Technical Report TR-88-26†

SIMULATION MODEL DEVELOPMENT: THE MULTIDIMENSIONALITY OF THE COMPUTING TECHNOLOGY PULL

by

Osman Balci
Richard E. Nance

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

1 November 1988

† Cross-referenced as SRC-88-012, Systems Research Center, VPI&SU. A shorter version of this paper will appear in the *Proceedings of the Conference on Impact of Recent Computer Advances on Operations Research* (Williamsburg, Va., Jan. 4-6, 1989) to be published as a book by Elsevier, New York, N.Y.

ABSTRACT

Recent computing advances in artificial intelligence, computer graphics, human-computer interface, networking, parallel processing, and software engineering pull simulation model development in different directions. This paper discusses the impacts of significant advances in these areas on simulation model development, overviews how our Simulation Model Development Environment design is influenced by the recent advances, and speculates on the impacts of predicted future advances in these areas on simulation model development. The paper concludes that the computing advances in the six areas can be integrated under a simulation environment to provide an extremely powerful platform for simulation model development and execution.

CR Categories and Subject Descriptors: I.6.0 [**Simulation and Modeling**]: General; I.6.1 [**Simulation and Modeling**]: Simulation Theory—*model classification, types of simulation*

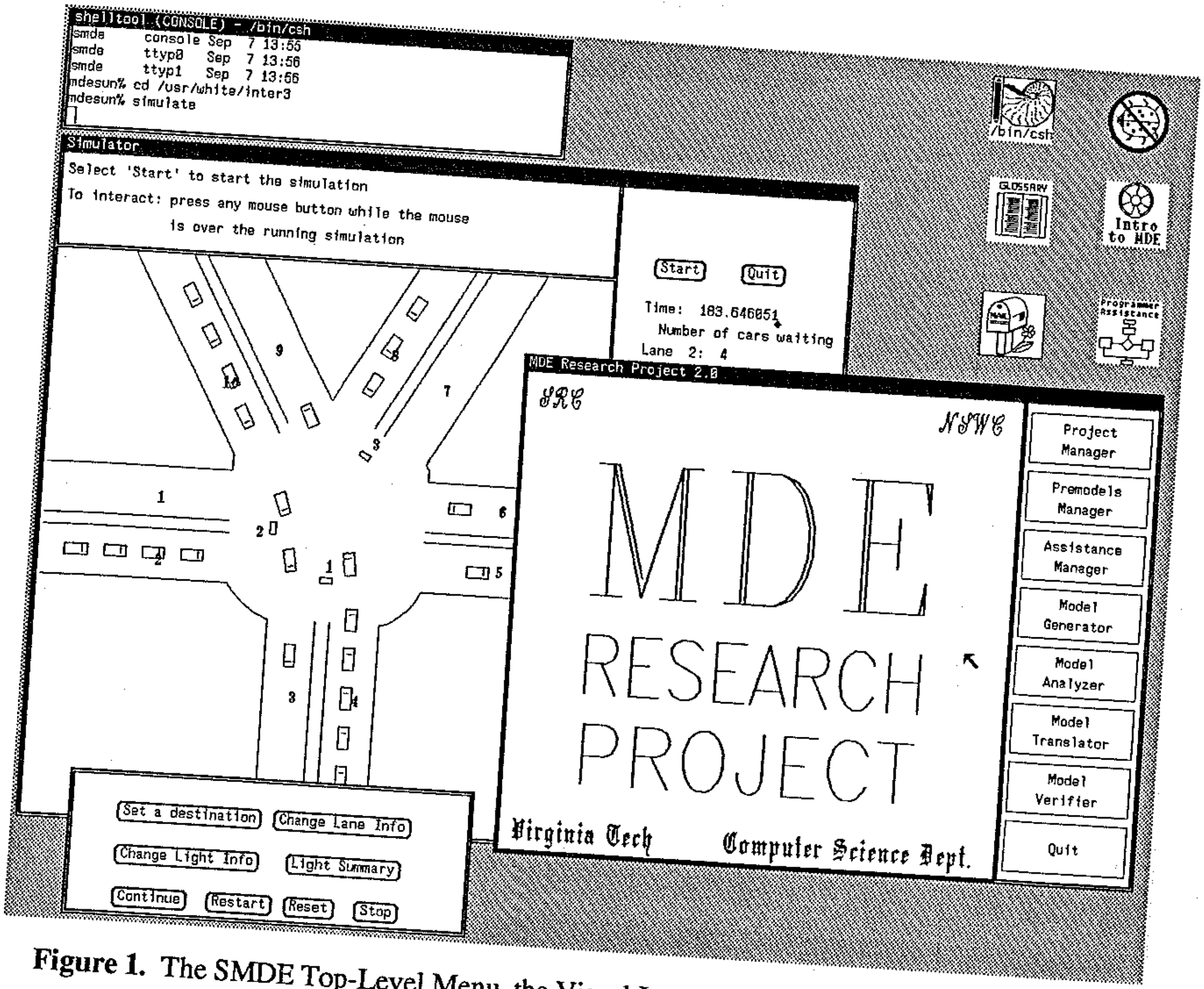
Additional Key Words and Phrases: Model development, simulation modeling

1. INTRODUCTION

Systems design in technology sensitive applications is often characterized as advanced by "needs push" or "technology pull". The reality of the latter descriptor is that a need can be difficult to recognize, and the perception of entirely different forms of meeting that need can prove nearly impossible. For example, one who had used only batch processing systems prior to 1972 could not envision how programming productivity could be enhanced through interactive time-sharing. Frequently, only after experiencing particular technical innovations can the definition of the need for them be formulated. This paper is motivated by the recognition that the "pull" of computing technology has caused the needs for simulation model development to undergo significant redefinition.

The early development of discrete event simulation models assumes one of two forms: (1) the use of functional routines to assist in the production of a model in a general purpose language, typically FORTRAN, or (2) the employment of one of the specialized simulation programming languages, which often entails a significant learning commitment. Both approaches characterize model development activities in the early to mid 1960s, and both continue to dominate the current scene. The former, or *simulator*, approach has persisted largely because of the predominance of FORTRAN as an instructional language in engineering curricula; the latter, because of strong advances in simulation programming language (SPL) capabilities in the 1970s. A weakness of both the simulator and SPL approaches is the inherent limitation of a model *representation* that must also serve as a model *implementation*. The 1980s mark a transition in the understanding of what is needed to cope with the size and complexity of models that are no longer bound by execution time constraints. The interested reader is referred to the following sources, which treat this problem in greater depth and from different perspectives [Nance 1983; Henriksen 1983; Kachitvichyanukul 1987].

Since June 1983 our MDE project has addressed a complex research problem: prototyping discrete-event Simulation Model Development Environments (SMDE) following the automation-based software paradigm [Balci and Nance 1987a]. The major research goal has been to provide an *integrated* and comprehensive collection of computer-based tools to: (1) offer cost-effective, integrated, and automated support of model development *throughout the model life cycle*, (2) improve the model quality by effectively assisting in the quality assurance of the model, (3) significantly increase the efficiency and productivity of the project team, and (4) substantially decrease the model development time. The SMDE prototype is being developed on a Sun color computer workstation. Figure 1 shows the SMDE top-level menu, a component of the visual interactive simulator [White 1988], and other tools illustrating the Sun technology. The reader is referred to [Balci and Nance 1987b] for details of the SMDE prototype.



```
shelltool (CONSOLE) - /bin/csh
smde     console Sep  7 13:55
smde     ttyp0   Sep  7 13:56
smde     ttyp1   Sep  7 13:56
mdesun% cd /usr/white/inter3
mdesun% simulate
```

```
Simulator
Select 'Start' to start the simulation
To interact: press any mouse button while the mouse
             is over the running simulation
```

```
Time:  183.646051
  Number of cars waiting
Lane  2:  4
```

Figure 1. The SMDE Top-Level Menu, the Visual Interactive Simulator, and Other Tools.

In this paper we review the impact of significant recent advances in several important areas of computing technology on simulation model development. While computing technology has experienced notable advances across a broad front, the following selected areas certainly must rank among the most significant: (1) artificial intelligence, (2) computer graphics, (3) human-computer interface, (4) networking, (5) parallel processing, and (6) software engineering. In every section devoted to each of the six areas, we: (a) review the impact of substantial recent advances in that area on simulation model development, (b) overview how our SMDE design is influenced by the recent advances in that area, and (c) speculate on the impact of predicted future computing advances in that area on simulation model development. Concluding remarks are presented in Section 8.

2. ARTIFICIAL INTELLIGENCE

A strong relationship exists between some areas of Artificial Intelligence (AI) and simulation as illustrated by the following three examples. Modeling and simulation of human intelligence has played an important role in AI [Oren 1987]. An expert system models a body of knowledge and simulates the way the knowledge is processed. Rule-based programming has been known to simulationists since 1962 when Buxton and Laski [1962] implemented Activity Scanning conceptual framework in Control and Simulation Language although it was not labeled as such.

Two "schools of thought" are apparent in the attitude toward the AI/discrete event simulation interaction in simulation model development: one ascribes to the primacy of AI as the fundamental basis for development; the other, sees AI as only a contributor. Because of space limitations, we confine the discussion to the former "school", and review the impact on simulation model development in five categories: (1) knowledge-based simulation, (2) intelligent simulation environments, (3) intelligent front ends for model generation, (4) goal-directed simulation, and (5) introspection.

The Rule Oriented Simulation System (ROSS) [McArthur et al. 1986], developed at the Rand Corporation, is an English-like, object-oriented, interactive, and visual language implemented in Lisp. Relying on recently developed AI techniques and expert systems technology, ROSS provides a simulation environment in which users can conveniently design, test, and modify large knowledge-based simulation models. It has been successfully applied to a strategic air penetration simulation [Klahr et al. 1982] and a tactical ground-based combat simulation [Klahr et al. 1986]. ROSS has significantly impacted the manner in which simulation models of soft problems are developed with its integration of the object oriented paradigm, visual interactive simulation, and AI techniques.

The Knowledge-Based Simulation system (KBS) [Reddy et al. 1986] is similar to ROSS; however, KBS stresses the automatic analysis of simulation results. All simulation models are descriptive in nature. They produce results with no indication of the "goodness" or "badness" of the results leaving the burden of analysis and interpretation to the simulationist. On the other hand, prescriptive models, like linear programming models, produce results (solutions) with a label such as "optimal", "feasible", or "infeasible", thereby facilitating the analysis of results. Analyzing and interpreting the results of a large and complex simulation model are very difficult. KBS tries to resolve this problem by way of providing prescriptive results.

The Knowledge-Based Model Construction (KBMC) system [Murray and Sheppard 1988] uses domain knowledge, extracts the necessary information from the modeler in a structured interactive dialog, and generates a complete model specification. A programmed model in the SIMAN language is automatically created by the KBMC system from this specification, utilizing modeling knowledge and SIMAN knowledge. Similarly, the Modeling Advisor for GEST programs (MAGEST) [Oren and Aytac 1985] uses knowledge about the GEST morphology, incremental knowledge about the user programs, and domain knowledge to generate a model specification in the GEST language. A programmed model in Simscript II.5 is then created from this specification. Representing the program generation approach, KBMC and MAGEST both exemplify ideas in the achievement of the automation-based paradigm for simulation model development [Balci and Nance 1987a].

Shannon [1986] provides an excellent overview and speculates that one of the major impacts of AI on simulation will be the creation of intelligent simulation environments to make the task of conducting simulation studies easier, faster and more accurate. The SMDE research project has been termed very ambitious by several colleagues although SMDE prototypes have been mostly “unintelligent.” (The term “intelligent environment” is used as described in [Shannon 1986].) Building one that is intelligent under the objectives of the SMDE research project is simply not possible within the current technology. We speculate that it will be a long time (at least a decade) for domain-independent, generic intelligent simulation environments to be realized.

Doukidis and Paul [1986] describe their experiences in building a natural language understanding system to aid the formulation of simulation models. The success of developing a natural language interface for model generation is dependent upon the advances in translation devices, voice recognition systems, and synthetic voice reproduction devices. Recent advances have provided only encouragement for further research and significant impact on simulation model development will not be realized for at least another decade.

The Goal-Directed Simulation approach proposes a major shift from the traditional problem analysis view to solution synthesis view. It suggests the construction of a simulation model as a collection of goals decomposed into sub-goals. At the execution and experiment levels, the model seeks to achieve the goals given the knowledge by the user about the objectives, goals, performance criteria, and behaviors [Umphress and Pooch 1987]. Recent research in this area has attempted to demonstrate the feasibility of this approach.

Using the introspection AI technique, the simulation model is built to learn about itself and to gain knowledge by tracing the dynamics of its execution. Introspection is a common objective of many knowledge-based systems. If achieved, this technique should prove very beneficial especially for model verification and validation. Currently, existing as an idea only, introspection requires further developments to prove a workable approach.

3. COMPUTER GRAPHICS

Recent advances in computer graphics have made visual simulation possible. Visual simulation can now be conducted as post-simulation animation or simulation-concurrent animation on many graphics workstations with graphics processors and on Personal Computers. Making visual simulation interactive requires more computing power than the already very demanding power required for noninteractive simulation. Thus the impact of recent advances in computer graphics on simulation is also dependent on the advances in the semi-conductor (computer architecture) technology to provide more powerful hardware required for visual simulation.

Grant and Weiner [1986] review the following ten commercially available graphically animated simulation systems: AutoGram, BEAM, Cinema, Modelmaster, PCModel, RTCS, SeeWhy, SimFactory, SIMPLE1, and TESS. They indicate that "Collectively, about 500 animated simulation systems have been installed in the U.S. Four years ago there were less than ten such installations." Other work in visual simulation includes: ANDES [Birtwistle et al. 1984], GMBS [Yamamoto and Lenngren 1983], Performance Analysis Workstation [Melamed and Morris 1985], RESQME [Gordon et al. 1986], and SIMSEA [Langlois 1984]. Bell and O'Keefe [1987] present a short history of Visual Interactive Simulation (VIS), describe some recent developments, and suggest four future areas of research: (1) type and quality of visual display, (2) software and hardware for VIS, (3) the need for methodology, and (4) the role of expert systems.

Stephen Jobs (co-founder of Apple Computer) has founded NeXT, Inc. in September 1985 with the focus of developing the next generation of computers and software for higher education under the vision of *simulated learning environment* (SLE). The key focus of the SLE is visual simulation/animation for the purpose of education. John Sculley, CEO of Apple Computer, in his keynote speech at the Macworld Exposition in Boston on August 11, 1988, showed a video of a computer animation of flying over a three-dimensional picture of California taken from a satellite. He indicated that in this animation, which will be ported to an upcoming Macintosh, the user will be the pilot not the animator. Apple is also working on the development of a SLE. The development is taking place on a Cray supercomputer for porting to an upcoming Macintosh in the near future. The moves of the two companies clearly indicate that visual simulation/animation will play a crucial role in education and training.

In our SMDE research project, White [1988] has prototyped a general purpose traffic intersection visual interactive simulator to gain some experience with VIS and to develop some expertise in using SunView and SunCore software packages for VIS. This work is being extended by the implementation of a new conceptual framework in the development of a visual interactive simulator, applicable to a large class of problems.

Predicted future advances in the high-speed computing technology and computer graphics should make the VIS a more viable approach. Computer graphics will play a crucial role in the development of simulation models used for training and educational purposes.

4. HUMAN-COMPUTER INTERFACE

The research at Xerox's Palo Alto Research Center between 1978 and 1982 provides the underpinnings for the current state-of-the-art technology for human-computer interfaces. In 1982, Sun Microsystems developed a window management system, SunWindows (later its improved version was called SunView), around the technology developed by Xerox. Apple Computer has implemented a different version of Xerox's graphical interface technology in their Macintosh computers in 1984, opening the door for a whole new type of software. Recently, the Presentation Manager interface for Microsoft's new operating system, OS/2, incorporates a graphical interface which is very similar to the Macintosh's.

Today, all the major players in the microcomputer industry believe in graphical interfaces and appreciate the "what-you-see-is-what-you-get" benefit of these visual interfaces. Unfortunately, the simulation community has continued ignoring the potential benefits of this interface technology in simulation model development. Graphical interfaces have been provided by some vendors to extract information from the user to automatically generate a simulation model in a particular simulation programming language for a particular problem domain (e.g., Network II.5 for computer-communications network design and analysis, Comnet II.5 for telecommunication network analysis). The state-of-the-art interface technology has not yet been effectively utilized for simulation model development.

The SMDE research project is highly dependent on the Sun interface technology (specifically the usage of a mouse, a high-resolution monitor, icons, windows, scroll bars, buttons, pull-down menus, pop-up menus, and hierarchical menus). The technology has significantly facilitated the rapid prototyping of SMDE tools. The prototypes have been built in much less time and in a much better form expediting the experimentation with the prototypes. Clearly, productivity gains have been realized over the dumb-terminal interface technology used earlier in the research project. In continuation of the current research [Derrick 1988], a new conceptual framework is under development for simulation modeling using Sun's state-of-the-art interface technology.

The competition for creating a well human-engineered graphical interface for the UNIX operating system is heating up in industry. Sun Microsystems and AT&T have formed an alliance for the UNIX market and they are developing a graphical interface, called Open Look, for their version of UNIX.

Seven major computer companies (IBM, DEC, HP, Apollo, Siemens, Nixdorf, and Bull) have formed the Open Software Foundation (OSF) to compete with the Sun-AT&T alliance for the UNIX market [Chandler 1988]. OSF is currently soliciting proposals from the industry for the development of the graphical interface for their version of UNIX. These developments clearly indicate that the human-computer interface technology will play a crucial role in the success of many operating systems (e.g., UNIX, OS/2, Macintosh OS).

The significant technological advances in the human-computer interfaces provided by operating systems will considerably change the manner in which information is extracted from a modeler, thereby facilitating simulation model development. An advanced interface can assist the modeler in representing a system at a level which is much closer to his conceptualization. Advances should hasten the transition from implementation-centered to specification-centered model development, the crux of the automation-based paradigm.

5. NETWORKING

In the past, computer networking has been divided into two subdomains: wide area and local networks. The criteria distinguishing the two – geographical separation, data rate, and ownership [Tanenbaum 1981] – have become blurred by rapid advances in communications technology.

Increases in data rates, from 56 Kbps for the typical wide area network (WAN) link to 1 Mbps currently, have forced a third intervening subdomain: Metropolitan Area Networks (MANs) [Tanenbaum 1988, p. 117]. The advances in data rates have enabled the distributed execution of simulation models, with run-time linking of components on machines separated by huge distances. The *virtual laboratory*, a concept of integration of simulation, emulation, and experimentation utilizing resources located across thousands of miles [Nance 1982], is becoming a reality. However, advances in wide area link transmission rates have exerted influence on simulation *model execution* far more than *model development.*

Both practitioners and researchers draw direct benefits from the sharing of ideas through the Simulation Digest, an electronic bulletin board administered by Professor Paul Fishwick of the University of Florida. In general, WAN developments have spawned far wider and more frequent sharing of ideas and writings among members of the simulation community. The communication channels of *The Invisible College* for simulation now link terminals, workstations, microprocessors, and main frames across the world. The appearance of communications products based on the Open System Interconnection model is now moving into the session and presentation layers [Stallings 1985, pp. 385-394], which assures the expansion of data sharing and a more open, progressive research community.

Since the early 1970s, following Farber and Larsen's [1972] seminal report, the advance of local network technology has appeared confused, sometimes even chaotic. Initially, the perception of an intimate tie with distributed processing added to the confusion. However, as concepts have become more clearly defined and distinctions more apparent, local network design and operation looms as a research and an application domain related to WAN, distributed systems, and even network operating systems. As the IEEE Local Network standards have evolved and the new technology has matured, the impact of network computing has become discernible.

Simulation modeling is already feeling the effect of workstations replacing large mainframes as the principal development engines. The effect on model development is only beginning to be perceived.

The super computers and mainframes do not and will probably not, in the foreseeable future, provide the human-computer graphical interface technology available today in microcomputers (Macintosh, SUN, etc.). This pulls simulation model development onto the microcomputers; however, microcomputers are not powerful enough to run large simulations. Today, there are simulations of nuclear power plants that take about eight hours of CPU time on a Cray II supercomputer. Making a Cray II ten times faster, however, does not necessarily imply a significant decrease in the simulation execution times. Because of the increased power, much more detailed and much larger simulation models will be constructed requiring eight hours of CPU time again.

A solution lies in the networking of microcomputers with super computers and mainframes. The modeler will develop the simulation model on a microcomputer but its execution will take place on a super computer or mainframe in such a way that the usage of the supercomputer or mainframe will be transparent to the user. This technology is now made possible by the development of the X window system at MIT. Sun Microsystems has proposed an improved version, Network extensible Window System (NeWS), based on the PostScript language with powerful graphics primitives. NeWS provides a platform on which highly diverse user interfaces and window applications can be built independently of any computer hardware or operating system. Recently, many companies have standardized on the X.11 window system.

We anticipate a much more detailed solution to this problem in the near future with the availability of Mach – a multiprocessor oriented operating system and environment. Mach is currently being developed at Carnegie Mellon University with the objective of providing an extensible kernel that supports an integrated, networked, and UNIX compatible computing environment consisting of both large and small multiprocessors and uniprocessors. It provides a binary compatible 4.3BSD UNIX interface to include: (1) multiple threads of control per address space, (2) extremely flexible memory sharing, (3) capability-protected, network-transparent interprocess communication which has been integrated with the virtual memory system, and (4) support for both loosely-coupled and tightly-coupled multiprocessors [Chew and Rashid 1988].

6. PARALLEL PROCESSING

Since the earliest perceptions of parallel processing as offering the next major source of computational speedup, discrete event simulation has proved a fertile domain both for theoretical and practical interest. The explanation of this interest most likely lies in the inherent preoccupation of simulation models with time and the underlying *conceptual* insistence on some global indexing attribute, usually labeled "system time." Such a view seems contrary to a perceived basic tenet of parallel processing: abandonment of centralized control of program execution.

Clearly, the emphasis of research in parallel (concurrent or distributed) simulation has been on model execution: how to reduce the time (and cost) of obtaining samples of model behavior. This point notwithstanding, the implications for model development, although indirect, are significant.

A recent result of research in this area is to categorize the execution of simulation models on different processors, perhaps with no motivation for speedup, as *distributed simulation*. *Parallel* or *concurrent simulation* is the attempt to exploit non-sequential execution to achieve speedup. Using this definitional distinction, distributed simulation is more closely related to computer networking, and the treatment in this paper follows that distinction.

The earliest treatments of parallel model execution focus on the consequent problems in eliminating global control, problems of immense concern in the operating systems research community as well. Loss of computational integrity – assuring the correctness of a computational process – and deadlock – the inability to exchange data necessary to complete a computation – are the challenging problems. Dominating these early works are the mechanisms for guaranteeing the conclusion of a correct computational process [Bryant 1977; Peacock et al. 1979]. Algorithms to accomplish this guarantee are described in various works [Chandy and Misra 1981; Reynolds 1982; Nicol and Reynolds 1984]. These algorithms have been classified as the *conservative approach* to exploiting parallelism. Underlying this approach is the *a priori* guarantee that permitting two *events* to occur simultaneously has no effect on the occurrence of future events [Chandy and Misra 1981, p. 205].

Another approach, typified by Jefferson [1985] and Sokol et al. [1987], is deemed the *optimistic approach*. The "optimism" is expressed in the philosophy of letting individual *processes* proceed in an independent fashion until the discovery of incorrect behavior, then roll back all affected processes to regenerate the correct behavior.

In addition to the *a priori* versus *a posteriori* distinction, note that the early optimistic approach has an event perspective while the optimistic approach takes a process view. The process view is also taken in the later development of conservative algorithms [Nicol and Reynolds 1984].

An entirely different exploitation of parallelism is to allocate simulation support functions, e.g. events list management, random number generation, etc., to separate processors [Comfort 1982; Wyatt et al. 1983]. Such an approach proves simpler to develop and more robust (less subject to inherent model dependencies). The model development task remains essentially impervious under this approach since the functional partitioning is model independent. Exploitation of parallelism at this high level, compared to the potentially very low level when dealing with model decomposition, could lead to diminished returns in execution time (cost). The level of parallelism is termed the *granularity*, which could be extremely important in the speedup realized from parallel execution.

The early "world views" (event scheduling, activity scan, and process interaction) represent a granularity that Overstreet has effectively described as forms of "model locality" [Overstreet 1982, pp. 164-165]. Both the event and process forms have stimulated model partitioning strategies, but neither might represent the most efficient level of granularity for a particular model. We suspect that the most efficient level is highly model dependent; however, the issue of granularity in the exploitation of parallelism remains a matter of investigation.

The speedup obtained through parallelism is also a key to utilizing AI techniques in model representation that do not impose an excessive penalty on model execution. This observation assists in judging the feasibility of approaches to intelligent simulation environments described in Section 2.

To date, the SMDE prototypes have been unconcerned with the challenges of parallel simulation in either the effect on model development or model execution. However, this is likely to change in the near future, prompted by a recognition of some promising relationships between world views and granularity.

7. SOFTWARE ENGINEERING

Quite expectedly, the initial impact of software engineering is seen at the program level: the use of structured programming and top-down design techniques in a simulation programming language (Simscript II.5) [Heimburger 1976]. Early efforts to relate software engineering to discrete event simulation view the latter as simply an applications programming domain. Within the research community in software engineering, Lehman emerges as one of the few early contributors to characterize the programming task as inherently a modeling activity [Lehman 1980, pp. 4-8]. This small group includes Zurcher and Randell, colleagues of Lehman at IBM T.J. Watson Research Center. The modeling characterization underscores a major linkage between computer science and operations research: both are essentially *problem solving disciplines.*

The perception of influence is often slow to materialize. For example, Simula, originally introduced in 1963 as a discrete event language extension of Algol, embodies several major conceptual advances for that time (and for today): implementations of the process mechanism, co-routine concept, abstract data type (class) concept, and inheritance (class concatenation). The advanced capabilities of Simula only now are beginning to be appreciated, stimulated by the popularity of the object oriented paradigm (OOP), primarily associated with Smalltalk. Simula is now recognized as the first language to exhibit the OOP principles.

The enunciation of the automation-based paradigm [Balzer et al. 1983; Balzer 1985] has served to elevate program specification and the attendant modeling concerns in the consciousness of the software engineering community. In fact, a much earlier work, depicting program development as a procession of (simulation) model developments [Zurcher and Randell 1969], presents a design prototyping approach (without the throwaway aspect of *rapid prototyping*). Perhaps the combined legacy of OOP and the automation-based paradigm is the credibility of the assertion that *development of a program should initiate with development of a model.*

Areas of interest mutual to software engineering and simulation model development are clearly identifiable: software (model) management [Nance et al. 1981], program (model) life cycle [Balci 1987; Nance 1981], software (model) development methodologies [Oren and Zeigler 1979; Nance 1979, pp. 89-93] and software (model) development environments [Henriksen 1983; Balci 1986]. As is evident from the citations, the SMDE Project has resulted in a confluence of simulation and software engineering concepts. Hopefully, the future impact shows that the results have contributed to support environments that enhance productivity in modeling activities beyond simulation and in *system* development projects that transcend the software domain.

8. CONCLUDING REMARKS

Clearly, recent and continuing advances in the six areas reviewed exert different directional "pulls" on simulation model development. The challenge is to meld these technological advances so as to mold a powerful environment for simulation model development and execution. This environment development should exploit these six areas to achieve complementary rather than competitive influences.

The OOP provides a desirable conceptual framework for simulation modeling and produces much needed reusable and maintainable models; however, because of the overhead involved, OOP-based models have relatively poorer execution speeds. The use of computer graphics significantly amplifies the need for more CPU power. As for AI, the need for computational power is so high that special computer hardware is manufactured to try to meet the demand (e.g., Lisp workstations). Simulation is also known to be extremely CPU intensive. Hence, combining OOP, computer graphics, and AI for the development and execution of simulation models poses a phenomenal demand for CPU power.

Advances in the semi-conductor technology (computer architecture), parallel processing, and networking jointly contribute to meeting the ever-increasing demand for CPU power. UNIX-based Mach environment integrates parallel processing and networking at a low level, providing a promising platform for simulation model development and execution. Thus, Mach is selected for the NeXT computer workstation which provides an interactive simulated learning environment combining computer graphics, visual simulation, animation, and some AI techniques.

Finally, the "bottom line" of this review can be succinctly stated for the SMDE project: In the rapidly changing technological context of today, rapid prototyping is a necessity not an alternative.

ACKNOWLEDGMENTS

This research was sponsored in part by the U.S. Navy under contract N60921-83-G-A165-B03 through the Systems Research Center at VPI&SU. We thank Professor James D. Arthur for reading the manuscript and his suggestions for improvement.

REFERENCES

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

Balci, O. (1987), "Guidelines for Successful Simulation Studies – Part I & II," Technical Report TR-85-2, Department of Computer Science, Virginia Tech, Blacksburg, Va., Mar., 56 pp.

Balci, O. and R.E. Nance (1987a), "Simulation Support: Prototyping the Automation-Based Paradigm," In *Proceedings of the 1987 Winter Simulation Conference* (Atlanta, Ga., Dec. 14-16). IEEE, Piscataway, N.J., pp. 495-502.

Balci, O. and R.E. Nance (1987b), "Simulation Model Development Environments: A Research Prototype," *Journal of the Operational Research Society 38*, 8 (Aug.), 753-763.

Balzer, R. (1985), "A 15 Year Perspective on Automatic Programming," *IEEE Transactions on Software Engineering SE-11*, (Nov.), 1257-1268.

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

Bell, P.C. and R.M. O'Keefe (1987), "Visual Interactive Simulation – History, Recent Developments, and Major Issues," *Simulation 49*, 3 (Sept.), 109-116.

Birtwistle, G., J. Joyce, and B. Wyvill (1984), "ANDES: An Environment for Animated Discrete Event Simulation," In *Proceedings of the U.K. Simulation Conference* (Bath, U.K.), Sept.

Bryant, R.E. (1977), "Simulation of Packet Communication Architecture Computer Systems," MIT LCS/TR-188, Massachusetts Institute of Technology, Cambridge, Mass.

Buxton, J.N. and J.G. Laski (1962), "Control and Simulation Language," *The Computer Journal 5*, 194-199.

Chandler, D. (1988), "The Birth of the Open Software Foundation," *UNIX Review 6*, 7 (July), 16-31.

Chandy, K.M. and J. Misra (1981), "Asynchronous Distributed Simulation via a Sequence of Parallel Computations," *Communications of the ACM 24*, 11, (Apr.), 198-206.

Chew, J.J. and R.F. Rashid (1988), "MACH – A Multiprocessor Oriented Operating System and Environment," *README – The Newsletter of the Sun Microsystems Users Group 3*, 2 (Spring-Summer), 1,11.

Comfort, J.C. (1982), "The Design of a Multi-Microprocessor Based Simulation Computer - 1," In *Proceedings of the Annual Simulation Symposium* (Tampa, Fla., Mar.). pp. 45-52. (Note: Companion papers appear in the 1983 and 1984 Proceedings.)

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

Doukidis, G.I. and R.J. Paul (1986), "Experiences in Automating the Formulation of Discrete Event Simulation Models," In *AI Applied to Simulation*, E.J.H. Kerckhoffs et al. Eds. SCS Simulation Series, Vol. 18, No. 1, (Feb.), 79-90.

Farber, D.G. and K.C. Larsen (1972), "The System Architecture of the Distributed Computer System – The Communications System," In *Proceedings of theSymposium on Computer Networks* (Brooklyn Polytechnic Institute, Brooklyn, New York, Apr.)

Gordon, R.F., E.A. MacNair, P.D. Welch, K.J. Gordon, and J.F. Kurose (1986), "Examples of Using the Research Queueing Package Modeling Environment (RESQME)," In *Proceedings of the 1986 Winter Simulation Conference* (Washington, D.C., Dec. 8-10). IEEE, Piscataway, N.J., pp. 494-503.

Grant, J.W. and S.A. Weiner (1986), "Factors to Consider in Choosing a Graphically Animated Simulation System," *Industrial Engineering 18*, 8 (Aug.), 36-38,40,65-68.

Heimberger, D.A. (1976), "Structured Programming Using SIMSCRIPT II.5," Presentation at the Simulation and SIMSCRIPT Conference, Washington, D.C., Sept. 21.

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

Jefferson, D. (1985), "Virtual Time," In *Transactions on Programming Languages and Systems 7*, 3 (July), pp. 404-425.

Kachitvichyanakul, V. (1987), "Simulation Environment of the 1990s (Panel)," In *Proceedings of the 1987 Winter Simulation Conference* (Atlanta, Ga., Dec. 14-16). IEEE, Piscataway, N.J., pp. 455-460.

Klahr, P., D. McArthur, S. Narain, and E. Best (1982), "SWIRL: Simulating Warfare in the ROSS Language," Publication Number N-1885-AF, The Rand Corporation, Santa Monica, Calif., Sept.

Klahr, P., J.W. Ellis, W. Giarla, S. Narain, E.M. Cesar, and S.R. Turner (1986), "TWIRL: Tactical

Warfare in the ROSS Language," In *Expert Systems – Techniques, Tools and Applications,* P. Klahr and D.A. Waterman, Eds. Addison-Wesley, Reading, Mass., pp. 224-273.

Langlois, L. (1984), "Simulation Visualization with SIMSEA: A General Purpose Animation Language," In *Proceedings of the SCS Conference on Simulation in Strongly Typed Languages* (San Diego, Calif., Feb.). SCS, San Diego, Calif.

Lehman, M.M. (1980), "Programs, Programming and the Software Life Cycle," Report No. 80/6, Department of Computing and Control, Imperial College of Science and Technology, London, Apr.

McArthur, D., P. Klahr, and S. Narain (1986), "ROSS: An Object-Oriented Language for Constructing Simulations," In *Expert Systems – Techniques, Tools and Applications,* P. Klahr and D.A. Waterman, Eds. Addison-Wesley, Reading, Mass., pp. 70-94.

Melamed, B. and R.J.T. Morris (1985), "Visual Simulation: The Performance Analysis Workstation," *Computer 18*, 2 (Aug.), 87-94.

Murray, K.J. and S.V. Sheppard (1988), "Knowledge-Based Simulation Model Specification," *Simulation 50*, 3 (Mar.), 112-119.

Nance, R.E. (1979), "Model Representation in Discrete Event Simulation: Prospects for Developing Documentation Standards," In *Current Issues in Computer Simulation*, N. Adam and A. Dogramaci, Eds. Academic Press, New York, pp. 83-97.

Nance, R.E. (1981), "Model Representation in Discrete Event Simulation: The Conical Methodology," Technical Report TR-81-3, Department of Computer Science, Virginia Tech, Blacksburg, Va., Mar.

Nance, R.E. (1982), "Data Transfer Architectures: Development of the Capability for Comparative Evaluation," Technical Report NSWC TR 82-345, Naval Surface Weapons Center, Dahlgren, Va., Mar.

Nance, R.E. (1983), "A Tutorial View of Simulation Model Development," In *Proceedings of the 1983 Winter Simulation Conference* (Arlington, Va., Dec. 12-14). IEEE, Piscataway, N.J., pp. 325-331.

Nance, R.E., A.M. Mezaache, and C.M. Overstreet (1981), "Simulation Model Management: Resolving the Technological Gaps," In *Proceedings of the 1981 Winter Simulation Conference* (Atlanta, Ga., Dec. 9-11). IEEE, Piscataway, N.J., pp. 173-179.

Nicol, D.M. and P.F. Reynolds, Jr. (1984), "Problem Oriented Protocol Design," In *Proceedings of the 1984 Winter Simulation Conference* (Dallas, Tex., Nov. 28-30). IEEE, Piscataway, N.J., pp. 471-474.

Oren, T.I. (1987), "Artificial Intelligence and Simulation: From Cognitive Simulation Toward Cognizant Simulation," *Simulation 48*, 4 (Apr.), 129-130.

Oren, T.I. and B.P. Zeigler (1979), "Concepts for Advanced Simulation Methodologies," *Simulation 32*, 3 (Mar.), 69-82.

Oren, T.I. and Z.K. Aytac (1985), "Architecture of MAGEST: A Knowledge-Based Modeling and Simulation System," In *Simulation in Research and Development*, A. Javor, Ed. North Holland, Amsterdam, pp. 99-109.

Overstreet, C.M. (1982), "Model Specification and Analysis for Discrete Event Simulation," Ph. D. Dissertation, Department of Computer Science, Virginia Tech, Blacksburg, Va., Dec.

Peacock, J.K., J.W. Wong, and E. Manning (1979), "Distributed Simulation Using a Network of Processors," *Computer Networks 3*, 44-56.

Reddy, Y.V.R., M.S. Fox, N. Husain, and M. McRoberts (1986), "The Knowledge-Based Simulation System," *IEEE Software 3*, 2 (Mar.), 26-37.

Reynolds, P.F., Jr. (1982), "A Shared Resource Algorithm for Distributed Simulation," In *Proceedings of the Ninth Annual International Computer Architecture Conference* (Austin, Tex., Apr.). pp. 259-266.

Shannon, R.E. (1986), "Intelligent Simulation Environments," In *Proceedings of the Conference on Intelligent Simulation Environments* (San Diego, Calif., Jan. 23-25). Published as *Simulation Series 17*, 1 (Jan.), 150-156. SCS, San Diego, Calif.

Sokol, L.M., D.P. Driscoe, and A.P. Wieland (1987), "MTW: A Strategy for Scheduling Discrete Simulation Events for Concurrent Execution," MP-8700018, The MITRE Corporation, McLean, Va., Oct.

Stallings, W. (1985), *Data and Computer Communications*, Macmillan, New York.

Tanenbaum, A.S. (1981), *Computer Networks*, Prentice-Hall, Englewood Cliffs, N.J.

Tanenbaum, A.S. (1988), *Computer Networks*, 2nd edition, Prentice-Hall, Englewood Cliffs, N.J.

Umphress, D.A. and U.W. Pooch (1987), "A Goal-Oriented Approach to Simulation," In *Proceedings of the Conference on Methodology and Validation* (Eastern Simulation Conferences, Orlando, Fla., Apr. 6-9). Published as *Simulation Series 19*, 1 (Jan. 1988), 44-49. SCS, San Diego, Calif.

White, R.A. (1988), "TRAVIS: A General Purpose Traffic Intersection Visual Interactive Simulator," M.S. Thesis, Department of Computer Science, Virginia Tech, Blacksburg, Va., Nov.

Wyatt, D.L., S. Sheppard, and R.E. Young (1983), "An Experiment in Microprocessor-Based Distributed Digital Simulation," In *Proceedings of the 1983 Winter Simulation Conference* (Arlington, Va., Dec. 12-14). IEEE, Piscataway, N.J., pp. 271-277.

Yamamoto, Y. and M.R. Lenngren (1983), "Graphic Model Building System," In *Proceedings of the 16th Annual Simulation Symposium* (Tampa, Fla., Mar. 16-18). IEEE CS Press, Silver Spring, Md., pp. 161-175.

Zurcher, F.W. and B. Randell (1969), "Iterative Multi-Level Modelling – A Methodology for Computer System Design," In *Information Processing 68*, North-Holland, Amsterdam, pp. 867-871.

Peacock, J.K., J.W. Wong, and E. Manning (1979), "Distributed Simulation Using a Network of Processors," *Computer Networks 3*, 44-56.

Reddy, Y.V.R., M.S. Fox, N. Husain, and M. McRoberts (1986), "The Knowledge-Based Simulation System," *IEEE Software 3*, 2 (Mar.), 26-37.

Reynolds, P.F., Jr. (1982), "A Shared Resource Algorithm for Distributed Simulation," In *Proceedings of the Ninth Annual International Computer Architecture Conference* (Austin, Tex., Apr.). pp. 259-266.

Shannon, R.E. (1986), "Intelligent Simulation Environments," In *Proceedings of the Conference on Intelligent Simulation Environments* (San Diego, Calif., Jan. 23-25). Published as *Simulation Series 17*, 1 (Jan.), 150-156. SCS, San Diego, Calif.

Sokol, L.M., D.P. Driscoe, and A.P. Wieland (1987), "MTW: A Strategy for Scheduling Discrete Simulation Events for Concurrent Execution," MP-8700018, The MITRE Corporation, McLean, Va., Oct.

Stallings, W. (1985), *Data and Computer Communications*, Macmillan, New York.

Tanenbaum, A.S. (1981), *Computer Networks*, Prentice-Hall, Englewood Cliffs, N.J.

Tanenbaum, A.S. (1988), *Computer Networks*, 2nd edition, Prentice-Hall, Englewood Cliffs, N.J.

Umphress, D.A. and U.W. Pooch (1987), "A Goal-Oriented Approach to Simulation," In *Proceedings of the Conference on Methodology and Validation* (Eastern Simulation Conferences, Orlando, Fla., Apr. 6-9). Published as *Simulation Series 19*, 1 (Jan. 1988), 44-49. SCS, San Diego, Calif.

White, R.A. (1988), "TRAVIS: A General Purpose Traffic Intersection Visual Interactive Simulator," M.S. Thesis, Department of Computer Science, Virginia Tech, Blacksburg, Va., Nov.

Wyatt, D.L., S. Sheppard, and R.E. Young (1983), "An Experiment in Microprocessor-Based Distributed Digital Simulation," In *Proceedings of the 1983 Winter Simulation Conference* (Arlington, Va., Dec. 12-14). IEEE, Piscataway, N.J., pp. 271-277.

Yamamoto, Y. and M.R. Lenngren (1983), "Graphic Model Building System," In *Proceedings of the 16th Annual Simulation Symposium* (Tampa, Fla., Mar. 16-18). IEEE CS Press, Silver Spring, Md., pp. 161-175.

Zurcher, F.W. and B. Randell (1969), "Iterative Multi-Level Modelling – A Methodology for Computer System Design," In *Information Processing 68*, North-Holland, Amsterdam, pp. 867-871.