Proceedings of the 2001 Winter Simulation Conference
B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, eds.

SIMULATION ENVIRONMENT FOR THE NEW MILLENNIUM (PANEL)

Chair:
Voratas Kachitvichyanukul

School of Advanced Technologies
Asian Institute of Technology
P.O. Box 4
Bangkok, THAILAND

Panelists:

James O. Henriksen

Wolverine Software Corporation
4115 Annandale Road
Annandale, VA 22003-2653, U.S.A.

C. Dennis Pegden

Rockwell Software Inc.
504 Beaver Street
Sewickley, PA 15143, U.S.A.

Ricki G. Ingalls

Oklahoma State University
Stillwater, OK 74078, U.S.A.

Bruce W. Schmeiser

Purdue University
1287 Grissom Hall
West Lafayette, IN 47907-1287, U.S.A.

1 INTRODUCTION

In the late 80's, a session was organized on "Simulation Environment of the 1990's". Fourteen years have passed. Where are we now with respect to the predictions made the last time? Can we make new predictions, now that new hardware and software are ever more powerful? Are we any closer to where we wanted to be with methodologies, tools, etc.?

Looking back at the paper for the session "Simulation Environment of the 1990's" in the Proceedings of the 1987 Winter Simulation Conference, many of the questions posted then are still valid since many of the issues are still unresolved. Figure 1 below is taken from Kachitvichyanukul et.al.(1987) as a reference point. Some of the questions for the panelists to addressed are listed as follows:

- The concept of what constitutes a simulation environment is growing faster than the ability of any one company to keep up.
- The addition of gee-whiz features has taken precedence over addressing architectural shortcomings.
- There's a disproportionate degree of interest in web-based simulation technology in academia and the military/government. It is rare to find anyone in the real world who is interested in web-based simulation.
- A simulation environment also should support the model analysis. In addition, modeling can be partitioned into the logical model and the input model. Most simulation software focuses on the logical model.
- Not enough efforts are spent on development of tools to support model analysis. Many such tools are published and known for quite sometime but no commercial firm seems to be interested in including them in the software.
- Most of the new features added in new software tools are "gee whiz" items that do not improve basic simulation methodology.
- Not everybody needs an integrated system, a toolbox approach is more appropriate.

2 POSITION STATEMENT BY JAMES O. HENRIKSEN

Back in the late 1980s, I wrote a paper entitled "Simulation Software of the 1990s: The Integrated Simulation Environment." In the dozen or so years since writing that paper, I've changed my mind on some issues and become even more assured that I was right on others. In the paragraphs that follow, I'll present an updated position statement on these issues.

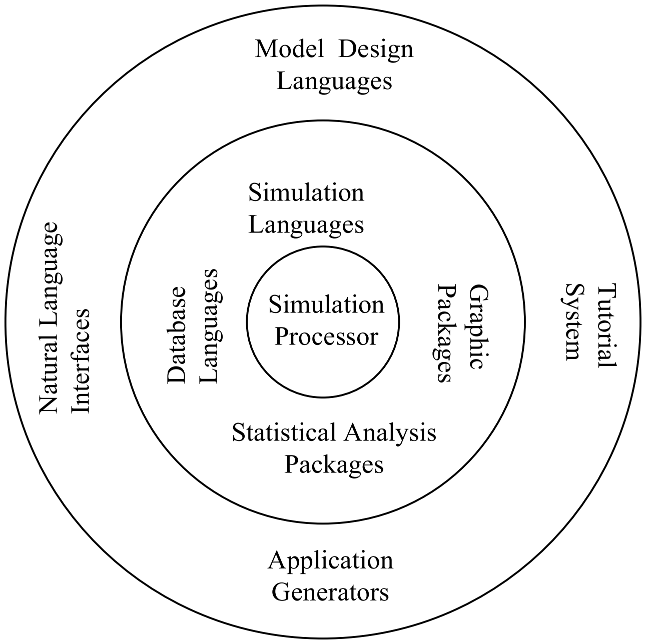

Figure 1: Components of Simulation Environment.

First of all, we are guaranteed that progress will never be orderly. Software developers are subject to a wide range of pressures, the greatest of which is to earn enough money to survive. Consequently, features that offer the greatest chance of earning money will always receive the highest priority. Note that this not the same as saying features with the broadest appeal will always receive the highest priority, because there are things that many people would like to have, but few would be willing to pay for.

In my earlier paper, I envisioned the emergence of integrated simulation environments, i.e., suites of tools all provided by a single vendor. Over time, the simulation community's vision of what such an environment should include has grown to include more and more capabilities. I believe that we have long since past the point at which a single vendor could "do it all." Over the last ten years, we've seen the emergence of firms that specialize in narrowly defined areas of simulation, e.g., distribution fitting, optimization, etc. Furthermore, a number of long-time, "old-line" simulation companies have disappeared as the result of corporate mergers and acquisitions. All of this has created pressures on small companies to focus on what they do well and collaborate or cooperate, at the very least, with other companies in order to provide a broader range of capabilities. Thus we have companies whose distribution-fitting, animation, optimization, and other software is designed to work with software from other vendors. Increasingly, licensing arrangements are entered into allowing one company to package and sell another's software along with its own.

There are those who lament the absence of certain capabilities in simulation software. To them I say "Microeconomic systems are self-correcting." If there were money to be made (or the hope thereof), a specialty firm would arise to meet that need.

In a lot of respects we've made a great deal of progress. For example, the tremendous increase in personal computing power now enables us to easily do things on personal machine that were difficult to do on a large (but shared) mainframe ten years ago. The people providing software to the simulation community are not stupid (at least not all the time). They listen to their customers, and with each new release of their software, new, useful capabilities are added.

One unfortunate consequence of incremental improvement is that, like successive rings around a tree, the increments make it harder and harder to retool core concepts. As a consequence, we have a lot of software that rests on 20-year old implementations. Fortran still casts a long shadow over simulation software. For example, in a lot of simulation software, arrays are still the primary, if not the only data structuring mechanism. In some software, attributes of entities flowing through a system are stored in arrays. It's not unusual to see systems in which all entities flowing through a system carry all attributes that have been defined for any entity. This is because the underlying array implementation dictates using a row index to specify a given entity and a column index to specify the desired attribute. The two disadvantages of this approach are (1) it enables inappropriate attribute references to go undetected; and (2) it greatly reduces scalability. Consider a model of an airport. If a passenger has a "number of suitcases" attribute, and an airplane has a "fuel capacity" attribute, and all entities carry all attributes, then an airplane has a number of suitcases, and a passenger has a fuel capacity. While the former is within reason, the latter most certainly is not. If a model contains only five types of entities, and each entity type has five attributes, the array-based approach is probably adequate; however, if a model has dozens of entity types and each type has dozens of attributes, since the memory required to store all the attributes grows as the product of the numbers of entity types and attributes, this approach scales poorly to large problems.

With respect to reexamination of kernel implementation issues, I practice what I preach. I've spent much of the last five years reexamining simulation implementation foundation issues. Not surprisingly, the results of those efforts are now paying off. Hosting the best of the old ideas in a new, modern framework has proved to be highly beneficial.

Another way in which I've personally reacted to increasing demands for software functionality is to incorporate improved extensibility into the software my company produces. This is "enabling" technology, because it places our users in a position to do for themselves those things we cannot do. One of our customers is a large manufacturing company. They've used our software's customization features to build an assembly line modeling package perfectly tailored to their needs. They once told me about another software vendor who came to visit and essentially said "Tell us what you need, and we'll build it for you." The customer told me, "In the two years it would take them to provide what we need, we will have moved two years downstream." They elegantly summarized: "We're in the business of inventing new paradigms, and you can't invent your own using someone else's."

The jury's still out on some of today's "gee whiz" capabilities. How important will web-based technology be in five years? While there's enormous interest in web-base technology in the academic and government/ military communities, the preponderance of simulation applications are still one-person efforts. No one has a crystal ball. Some will invest heavily and lose. Technology is a cruel master.

Finally, I'd like to discuss the environment in which software developers operate. The complexity of our development environment, and the complexity of the systems on which our software runs, are both growing explosively. I'd like to offer one very mundane, but illustrative example. Consider the issue of software installation procedures for the PC. Microsoft, in its wisdom, has implemented its own Windows Installer. Companies such InstallShield, who used to supply software that performed installations, have had to shift gears and provide software that in essence prepares inputs to be used with the Windows Installer. Some of the third-party tools that we require for use with our products now require the use of the Windows Installer. Thus, not using the Windows Installer is no longer an option for anyone. I won't regale you with all the details of the complexity that Microsoft has introduced into what used to be a fairly straightforward process. Moving to Windows Installer technology cost my company over a man-month's worth of effort. This is just one small example of the explosive growth of complexity.

One of the great strengths of the simulation community has been the fact that it is serviced by small companies whose livelihood depends on providing technology-specific, high-quality service. As the complexity of our development environment increases, the long-term existence of small companies is imperiled. The threshold company size for doing anything is increasing. Over the long haul, this portends the disappearance of small, responsive firms and treatment of simulation software as a commodity rather than a labor of love, by increasingly larger firms. In other words "Microsoft begets Microsoft." This is a real threat to the simulation community.

3 POSITION STATEMENT BY RICKI G. INGALLS

When I was asked to be on this panel, I started to thing on simulation environments, their use, and their evolution over time. After some thought, I brought this down to three simple questions that will form the basics of my position on future simulation environments. The questions are (1) who cares about simulation environments? (2) What are they good for? and (3) if simulation environments changed dramatically, would the commercial use of simulation increase dramatically? The following statement addresses these three questions.

Who cares about simulation environments? The answer to that question seems very straightforward: the person who has to "program" and analyze the simulation. A simulation environment to that type of user is critical to the speed of implementation and the eventual recommendations that will come from the simulation study. It should be clear that most of the work in simulation environments over the last 15 years has been done with this type of person in mind. This person must be at least acquainted with simulation and it capabilities.

However, a second type of person that would care about simulation environments is a programmer that would like to embed simulation capability into a larger system. The problem with this type of person is that the simulation environments of today do not help this issue at all. Most of the environments have OLE or some other type of handshaking that can be accomplished between a simulation program (or function) and a main program, but that handshaking is not straightforward and it certainly is not an integral part of any simulation environment. Perhaps the best simulation environment for this type of person would be an integrated environment with Microsoft Visual Studio, where the simulation looked like any other embedded system.

What are simulation environments good for? The real purpose of a simulation environment should be the quick and efficient execution of a simulation study or embedding simulation in a larger system. Some work can be accomplished for both types of users at this point. For the person performing a simulation study, a simulation environment that does not translate to "code" would be an improvement. Almost all of the simulation packages eventually convert the environment information to simulation code for execution. This problem with this is that the user is not sure what code is being generated and how it is really performing. If the information seen on the user interface were complete, meaning that no additional assumptions were being made behind the scenes, that would be a welcome change.

If simulation environments changed dramatically, would the commercial use of simulation increase dramatically? The two types of users that we have are two different markets for the simulation software. The market that consists of the person who executes a simulation study is limited and, I would argue, saturated. So, if the simulation environment for this market improves, the simulation industry would see limited growth. However, the market that includes our person who wants to embed simulation in larger systems is quite large. It could be anyone working on larger systems. Most of the people in that market are ignorant of simulation and its capabilities. Creating a simulation environment where simulation capabilities can be easily integrated into the primary business software used today would open large markets. The strength of current simulation systems does not lie in the user interface or data analysis capability. It does lie in its ability to model variance. The vendors who have environments where this is easily accomplished will open new markets for simulation as a tool.

4 POSITION STATEMENT BY C. DENNIS PEGDEN

Simulation modeling has become a critical technology for the 21st century. It is used by enterprises throughout the world to improve the design and operation of complex systems.

Simulation technology is in a state of rapid change. The technology is becoming more powerful, easier to use, and useful for an expanding range of applications. However much remains to be done to unlock the full benefits that this technology can bring to the world. The presentation will focus on some of the key challenges facing vendors, customers, and researches in advancing simulation in the new millennium.

4.1 Challenge #1: Expanding Applications

In the past decades the focus within the simulation community has been on making it possible to model a wide range of systems. This has led to the development of very rich and powerful modeling tools. However rich and powerful tools are by their nature complex and difficult to learn. There are still many potential applications of simulation that are passed by because of the complexity of the tools and technology.

What users need are tools that are powerful and flexible, yet very easy to learn and use. Without a doubt, the number one barrier to the broad deployment of simulation technology is the complexity of the technology. Reducing the complexity - while keeping the flexibility to accurately model a wide range of systems - remains the number one challenge from the user to the industry.

During the past 40 years, simulation has been a tool used by a small group of trained experts to model complex and expensive systems. In the future, analyst throughout the enterprise will routinely use this technology. To support this new class of users, the tools will become significantly easier to learn and use.

4.2 Challenge #2: Collaborative Model Building

As the number and size of simulation models increase, there will be new demands placed on simulation tools to make it easier for people to share models across the enterprise, and also collaborate on the development and maintenance of models. The Internet will clearly play a significant role in this evolution. The Internet is changing the entire information technology field, and simulation is no exception. The Internet will play an important part in building and viewing simulation models.

In the future, and enterprise will maintain a knowledge base of their systems, process, and products that can be accessed across the Internet. The processes will be defined in terms of animated, simulation models that can be executed by any individual within the enterprise that has privileges to access the system. Simulation will emerge as the preferred way of documenting and communicating processes within the enterprise.

4.3 Challenge #3: Multipurpose Models

Since the beginning days of simulation, the conventional wisdom has been that a successful simulation begins with a clear statement of the purpose of the model. This point is hammered home in nearly every introductory textbook on simulation methodology. One begins with a statement of the purpose and develops a model to meet that purpose. This statement of purpose includes the specific questions that need to be answered (e.g., predict daily production capacity) and the accuracy required (e.g., within 5%). The stated purpose then drives the level of detail (and the amount of work) that is put into the model.

As we look to the future in simulation, one of the promising ideas is the concept of having pre-built models or model components that can be plugged together to form a model of our system. The idea is that we simply select these components from a library and use them directly. For example, we might build a model of our entire supply chain by simply connecting together pre-built, generic models of our plants, distribution centers, and transportation centers. The goal is to build each model component once, verify its operation, and then make it available in a library to be used in many different applications.

To make this concept work, we need to rethink completely the concept of a purpose-built model. Our generic model components must be built without knowing the specific questions that they will be used to answer. How do we decide on the level of detail to incorporate into these generic models? If we build a highly detailed model of our plant, then it will be useful for accurately predicting our plant system performance, but much too detailed for incorporation into an enterprise-wide supply chain model. On the other hand, if we build a rough-cut capacity model of our plant, it will be useful in our enterprise-wide supply chain model, but useless for predicting detailed plant system performance.

The challenge is to build model components that have multiple levels of fidelity that can be changed by the user based on the purpose of the model. The generic model must include high-level representations as well as detailed representations of the same system. When a model or model component is selected, the user specifies the level of detail required, which causes the appropriate model representation to be used.

4.4 Challenge #4: Expanding Model Scope

The mainstream application for simulation has been in the design and analysis of complex systems. Models have been used to select between competing systems, and to optimize a specific design. However models have the potential to be used in many different ways – including operational scheduling and real-time system control.

In the future our models will be used to help improve performance throughout the life cycle of the system – including both design and execution. This has already begun in manufacturing applications, but will expand dramatically in the future. Although manufacturing design and execution are typically viewed as separate – unrelated activities, both manufacturing design and execution can benefit greatly from a model of the factory. This model must include basic components of the plant such as machines, workers, transport devices, etc. In the future a common model-based framework will exist that can be leveraged across both the design and execution of the manufacturing system. A single model will serve multiple purposes in the life cycle of a factory: including visualization, simulation/animation, hardware emulation/testing, factory scheduling, and real-time factory control.

5 POSITION STATEMENT BY BRUCE SCHMEISER

My interest in simulation centers on stochastic simulation viewed as a statistical experiment. Random variates from a known input model are passed through a known logical model to produce output data, from which a point estimate of a performance measure is calculated; the purpose of the experiment is to determine the value of the performance measure.

With this view, I am excluding other important forms of simulation, including deterministic simulations and stochastic simulations whose purpose is training or general insight. In addition, I am excluding various other interesting issues, such as computer graphics, documentation, verification, and validation.

Within this view, most commercial simulation environments focus almost entirely on the logical model. Impressive gains in logical-model development (especially since the the fall of 1984 when I first saw Dennis Pegden present Cinema at the Dallas ORSA/TIMS meeting) allow non-simulation practitioners to create complex logic models with little formal training, often with drag-and-drop quickness. User-friendly logical-model development is central to having a commercially successful simulation product. Of all components of simulation modeling and analysis, only one---the logical model---is absolutely necessary. Randomness in the exogenous variables can be assumed away, validation can be ignored, documentation never written, but a logical model must be created. Therefore, the focus on the logical model is natural and the incredible improvements not unexpected.

Surrounding the logical model are the input model and the point estimate. The input model, which specifies the distributions from which the exogenous variables are sampled, feeds the logical model. In turn, the logical model creates output data from which the point estimate(s) are computed. Most commercial simulation environments support input modeling only minimally; random-variate generators to produce independent random variates from several classical distributions. Similarly, most commercial simulation environments support analysis of the output data only minimally; usually calculation of the sample mean and sometimes calculation of the standard deviation and sometimes a histogram or empirical cumulative distribution function.

This minimal support for input modeling and output analysis occurs, I think, for two reasons. First, a commercial product needs a large user base and a large user base means that the probability and statistics issues need to be minimized so as to not confuse ill-prepared practitioners. Second, ease-of-use sells, while model and analysis sophistication do not, at least not for the majority of users.

There is one exception. From the early days of commercial simulation languages, wide-spread support has been provided for the use of common random numbers to reduce the variance of point estimators of the difference in performance between two system designs. Multiple random-number streams, the heart of common random number support is so embedded in simulation lore that some (many?) practitioners (incorrectly) think it necessary to use a different random-number stream for "different" types of random variates. The reasons, I think, that the concept of common random numbers is supported is because the variance reduction is often substantial, seldom is the variance increased, additional user effort is minimal, and the user does not need to understand the theory (or even purpose) to incur the benefit.

These reasons stand in sharp contrast to input modeling and output analysis, where even simple efforts at sophistication run into trouble. Suppose, for example, that a simulation environment's input-model capabilities included non-homogeneous Poisson processes, random vectors (composed of dependent non-normal random variables), and time series. Although model validity sometimes (often?) would be improved, other times the added capabilities could scare away users; worse, the additional capabilities could be misused, thereby harming validity.

Similarly, suppose that a simulation environment's output-analysis capabilities included standard-error estimates or confidence intervals to indicate the sampling error present in the point estimates. Although the interpretation of the point estimates would be improved for some users, less-sophisticated users would be confused. Many users simply don't know the difference between the standard error of the point estimator and the standard deviation of the output data. To emphasize the point, notice that most users have seen confidence intervals in a college class room, yet few know how to interpret such intervals. If, for example, a prediction interval were presented instead, few would notice or care.

There is another reason for the lack of probability and statistical support in commercial simulation environments. Those drawn to software design tend to have strong computer-science background. Product structure is designed with substantial care, with much attention to, for example, object orientation or the ability to be web friendly. Random-number and random-variate generation are routines added later; output analysis and variance reduction capabilities are added later or not at all.

There is one other capability that has become popular during the last few years: the ability to seek a good system design by searching over a large set of possible designs. Such a capability is important in that simulation is naturally good at estimating performance of a given design but has no natural ability to find a good design (unlike in the deterministic world where commercial mathematical-programming environments have flourished). Unfortunately, most commercial optimization capabilities are black boxes, with the search logic being kept secret. Because in real-world complex models the optimal design is unknown, a user seldom can know whether a given design is close (or far) from the optimal design.

The primary reason for lack of statistical support is not, I think, because of lack of methodology. During the 1990s I (and others) wrote various articles discussing ideas and algorithms and (non-commercial) software under the assumption that adoption in commercial simulation environments would follow, as it had earlier for random-number and random-variate generation. What quickly comes to mind? Input modeling ideas by Jim Wilson, Barry Nelson, and colleagues; methods for comparing systems by Barry Nelson and Dave Goldsman and colleagues; output-analysis ideas by many authors.

Where can one find discussions of how to incorporate better support for input modeling, output analysis, variance reduction, and comparing systems? I created a quick, but biased, list by stripping some references from my vitae. Schmeiser (1990) discusses the statistical aspects of simulation in general, as do many textbooks. Schmeiser (1992) discussed simulation environments specifically. Schmeiser (1999) and the panel Fox et al. (1990) discussed input-modeling issues; the Glynn et al. (1995) panel discussed output-analysis issues. Goldsman et al. (1991) discussed methods for selecting the best system. Schmeiser and Scott (1991) discuss the SERVO simulation environment, which supports many point estimators, all with standard errors. Schmeiser and Kachitvichyanukul (1986) discuss how to preserve common-random-number when random-variate generation is not via the inverse transformation. Song and Schmeiser (1994) discuss reporting point-estimator precision to non-sophisticated users.

Fourteen years ago, at the time of the 1987 panel discussion, I would have said that the primary obstacle to more-sophisticated support of the probability and statistical issues was lack of methodology. I was insightful enough to understand that the key was to obtain "pretty good" results automatically, rather than the usual goal of the statistically oriented literature to extract maximal information for a data set regardless of computational or practitioner cost. I was naive, however, in understanding that interface design (both graphical beauty and ease of use) is the key to developing a large user base.

In the best of worlds, simulation environments would support all types of users. Can we move closer?

REFERENCES

H. Chen and B. Schmeiser. Stochastic root-finding via retrospective approximation. *IIE Transactions* 33 (2001), 259-275 (Special issue of *Operations Engineering* honoring Alan Pritsker).

B.L. Fox, M.E. Johnson, W.D. Kelton, A.M. Law, B.W. Schmeiser, and J.R. Wilson. Alternative approaches for specifying input distributions and processes (panel). *Proceedings of the 1990 Winter Simulation Conference,* ed. O. Balci, R. Sadowski, and R. Nance, 382-386. Institute of Electrical and Electronics Engineers, Piscataway, NJ, 1990.

P.W. Glynn, J.O. Henriksen, C.D. Pegden, B.W. Schmeiser, and L.W. Schruben. The interface between simulation output analysis research and practice. *Proceedings of the Winter Simulation Conference*, (C. Alexopoulos, K. Kang, D. Goldsman, and W. Lilegdon, eds.), 1995, 346.

D.M. Goldsman, B.L. Nelson, and B. Schmeiser. Methods for selecting the best system. *Proceedings of the Winter Simulation Conference,* 1991, 177-186.

V. Kachitvichyanukul, J.O. Henriksen, R. E. Nance, C. D. Pegden, C. R. Standridge, B. W. Unger, Simulation Environment of the 1990's, Proceedings of the 1987

Winter Simulation Conference (A. Thesen, H. Grant, W. David Kelton eds), 1987, 455-460.

B. Schmeiser. Modern simulation environments: Statistical issues. *Proceedings of the First IE Research Conference,* May 1992, 139--144.

B. Schmeiser. Advanced input modeling for simulation experimentation. *Proceedings of the Winter Simulation Conference*, (P.A. Farrington, H.B. Nembhard, D.T. Sturrock, and G.W. Evans eds.), 1999, 110-115.

B. Schmeiser. Simulation experiments. Chapter 7 in *Handbooks in Operations Research and Management Science, Volume 2: Stochastic Models,* D.P. Heyman and M.J. Sobel (eds.), North-Holland, Amsterdam, 1990, 295-330.

B. Schmeiser and V. Kachitvichyanukul. Non-inverse correlation induction: guidelines for algorithm development. *Journal of Computational and Applied Mathematics* 31 (1990), 173-180.

B. Schmeiser and M.D. Scott. SERVO: Simulation experiments with random-vector output. *Proceedings of the Winter Simulation Conference*, 1991,927-936.

B. Schmeiser and V. Kachitvichyanukul. Correlation induction without the inverse transformation. *Proceedings of the Winter Simulation Conference*, 1986, 266-274.

W.T. Song and B. Schmeiser. Reporting the precision of simulation experiments. New Directions in Simulation for Manufacturing and Communications, (S. Morito, H. Sakasegawa, K. Yoneda, M. Fushimi, and K. Nakano, eds.), *Operations Research Society of Japan*, 1994, 402-407.

AUTHOR BIOGRAPHIES

VORATAS KACHITVICHYANUKUL is an Associate Professor in Industrial Systems Engineering, School of Advanced Technologies, Asian Institute of Technology, Bangkok. He received a Ph. D. from the School of Industrial Engineering at Purdue University in 1982. He has extensive experiences in simulation modeling of manufacturing systems. He had worked for FORTUNE 500 Companies such as Compaq Computer Corporation and Motorola Incorporated. He had also worked for SEMATECH as technical coordinator of the future factory program. His teaching and research interests include planning and scheduling, high performance computing and applied operations research with special emphasis on industrial systems. He is a senior member of IIE and member of SCS. His email address is `<voratas@ait.ac.th>`.

JAMES O. HENRIKSEN is the president of Wolverine Software Corporation. He was the chief developer of the first version of GPSS/H, of Proof Animation, and of SLX. He is a frequent contributor to the literature on simulation and has presented many papers at the Winter Simulation Conference. Mr. Henriksen has served as the Business Chair and General Chair of past Winter Simulation Conferences. He has also served on the Board of Directors of the conference as the ACM/SIGSIM representative.

RICKI G. INGALLS is an Associate Professor and Director of the Supply Chain Design Lab in the School of Industrial Engineering and Management at Oklahoma State University. He joined OSU after 16 years in industry with Compaq, SEMATECH, General Electric and Motorola. He has a B.S. in Mathematics from East Texas Baptist College, a M.S. in Industrial Engineering from Texas A&M University and a Ph.D. in Management Science from the University of Texas at Austin. His research interests include the supply chain design issues and the development and application of qualitative discrete-event simulation. He is a member of IIE. His email address is `<ingalls@okstate.edu>`.

C. DENNIS PEGDEN is Director, Software Development at Rockwell Software, Inc. Sewickley. Prior to this position he was the founder and CEO of Systems Modeling Corporation, now part of Rockwell Software, Inc. He has held faculty positions at the University of Alabama in Huntsville and The Pennsylvania State University. He led in the development of both the SLAM and SIMAN simulation languages. He is the author/co-author of three textbooks in simulation and has published papers in a number of fields including mathematical programming, queuing, computer arithmetic, scheduling, and simulation. His email and web addresses are `<cdpegden@software.rockwell.com>` and `<http://www.rockwellautomation.com>`.

BRUCE W. SCHMEISER is a professor in the School of Industrial Engineering at Purdue University. He received a Ph.D. from the School Industrial and Systems Engineering at Georgia Institute of Technology in 1975. His teaching and research interests lie in applied operations research, with emphasis in stochastic models, especially the probabilistic and statistical aspects of stochastic simulation. He is an active participant in the Winter Simulation Conference, including being Program Chair in 1983 and chairing the Board of Directors during 1988-1990.