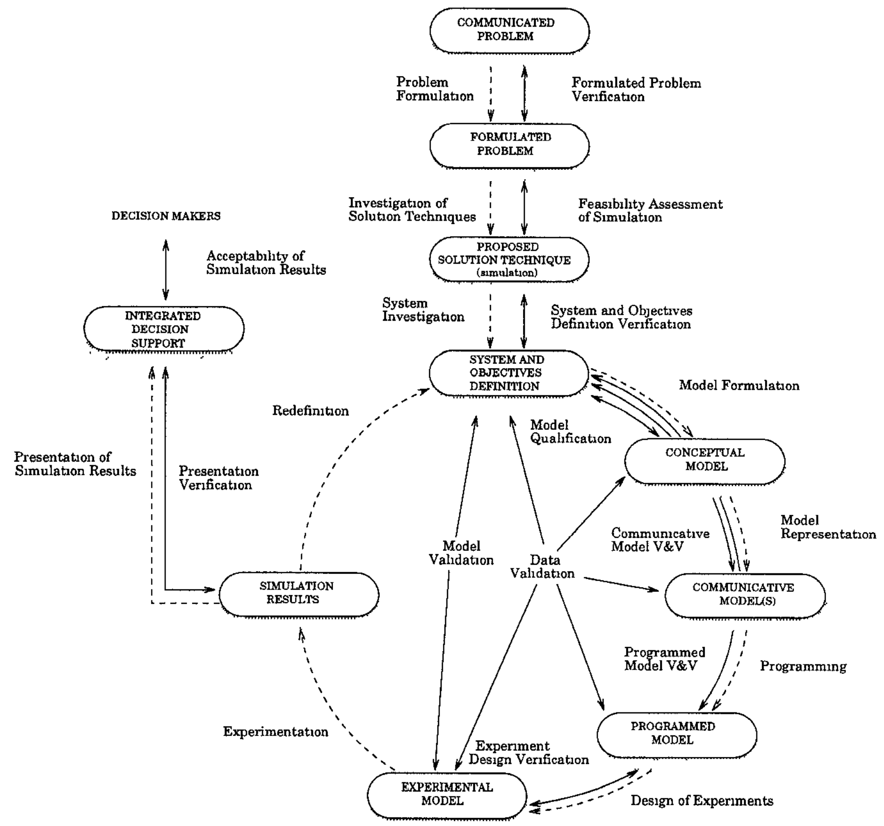
COMMUNICATED PROBLEM
Problem Formulation
Formulated Problem Verification
FORMULATED PROBLEM
Investigation of Solution Techniques
Feasibility Assessment of Simulation
PROPOSED SOLUTION TECHNIQUE (simulation)
DECISION MAKERS
Acceptability of Simulation Results
INTEGRATED DECISION SUPPORT
System Investigation
System and Objectives Definition Verification
SYSTEM AND OBJECTIVES DEFINITION
Model Formulation
Redefinition
Model Qualification
CONCEPTUAL MODEL
Presentation of Simulation Results
Presentation Verification
Communicative Model V&V
Model Representation
Model Validation
Data Validation
SIMULATION RESULTS
COMMUNICATIVE MODEL(S)
Programmed Model V&V
Programming
PROGRAMMED MODEL
Experimentation
Experiment Design Verification
EXPERIMENTAL MODEL
Design of Experiments

Figure 1: A Life Cycle Model of a Simulation Study.

modeler's mind and the SPL utilize the same conceptual framework, the distance is closed.

The traditional conceptual frameworks can best be described as providing a view of a system through varying *localities* (Overstreet 1982, p. 164). The behavior of a system can be modeled according to:

1. the times at which things "happen" (the event scheduling world view; *locality of time*),
2. a state precondition on the occurrence of something happening (the activity scanning world view; *locality of state*), or
3. the ordered sequence of actions performed on (or by) a given model object (the process interaction world view; *locality of object*).

In his thesis, Derrick (1988) classifies thirteen conceptual frameworks and identifies both positive and negative aspects of their influence on model representation.

The provision of conceptual frameworks within simulation programming languages ostensibly affords significant benefits for modeling as compared to general purpose languages. These conceptual bridges are not a panacea, however; the tendency to use the language best known by the modeler often results in a contrived "fitting" of the natural model description into the form provided by the simulation programming language. An important result from (Derrick 1988) is the identification of the need to *select* a conceptual framework suitable for a particular model and a given set of objectives.

4.2 Do Extant PDES Paradigms Permit "Natural" Model Descriptions?

Unfortunately, the importance and role of the conceptual framework within the model development process has had little recognition in the PDES community. A typical PDES problem description reads:

> We assume the system being modeled consists of some number of physical processes which interact in some manner. The simulation program consists of a collection of logical processes each modeling a single physical process.

Usually, little distinction between *model* and *program* is made, and the paradigms prescribe a singular perspective: a contrived form of process interaction. We characterized this typical "logical process" view as contrived because its definition is motivated more by execution concerns than by an accurate representation of the physical counterpart. Fujimoto (1990, p. 33) notes that this (logical process) view allows application programmers to partition the simulation state variables into a set of disjoint states, and ensure that no "simulator event" accesses more than one state. This partitioning permits "minimal" processor synchronization, and thus has become a de facto standard for PDES paradigms.

We see here the evidence that the requirements of a "process-oriented" implementation are allowed to dictate the conceptualization of the model itself. Clearly, any requisite perspective is counter to precepts of conceptual framework selection, and further, would seem to violate a fundamental principle of software development – the principle of *separation of concerns* – advanced by Edsger Dijkstra nearly twenty years ago (Dijkstra 1976, p. 203).

The conceptual restrictions prevalent within PDES, as well as a penchant for working from a program view, result in model representations (the programs) that are often contrived and unnatural descriptions of the system being modeled. The program is typically the only representation provided in extant PDES paradigms. Execution requirements force model perturbations unrelated to the study objectives and the natural system description that exists in the mind of the modeler; the problems of model verification and validation as well as those of life-cycle support become greatly exacerbated. Again, for PDES to realize the desired level of acceptance in the DES community, performance gains must be achievable without sacrificing, or ignoring, other software quality objectives.

4.3 On the Role of Methodology

Nance and Arthur (1988) discuss the influence of modeling methodology on the structure of environments for model development. The authors indicate that the role of a methodology is to identify those *principles, e.g.* life-cycle verification and specification-derived documentation, that should govern the modeling process so that a given set of *objectives* can be attained.

The intent of a modeling methodology is to define a process by which factors inherent in the task at hand, *e.g.* the number of objects comprising the model, the frequency of interactions among them, and the degree of concurrency, can be overcome. The value of a methodology is derived from its ability to produce a product (a model) that exhibits validity (correctness) in conformance with the tolerance level prescribed for the study.

5 WHY IS DISCRETE EVENT SIMULATION UNIQUE?

Projects in discrete event simulation possess several characteristics that distinguish them from most software-intensive efforts. We consider these as they relate to the potential applicability of parallelism.

5.1 Time

The first distinguishing characteristic is the presence of the indexing variable *time* which clearly establishes an ordering of behavioral events in the referent system and delimits the potential for parallel execution of the proposed model. This characteristic is that which has placed DES in the forefront of challenge for those interested in parallel computation.

5.2 Correctness

A second characteristic, as already noted, is the very constrained interpretation of correctness as a project objective. Correctness – it should be duly noted – is but one of several software engineering objectives; reliability, maintainability, testability are examples of others. However, without a doubt, correctness assumes a high priority in more simulation projects than any of the other objectives. How useful is a highly portable simulation program if questions remain concerning its validity? As a result, validation techniques have received far more attention by DES researchers than by those working in software engineering. The existence of the referent system in most projects and the insistence that model behavior reflect system behavior within some prescribed tolerance levels forces a definitive statement of correctness that admits no renegotiation of requirements. A consequent relaxation of the tolerance levels occurs with a clear admission of deficiency.

5.3 Computational Intensiveness

The importance of execution efficiency persists as a third distinctive characteristic. While model development costs are considerable, as is the human effort throughout the period of operation and use, the necessity for repetitive sample generation for statistical analysis and the testing of numerous alternatives forces concerns for execution efficiency that are seen in few software-intensive projects.

While the need for execution efficiency spotlights the potential contribution of parallel execution, a focus on that to the exclusion of the second characteristic is to commit a form of the Type 3 error. Type 3 error is that of solving the wrong problem. Here, the precise statement is that *simulation using a highly efficient, incorrect model represents a non-solution to any problem.* Consequently, the parallel simulation community needs to overcome its preoccupation with efficiency *to the exclusion of other issues*, or it risks the commission of Type 3 error which could lead to the exile of its findings to the domain of irrelevant results.

5.4 Varied Approaches

A fourth characteristic of simulation is the manner of its use; no *typical* use for simulation can be described. Discrete event simulation models may adopt myriad forms:

- A single, large, relatively static model that serves over a protracted period of use, *e.g.* a weather simulation.
- A single model which evolves rapidly during experimentation for system design or optimization, *e.g.* a cache model.
- A model which consists of a concatenation of several existing models in an effort to answer questions on a *metasystem* level.
- Models used for the purpose of analysis.
- Models used for the purpose of interactive training.
- Models used for real time decision support.
- Models targeted to provide various combinations of the above.

Furthermore these models may be developed by groups distributed throughout a company, or across continents. Model analysis may be the purview of an entirely separate group, or groups. And the model users may represent yet another diverse collective. Clearly, each of these uses and development paths have unique criteria that determine the acceptability of the simulation. Thus methods to apply parallelism must be reconciled with a given approach.

6 RECOMMENDATIONS

In the previous sections we have described what discrete event simulation *is* and *how* the technique is applied. The discussion reflects the view of DES as a model-centered problem solving technique. This view is not exclusive; indeed others are possible, and the prevalent treatment of DES by the parallel discrete event simulation community would seem to offer an example of one such difference.

We make the case, then, that the view of discrete event simulation presented here largely reflects that of the "mainstream" of DES. If PDES seeks to make a broader impact within the general DES community, then the methods of PDES *must* be resolved with these views. This resolution may be accomplished any number of ways; no single "silver bullet" is evident. Nonetheless, the following appear an obvious starting point:

1. *Focus on the model.* The move from a program-view to a model-view has been documented. Many methodologies and environments for DES provide programming language independent forms for model description. Even those approaches that advocate a "program-as-model" description provide architecture independence: the same program is executable on machines with disparate architectures, so long as each machine has the requisite compiler. Without question, *any*

approach to discrete event simulation that ties the modeling task directly to the implementing architecture is doomed to failure.

2. *Consider the simulation study objectives.* Many published efforts in PDES describe how speedup can be achieved by *changing the model.* The degree to which this is acceptable is a direct function of the objectives of the study. Rarely are these objectives stipulated, however, and the reader is often left with questions concerning the validity of the resulting program, and by implication, the viability of the approach. With very few exceptions, every paper in which a simulation is involved should include a set of study objectives. Changes to the model made in an effort to exact speedup must then be addressed in terms of their effect on model validity. Some popular PDES benchmarks such as Colliding Pucks and Shark's World are unsatisfactory in this regard.

3. *Examine the methods in terms of a given simulation approach.* If the simulation is used in an evolutionary form to design or optimize a system, then *a posteriori* analysis and modification of code to obtain speedup is likely infeasible.

4. *Consider the relationship of speedup to software quality.* How does the method used to exact speedup affect the model in terms of its: maintainability, correctness, reusability, testability, reliability, portability, adaptability? To what extent are these objectives enhanced or sacrificed in a given approach?

7 CONCLUDING SUMMARY

Stimulated by publication of recent concerns that PDES is having little impact within the general discrete event simulation community, we begin this piece with an assessment of PDES research as consumed with program execution and showing little knowledge or appreciation of fundamental issues in model representation, validation and verification, and statistical analysis of simulation results. The overriding goal of decision support appears to be lost in the quest for speedup. The central question is posed: Where does PDES go from here?

An answer to this questions presupposes a destination point. Our view is that PDES can – and should – play a significant role; the need for computational efficiency is a distinctive characteristic of simulation software. However, the path to this destination requires a recognition of the larger map formed by the DES research community. Working within recognition of the mainstream of DES research places PDES in a contributive posture. Continuing the preoccupation with execution efficiency to the exclusion of constraining and overriding issues leave the PDES research community navigating the curvy – and perhaps enjoyable – back roads, some distance from the interstate. Such activities can be rewarding, but the rewards stem from the drive, not from reaching the destination.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous referees for their valuable comments. Thanks also to Mike Overstreet and Osman Balci for the many discussions which have contributed to the content of this paper.

REFERENCES

Abrams, M. (1993). "Parallel Discrete Event Simulation: Fact or Fiction?" *ORSA Journal on Computing,* **5**(3), pp. 231-233.

Bagrodia, R. (1993). "A Survival Guide for Parallel Simulation," *ORSA Journal on Computing,* **5**(3), pp. 234-235.

Derrick, E.J. (1988). "Conceptual Frameworks for Discrete Event Simulation Modeling," M.S. Thesis, Department of Computer Science, Virginia Tech, Blacksburg, VA, August.

Dijkstra, E. (1976). *A Discipline of Programming,* Prentice-Hall, Englewood Cliffs, NJ.

Fishman, G.S. (1973). *Concepts and Methods in Discrete Event Digital Simulation,* John Wiley and Sons, New York.

Franta, W.R. (1977). *Process View of Simulation,* American Elsevier.

Fujimoto, R.M. (1990). "Parallel Discrete Event Simulation," *Communications of the ACM,* **33**(10), October, pp. 31-53.

Fujimoto, R M. (1993a). "Parallel Discrete Event Simulation: Will the Field Survive?" *ORSA Journal on Computing,* **5**(3), pp. 213-230.

Fujimoto, R.M. (1993b). "Future Directions in Parallel Simulation Research," *ORSA Journal on Computing,* **5**(3), pp. 245-248.

Kiviat, P.J. (1963). "Introduction to Digital Simulation," Applied Research Laboratory 90.17-019(1), United States Steel Corporation, April 15 (stamped date).

Kiviat, P.J. (1967). "Digital Computer Simulation: Modeling Concepts," RAND Memo RM-5378-PR, RAND Corporation, Santa Monica, CA, January.

Lackner, M.R. (1962). "Toward a General Simulation Capability," In: *Proceedings of the AFIPS Spring Joint Computer Conference,* pp. 1-14, San Francisco, CA, May 1-3.

Lackner, M.R. (1964). "Digital Simulation and System Theory," System Development Corporation, Santa Monica, CA.

Law, A.M. and Kelton, W.D. (1991). *Simulation Modeling and Analysis,* Second Edition, McGraw-Hill, New York.

Lin, Y-B. (1993). "Will Parallel Simulation Research Survive?" *ORSA Journal on Computing,* **5**(3), pp. 236-238.

Nance, R.E. (1971). "On Time Flow Mechanisms for Discrete Event Simulation," *Management Science,* **18**(1), pp. 59-93, September.

Nance, R.E. (1981). "The Time and State Relationships in Simulation Modeling," *Communications of the ACM,* **24**(4), pp. 173-179, April.

Nance, R.E. (1983). "A Tutorial View of Simulation Model Development," In: *Proceedings of the 1983 Winter Simulation Conference,* pp. 325-331, Arlington, VA, December 12-14.

Nance, R.E. (1986). "The Conical Methodology: A Framework for Simulation Model Development," Technical Report SRC-87-002, (TR-87-8), Systems Research Center and Department of Computer Science, Virginia Tech, Blacksburg, VA, December.

Nance, R.E. (1994). "The Conical Methodology and the Evolution of Simulation Model Development," *Annals of Operations Research,* Special Volume on Simulation and Modeling, O. Balci, (Ed.), To appear.

Nance, R.E. and Arthur, J.D. (1988). "The Methodology Roles in the Realization of a Model Development Environment," In: *Proceedings of the 1988 Winter Simulation Conference,* pp. 220-225, San Diego, CA, December 12-14.

Nance, R.E. and Balci, O. (1987). "Simulation Model Management Objectives and Requirements," In: *Systems and Control Encyclopedia: Theory, Technology, Applications,* M.G. Singh (Ed.), Pergamon Press, Oxford, pp. 4328-4333.

Overstreet, C.M. (1982). "Model Specification and Analysis for Discrete Event Simulation," PhD Dissertation, Department of Computer Science, Virginia Tech, Blacksburg, VA, December.

Tocher, K.D. and Owen, D.G. (1960). "The Automatic Programming of Simulations," In: *Proceedings of the Second International Conference on Operational Research*, pp. 50-68.

Tocher, K.D. (1979). Keynote Address, In: *Proceedings of the 1979 Winter Simulation Conference*, pp. 640-654, San Diego, CA, December 3-5.

Reynolds, P.F., Jr. (1993). "The Silver Bullet," *ORSA Journal on Computing*, **5**(3), pp. 239-241.

Shannon, R.E. (1975). *Systems Simulation: The Art and Science*, Prentice-Hall, Englewood Cliffs, NJ.

Unger, B.W. and Cleary, J.G. (1993). "Practical Parallel Discrete Event Simulation," *ORSA Journal on Computing*, **5**(3), pp. 242-244.

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

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

AUTHOR BIOGRAPHIES

ERNEST H. PAGE is a Research Associate with the Systems Research Center and a Ph.D. candidate in the Department of Computer Science at Virginia Polytechnic Institute and State University (VPI&SU). He received B.S. and M.S. degrees in Computer Science from VPI&SU in 1988 and 1990. During 1993, Mr. Page served as an Instructor in the Department of Computer Science at Radford University. He has been the Chairman of the ACM Student Chapter at VPI&SU, 1988-89, and held committee positions for the Virginia Computer Users Conference, 1990-92. Since 1990, Mr. Page has served as a referee in the area of discrete event simulation for several technical journals including, *ACM Transactions on Modeling and Computer Simulation, Annals of Operations Research*, and the *Journal of Parallel and Distributed Simulation*. His research interests include discrete event simulation, parallel and distributed systems, and software engineering. He is a member of ACM, ACM SIGSIM, IEEE CS, SCS, and Upsilon Pi Epsilon.

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 (VPI&SU). He received B.S. and M.S. degrees from N.C. State University in 1962 and 1966, and the Ph.D. degree from Purdue University in 1968. He as served on the faculties of Southern Methodist University and VPI&SU, where he was Department Head of Computer Science, 1973-1979. Dr. Nance has held research appointments at the Naval Surface Weapons Center and at the Imperial College of Science and Technology (UK). Within ACM, he has 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 and several ACM committees. The author of over 100 papers on discrete event simulation, performance modeling and evaluation, computer networks, and software engineering, Dr. Nance has served on the Editorial Panel 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 a Distinguished 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.