Proceedings of the 1987 Winter Simulation Conference
A. Thesen, H. Grant, W. David Kelton (eds.)

SIMULATION ENVIRONMENT OF THE 1990'S (PANEL)

Chair:
Voratas Kachitvichyanukul
Industrial and Management Engineering
The University of Iowa
Iowa City, Iowa 52242

Panelists:
James O. Henriksen
Richard E. Nance
C. Dennis Pegden
Charles R. Standridge
Brian W. Unger

Introduction

Integrated Simulation Environment or Ideal Simulation System is the most talked about topic in the simulation software community in the past few years, (see for examples, Henriksen (1983,1984), Nance (1983), and Schoemaker (1986)). Henriksen (1983) proposed an architecture for the integrated simulation environment and described how each element of the system interacts among one another. Since then progress are made in various aspects of the system and the added features are apparent in the commercially available software.

Conceptually, the advancement of an integrated simulation environment can be viewed in terms of layers of user interfaces as represented in Figure 1 as concentric circles. At the core is the simulation processor which contains the basic kernel routines for executing the simulation. This represents the most primitive user interface outside of the hardware and the operating system. At this level, the users are required to have the expertise and detailed knowledge of the way the simulation processor operates. The next layer contains the simulation languages, database packages, statistical analysis packages, and graphic packages. At this level, the tools are procedure-oriented and/or problem-oriented packages and the users are freed to concentrate on the abstraction of the system. Converting from the abstract model into a computer model is straightforward and only required that the abstract model be built using a particular modeling system. The integration of parts of this layer represent the current state of the art of the simulation environment today (see for example, Standridge and Pritsker (1987)). The last layer is the natural interface where the model builder interacts with the system at the conceptual level. Some examples for this layer are the natural language modeling environment by Su (1987), model design language by Balci (1986), and specification language by Overstreet and Nance (1985).
In spite of the progress made, criticisms abound from all sides. Some of the criticisms are :

From academic purists

1. Current efforts by commercial firms are regressive; i. e., the firms mostly try to perfect the outdated technology in stead of seeking new methods or solutions.

2. Most of the commercially available software today do not include artificial intelligence technique. AI needs to be included somehow.

3. Statistical tools for input and output analysis are not built in. Many such tools are published and known for quite sometime but no commercial firm seems to be interested in including them in the software.

4. Most of the new features added in new software tools are "gee whiz" items that do not improve basic simulation methodology.

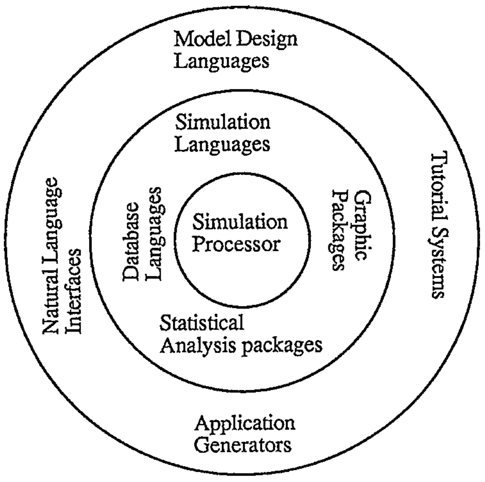

Figure 1. Layers of User Interface for Simulation.

From simulation practitioners

5. Too many claimed features did not live up to the expectation.

6. The new integrated systems cost too much.

7. Not everybody needs integrated system, a toolbox approach is more appropriate.

8. Optimization and design of experiment tools are not built-in.

Impacts of New Technology

Past advancement in simulation methodology has always been tightly paced with the advances of hardware and software technologies. Two emerging new technologies that received a great deal of attention are the parallel computers and the artificial intelligence. Some of the research in simulation related issues on parallel computers are a) concurrent and/or distributed simulation, (Jefferson and Sowizral (1985), Jones (1986), and Gilmer and Hong (1986)); b) simulation optimization with parallel computers, (Lyu (1987)); and c) statistical analysis of parallel simulation, (Heidelberger (1986)).

The extent of the impacts that artificial intelligence technique may have on simulation is yet unclear despite the current "hype". The advancement of AI itself is also tied to progress in hardware and software development. Recent literatures ranged from simulation support diagnostic (Hill and Roberts (1987)) to model generation (Hwang (1985)) to programming paradigm (Ruiz-Mier and Talavage (1987) to knowledge based simulation (Klahr and Faught (1980)).

References

Balci, O. (1986). Requirements for Model Development Environments. Computers and Operations Research, 13, 53-67.

Henriksen, J. O. (1983). The Integrated Simulation Environment (Simulation Software of the 1990's). Operations Research, Vol. 31, no. 6, 1053-1073.

Heidelberger, P. (1986). Statistical Analysis of Parallel Simulation. Proceedings of the 1986 Winter Simulation Conference, Washington D. C., 290-295.

Henriksen, J. O. (1984). Discrete Event Simulation Languages : Current Status and Future Directions. Proceedings of the 1984 Winter Simulation Conference, Dallas, 82-88.

Hill, T. R. and Roberts, S. D. (1987). A Prototype Knowledge-Based Simulation Support System. Simulation, 48, 4, 152-161.

Hwang, S. (1985). Automatic Model Building Systems : A Survey. Proceedings of the 1985 Decision Support Systems Conference, San Francisco, 22-32.

Jefferson, D. and Sowizral, H. (1985). Fast Concurrent Simulation Using the Time Warp Mechanism. Distributed Simulation 1985, The 1985 Society for Computer Simulation Multiconference, San Diego, California.

Klahr, P. and Faught, W. S. (1980). Knowledge Based Simulation. Proceedings of the First AAAI Conference, Stanford, 181-183.

Lyu, J. (1987). Simulation Optimization on Parallel Computers. Unpublished Ph. D. Research Proposal, Industrial and Management Engineering, The University of Iowa.

Nance, R. E. (1984). Model Development Revisited. Proceedings of the 1984 Winter Simulation Conference, Dallas, 74-80.

Nance, R. E. (1981). Model Representation in Discrete Event Simulation: the Conical Methodology. Technical Report CS81003-R, Department of Computer Science, Blacksburg, VA.

Overstreet, C. M. and Nance, R. E. (1985). A Specification Language to Assist in Analysis of Discrete Event Simulation Models. Communications of the ACM, 28, 190-201.

Ruiz-Mier, S. and Talavage, J. (1987). A Hybrid Paradigm for Modeling of Complex Systems. Simulation, 48, 4, 135-141.

Standridge, C. R. and Pritsker, A. A. B. (1987). The Extended Simulation Support System. John Wiley and Sons, New York.

Schoemaker, S. (1986). Simulation Today, in Computer Networks and Simulation III. Schoemaker (Editor), Elsevier Science, North-Holland, 27-39.

Su, H. M. (1987). Natural Language Simulation Modeling Environment. Unpublished Ph. D. Research Proposal, Industrial and management Engineering, The University of Iowa.

Author's Biography

Voratas Kachitvichyanukul is an assistant professor in Industrial and Management Engineering at The University of Iowa. He holds a BS in Chemical Engineering from National Taiwan University, an M Eng from the Asian Institute of Technology, and a Ph D in Industrial Engineering from Purdue University.

Dr. Kachitvichyanukul's current research interests are the development of integrated simulation environment, simulation optimization on parallel computers, special purpose simulation language, random variate generation, and industrial applications of artificial intelligence techniques. His research publications have appeared in Journal of Statistical Computation and Simulation, Communications of the ACM, Simulation, and IIE Transactions. He is member of ACM, TIMS, SCS, and IIE.

Voratas Kachitvichyanukul
Industrial and Management Engineering
The University of Iowa
Iowa City, Iowa 52242
U.S.A.

(319) 335-5931
aegvorwy@uiamvs.bitnet

POSITION STATEMENT

James O. Henriksen
Wolverine Software Corporation
Annandale, Virginia

Historical Perspective

In 1983, I wrote a paper [1] which attempted to identify "...significant improvements that will be made in simulation software in the next 10 years." The paper reviewed trends apparent in 1983 both inside and outside the simulation community, and it presented some "blue-sky" speculative opinions, in very broad terms. It did not attempt to exhaustively catalog pertinent research and/or commercial software tools. Now that four years' time has passed, we are approaching the halfway point of the 10-year period covered by the paper. The purpose of this panel session is to take stock of the current state of the art in simulation software.

Hardware Tools

Improvements in software tools are evolutionary. In contrast, improvements in hardware tools are revolutionary. We live in a menu-driven, point-and-click world, in which desktop machines perform tasks which would have required room-sized equipment ten years ago. The availability of low cost, high performance, personal, desktop hardware has given rise to new expectations for software functionality and ease of use. User expectations create pressures on software developers to harness these capabilities.

Graphics

The use of graphic display hardware to portray the operation of, and results from, simulation models has gained widespread acceptance. For example, in models of manufacturing systems, animation has gained widespread use, both as a model verification tool, and as a means of communicating system behavior to non-simulationists. Similarly, the ability to quickly construct charts and graphs of model results serves a dual role: it provides the simulationist tremendous tools for exploring a model, and once key relationships have been determined, it facilitates communication with non-simulationists. Desktop graphics and statistics are a match made in heaven.

Users expect graphics tools to be easy to use. Producing "quick-and-dirty" displays ought to be easy to do, via non-procedural specifications. Users are willing, up to a point, to expend additional effort to clean up their own "quick-and-dirty" displays, to make them acceptable for presenting results to others (frequently non-simulationist managers).

Artificial Intelligence

The impending marriage of artificial intelligence and simulation is an exciting prospect. Some would maintain that this marriage has already taken place. I feel that although combined AI-Simulation applications are increasing in number, as of 1987, such combined usage has yet to come into widespread use. Therefore, I speak of the marriage in the future tense.

The need for AI in simulation is virtually provable. As simulationists model increasingly larger and more complex systems, the sizes of system decision spaces are exploding. Simulationists need tools which can offer them inferential assistance, i.e., well-designed inference engines ought to thrive on large decision spaces. Unfortunately, AI technology has a long way to go before it acquires the industrial strength required for use with simulation. Many examples of the use of Prolog in conjunction with simulation are beginning to appear in the literature. While I am encouraged to see such examples, I question whether the technology which works for small prototype applications can be readily extended to full-scale, real-world applications.

Two components of AI technology that are of special interest to simulationists are goal seeking and backtracking. Simulationists would like to be able to work with an AI package that was able to respond to requests like "alter the design parameters of this assembly line, so as to achieve the least-cost system capable of producing N widgets per day, and having arrived at the 'solution,' within a specified confidence interval, show me the parameter values and the supporting statistical documentation." Such technology will probably be available within another five years, but certainly not tomorrow.

Methodology

Researchers interested in advancing the state of the art of simulation methodology must come to grips with one unfortunate fact: outside the government and academic communities, methodology does not sell. Consumers of technology will always be willing to spend more money on "gee whiz" items, such as 3D animation, than they will be willing to spend on methodology, even though the reverse makes more sense. We all owe a vote of thanks to the small, but dedicated minority of researchers who are improving our basic methodology.

Commercially Available Simulation Environments

Since the publication of my 1983 paper, Pritsker & Associates have developed and brought to the marketplace a package known as TESS (The Extended Simulation System). To my knowledge, TESS was the first commercially available simulation environment. The offering of similar packages by other vendors is inevitable.

Progress to Date - a Report Card

In my 1983 paper, I identified eight components of a simulation environment. The table which follows expresses my opinion as to the progress that has been made toward realization of each component, expressed as a percentage. I am sure that not all panelists will share this personal assessment.

Component	Progress
Model Editor	5%
Input Preparation Subsystem	50%
Statistics Collection Definition Facility	10%
Experimental Design Facility	5%
Output Definition Facility	5%
Program Editor	50%
Compiler	0%
Run-Time Support	60%

The Future

Over the past four years, substantial progress has been made in improving the simulation technology. Some of the "blue sky" capabilities I wished for in 1983 have already been achieved, but others seem as far away in 1987 as they were in 1983. Overall, the 10-year time frame I hypothesized in 1983 seems to remain reasonable.

References

1. Henriksen, J. O. (1983) The Integrated Simulation Environment (Simulation Software of the 1990's), Operations Research, Vol. 31, no. 6, 1053-1073.

Author's Biography

James O. Henriksen is the president of Wolverine Software Corporation, located in Annandale, Virginia (a suburb of Washington, D. C.) Wolverine Software was founded in 1976 to develop and market GPSS/H, a-state-of-the-art version of the GPSS language. Since its introduction in 1977, GPSS/H has gained wide acceptance in both industry and academia. From 1980-1985, Mr. Henriksen served as an Adjunct Professor in the Computer Science Department of the Virginia Polytechnic Institute and State University, where he taught courses in simulation and compiler construction at the university's Northern Virginia Graduate Center. Mr. Henriksen is a member of ACM, SIGSIM, SCS, the IEEE Computer Society, ORSA, and SME.

Mr. Henriksen is a frequent contributor to the literature on simulation. He has given invited presentations at the Winter Simulation Conference, the Summer Computer Simulation Conference, and at the Annual Simulation Symposium. He served as the Business Chairman of the 1981 Winter Simulation conference and as the General Chairman of the 1986 Winter Simulation Conference.

James O. Henriksen
Wolverine Software Corporation
7630 Little River Turnpike, Suite 208
Annandale, Virginia 22003-2653

(703) 750-3910

POSITION STATEMENT

Charles R. Standridge
Pritsker and Associates, Inc.
West Lafayette, Indiana

Recent Advances in Simulation Software

During the 1980's simulation software evolved from simulation languages which expected textual input and produced standard textual output to simulation systems which provide for graphical input of models and produce animations of simulation dynamics and graphs of variable values. Simulation has become more widely accepted as results and models can be presented in an understandable way and simulation software has become easier to use.

This evolution in simulation software was stimulated by developments in personal computer technology, graphics terminal hardware, graphics software, and database management. Personal computers can simulate a limited but significant number of systems of interest. Furthermore, the single user personal computer environment supports graphics capabilities upon which animations, presentation graphics, and model entry graphics have been effectively built. The coming of low cost graphics terminals and graphics software libraries has allowed these same capabilities to be brought to the minicomputer and mainframe environment. The development of database management systems for simulation-related data has allowed simulation systems to effectively use and control the data and models related to a simulation project.

Simulation systems have evolved to the point that common functionality across systems can be observed. This common functionality includes capabilities such as graphical entry and editing of process oriented models, post-simulation and simulation-concurrent animation, graphing and reporting of simulation results, and statistical analysis.

Simulation systems provide varying degree of integration. Several types of integration are possible. Database management system capabilities available in some systems allow simulation results and simulation inputs to be chosen across scenarios, for selected time intervals, for multiple performance measures, and/or according to any logical conditions in the data when preparing reports, graphs, and animations. Manipulations needed to affect the selection of simulation results are performed automatically by the database management system. Thus, data integration is achieved. Single languages which allow access to all system capabilities provide integration of functionality. Users can move between functions in any sequence as needed. Systems which manage all simulation project information provide integration of projects. Models (or portions of models), icons, graphic formats and the like developed in the course of one project can be share with other projects.

Future Advances

Future advances in simulation systems will come as new technology is incorporated and current issues are resolved.

Engineering workstation computers such as those provided by DEC, SUN, and Apollo will increase in importance as hosts for simulation systems. These systems provide an easy to use single user environment and sufficient computing power for the majority of simulation applications. These machines provide strong graphics support capabilities for hosting graphical model builders, presentation graphics and animations. Some workstations, such as the IRIS, are designed specifically to support animations. Simulation systems need to take advantage of the iconic user interface development tools provided on workstations and the ability to have multiple, concurrent visible processes in windows on a single workstation screen.

Artificial intelligence tools show promise in supporting the development of computer assistance for simulation project tasks previously performed manually. These tasks include model conception, model entry, data collection specification, model simulation, result presentation and animation design, conclusion drawing, and alternative scenario specification. To be most effective, the assistance should be cognizant of the particular system or class of systems of interest.
The role of simulation in overall system design process needs to be explored. An understanding of this role will allow simulation software to be integrated with other system design software, such as CAD/CAM systems, to provide better software support for the design process.

Author's Biography

Charles R. Standridge is a Senior Systems consultant with Pritsker & Associates, Inc. He holds a Bachelor of Science in Applied Mathematics and Computer Science from Washington University in St. Louis, Missouri, and both Master of Science and Doctor of Philosophy in Industrial Engineering/Operations Research from Purdue University.

Dr. Standridge has worked in the application of database management techniques in simulation including the development of integrated support systems for simulation. He led the development of the Simulation Data Language (SDL) and The Extended Simulation System (TESS). Dr. Standridge has been active in the application of this technology to industrial problems and in research to extend this technology. Currently, Dr. Standridge is a member of the software development of at P&A.

Charles R. Standridge
Pritsker and Associates, Inc.
1305 Cumberland Ave
P. O. Box 2413
West Lafayette Indiana 47906

317-463-5557

PARADIGMS AND METHODOLOGIES : ENVIRONMENT BLUEPRINTS

Richard E. Nance
Systems Research Center
and
Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, Virginia

"User friendly" has been displaced as the most misused buzzword in computing technology. This piece of news is not so shocking after all it has had its day and the time has come for new terminological "hype." The reigning term is "methodology" but coming on fast is "paradigm". Unfortunately, both seem destined to have significant effect, either positively or negatively, on the simulation support environments envisioned for the 1990's.

Methodology

Henriksen [1] performed a masterful job of separating fact from fancy in his thoughtful, forward looking description of what was to face us in eight years or so. Laced throughout with both provoking thoughts and pivotal guidance, the paper projects several messages. One which is central is, "Proper integration of software tools into a software development environment can be achieved only through consistent adherence to an underlying software development methodology" [1, p. 1061]. I am in complete agreement with this assertion, and if the word "model" is substituted for "software" in the three instances above, I would embrace the sentence wholeheartedly.

But, where are we five years hence in our understanding or clarification of the term "methodology"? Certainly, its use is pervasive, but do we find that fact comforting? Until we can come to grips with an answer to the question,"What is a methodology?" how can we hope to utilize the concept as a blueprint for model (software) development environments.

Paradigm

The term "paradigm" has major significance for the simulation community, particularly when preceded by either "object oriented" or "automation based". The object oriented paradigm is traced to Simula (more specifically Simula 67) [2], and the influence on future generations of simulation analysts (none of which have ever written an Algol statement) is likely to be pervasive. For the object oriented perspective is influencing artificial intelligence modeling languages and techniques also. Further, the prospect for reusability is becoming an extremely persuasive economic argument.

The automation-based paradigm is an expression of the next step in moving the user further from the details of program execution. Described by Balzer, Cheatham and Green [3], the forerunners of the specification-centered approach to program development and maintenance are already apparent in the numerous design languages populating the marketplace. What effect will this paradigm have on the simulation environment architecture proposed by Henriksen [1]? How will it affect the architecture proposed by Balci [4]?

References

1. Henriksen, J. O. (1983), "The Integrated Simulation Environment (Simulation Software of the 1990's)," Operations Research, 31, 6 (Nov-Dec), 1053-1073.

2. Goldberg, A. (1983), "The Influence of an Object-Oriented Language on the Programming Environment," In Proceedings of the 1983 ACM Computer Science Conference, Orlando, FL (Feb), pp. 35-54.

3. Balzer, R., Cheatham, T.E. and Green, C. (1983), "Software Technology in the 1990's: Using a New Paradigm," Computer, 16, 11 (Nov), 39-45.

4. Balci, O. (1986), "Requirements for Model Development Environments," Computers & Operations Research, 13, 1 (Jan-Feb), 53-67.

Author's Biography

Richard E. Nance is the Director of the Systems Research Center and Professor of Computer Science at Virginia Polytechnic Institute and State University. He serves as Principal investigator for the Navy-funded project in Model development Environment. He has chaired the TIMS College on Simulation and Gaming and has held editorial positions involving simulation publications in Operations Research and IIE Transactions. Dr. Nance is a former member of the WSC Board and the past Chairman of the ACM Special Interest Group on Simulation (SIGSIM).

Richard E. Nance
Department of Computer Science
Virginia Tech
Blacksburg, Virginia 24061

(703) 961-6144
Nance@vtcs1.bitnet

POSITION STATEMENT

C. Dennis Pegden
Systems Modeling Corporation
State College, Pennsylvania

Dramatic changes are taking place in the simulation environment leading up to the 1990's and beyond. These changes have been driven by i) the development of the engineering workstation, ii) the incorporation of animation in simulation software, and iii) improved software for modeling.

In the 1970's simulation were mostly performed in a batch mode on mainframe or mini computers. In the 1980's there has been a rapid transition in simulation from the use of mainframe and mini computers to desktop microcomputers and engineering workstations. Although most of the current popular desktop computers are limited by their memory address space, these limitations will soon disappear as improved operating systems and hardware become available in the next year or two. As a result, I believe that in the 1990's the vast majority of simulations will be performed on desktop engineering workstations. The mainframe computer and simulation software will be reserved for only very large scale models.

The animation capability of simulation software has dramatically improved in the last part of the 1980's. The effort required to generate a highly detailed animation of a system using currently available simulation languages is minor compared to the modeling effort itself. Although animation is currently viewed as an extra and is currently used on a minority of simulation applications, this situation is rapidly changing. The rapid transition towards the common everyday use of animation in simulation is being driven by a number of forces. These include the improved graphics capabilities on the standard desktop engineering computers and a growing appreciation by both modelers and management of the value of animation. I believe that animation will become a routine and standard part of most simulation applications during the 1990's.

Although the use of simulation has been increasing during the 1980's, I believe the real key to its rapid growth in use during the 1990's is the development of improved simulation software. Although there has been improvements in simulation software in the 1980's, the development of a simulation model remains a time consuming and expensive undertaking. What is needed is some dramatic improvements in the way we build models. I believe that a departure from the conventional methods of modeling is needed to reduce the reliance of simulation projects on simulation experts. I expect to see the availability of dramatically improved modeling systems towards the later part of the 1980's and I expect that these will become widely used during the 1990's.

Author's Biography

Dr. C. Dennis Pegden is an Associate Professor in the Industrial and Management Systems Engineering Department at The Pennsylvania State University and President of Systems Modeling Corporation. Prior to his current position, he taught at the University of Alabama in Huntsville where he led the development of the SLAM simulation language. Dr. Pegden received his Ph. D. in 1979 in Industrial Engineering from Purdue University where he studied optimization. His current research interests include both optimization and simulation.

C. Dennis Pegden
Systems Modeling Corporation
248 Calder Way
State College Pennsylvania 16801

(814) 238-5919

SIMULATION ENVIRONMENTS & PARALLELISM

Brian W. Unger
Computer Science Department
University of Calgary
Calgary, Alberta

Parallelism in the execution of simulations may make substantial performance improvement possible. The emerging multicomputer architectures that consist of hundreds to thousands of powerful node computers have tremendous potential for speeding up simulations. Faster simulations will expand the kinds of problems to which simulation can be applied. The ability to speed up execution by several orders of magnitude would make interactive, faster than real time, simulation possible for problems in urban and transportation planning, for example, which have not previously been feasible.

However, three issues must be addressed before substantial speedup via parallelism can be achieved with real simulation problems. The first involves the synchronisation of simulation time in the concurrent, distributed, asynchronous execution of system model components. The second involves the development of models in a way which identifies potential parallelism. The third issue is whether sufficient inherent parallelism exists within modelled systems to make distributed, concurrent execution practical.

In a multicomputer that consists of large numbers of computing nodes it is difficult to provide a globally available clock. This implies asynchronous execution on different nodes and thus a requirement to synchronise simulation time across nodes. Recently, a new "optimistic" synchronisation technique has been described by Jefferson that makes greater parallelism possible for models of arbitrary structure [Jefferson 85]. Past approaches have been based on "conservative" mechanisms which appear to restrict both model structure and the extent of parallelism achievable. An approach based on Jefferson's optimistic mechanism that enables the development of simulation components in multiple programming languages is currently under development at the University of Calgary [Unger 86].

The second model development issue is closely related to the general distributed software development problem. Distributed, parallel programs can be orders of magnitude more difficult to design, implement and test than sequential programs. One reason for this is that a distributed program is non-deterministic and thus errors, in general, cannot be reproduced. Our approach to distributed simulation is based on a programming environment called "Jade" that supports the development of distributed systems of simulations [Joyce 87].

The third issue, whether there exists sufficient inherent parallelism within simulation models to enable substantial speedup, cannot be resolved without further experimental research. Even when there is substantial parallelism in the system being modelled it is not clear that speedup via parallelism is possible during model execution. The key unknown is the ratio of computation overhead required for the synchronisation of simulation time to the computation required to mimic model behavior. Very preliminary performance results are presented in [Berry 86] and further results will be presented at the Multi'88 Distributed Simulation Conference [Unger 88].

Simulation environments of the future must build on traditional object oriented methods and the new logic program paradigms in a way which does not hide the parallelism inherent in models. Distributed operating system kernels and language run-time systems must then be able to exploit this parallelism to support concurrent execution of simulations on multicomputer. Finally, these simulation kernels must support this parallelism transparently, that is, without modification at the model source code levels. Without this transparency models will have to be continually re-written when moving from model development, testing and validation on sequential hardware to parallel hardware for simulation experiments.

References

Berry, O. (1986) "Performance Evaluation of the Time Warp Distributed Simulation Mechanism", Ph.D. Dissertation, University of Southern California, May.

Jefferson, D.(1985) "Virtual Time" ACM Transactions on Programing Languages and Systems 7(3) p.404-425, July.

Joyce, J., Lomow, G., Slind, K., and Unger, B. (1987) "Monitoring Distributed Systems", ACM Transactions on Computer Systems, 5(2), pp 121-150, May.

Unger, B., General Chairman, & Jefferson D., Program Chairman, (1988) SCS Multi'88 Distributed Simulation Conference, San Diego, February.

Unger,B., Cleary, J., Lomow, G., Slind, K. and Li, Xining, (1986) "Jade Virtual Time-Implementation Manual". Research Report #86/242/16, computer Science, University of Calgary, October.

Author's Biography

Brian Unger is a Professor of Computer Science at the University of Calgary, Alberta, Canada. He is also Director of the Software Research and Development Group of the University and was Principle Investigator of Project Jade. Jade is a prototyping and simulation environment that was released in 1985 which supports the development of distributed software and simulations. Dr. Unger received his Ph.D. in Computer Science from the University of California at San Diego in 1972 and has published widely in the field of simulation since that time.

Brian W. Unger
Computer Science Department
2500 University Drive, N. W.
University of Calgary
Calgary, Alberta Canada T2N1N4

(403) 220-6038
unger@calgary.