Contemplations of A Simulated Navel

Richard E. Nance

TR 88-35

Technical Report SRC 88-004

CONTEMPLATIONS OF A SIMULATED NAVEL

or

Recognizing the Seers Among the Peers

Richard E. Nance

Systems Research Center
and
Department of Computer Science
Virginia Tech
Blacksburg, VA 24061

January 1988

This research was supported in part by the Naval Sea Command under Contract No. N60921-83-G-A165-B030 through the Systems Research Center at Virginia Tech.

ABSTRACT

The Model Development Environment Project has the goal of defining the software utilities and the database support needed for creating, validating, and experimenting with complex simulation models. This project review, emphasizing the needs and explaining some of the guiding concepts and principles, serves to underscore key issues extending beyond discrete event simulation. An introspective summary presents an optimistic reaction to the fear that technically naive modelers might use the more sophisticated capabilities to produce catastrophic results.

CR Categories and Subject Descriptions: I.6.m [*Simulation and Modeling*]: Miscellaneous; D.2.6 [**Software Engineering**]: Programming Environments.

General Terms: Languages, Documentation

Additional Key Words and Phrases: Model development environments, rapid prototyping, interface progression

INTRODUCTION

The invitation to author this piece gave me wide latitude both in the choice and treatment of the topic. The decision to describe my perceptions of the technology for developing discrete event simulation (*des*) models in the 1990s, with some identification of the paths that have brought us to this juncture requires two immediate apologies. First, my apologies to Jim Henriksen for borrowing the theme of his excellent 1983 paper [Henriksen 1983]. Secondly, I apologize to the number of researchers and practitioners whose important works are not cited herein. Because of space limitations I have limited the description to the experiences of a single research project hopefully in a way that avoids the implication that this project has progressed without knowledge of the work of others. At times this progress appeared to stand on the shoulders of others; sometimes, only on the insteps. In either case I am admittedly committing the dangerous path of extrapolating from the specific to the general: projecting the state of the world based on an assessment of my navel; ergo the title.

Recognizing that brevity is not only "the soul of wit" but also the cornerstone of effective communication, I have organized this paper along the classical six interrogatives: *who, when, where, what, how,* and *why.* Emphasis is given to the latter three, which convey the technical essence of *des* modeling technology. A final section is devoted to brief speculation about the art and science of modeling in the 1990s.

WHO?

The Model Development Environment (MDE) Project is the quickest answer to the question, *Who are we?*, but that answer takes a dynamic form. The two constant members of the MDE Project are Professor Osman Balci and myself. Professor Balci is a truly dedicated researcher, and our work together has been a source of both personal and professional satisfaction. Professor C. Michael Overstreet of Old Dominion

University, during periods of project support and even when not directly supported by the project, has contributed very significantly to the concepts and principles reported in numerous publications. Significant contributions have also come from several creative graduate students over the four years of the project existence: Lynne F. Barger, E. Joseph Derrick, Valerie Frankel, Robert H. Hansen, Robert L. Moose, Jr., and Jack C. Wallace. Undergraduate students Charles W. Box, Matthew C. Humphrey, and David Maynard have also made contributions.

Representatives of the Navy sponsors have provided invaluable advice and direction, most prominently by Harry E. Crisp, Stephen W. Parker, David E. McConnell, and Daniel T. Green (Naval Surface Warfare Center), Philip Andrews and Jay D. Pastine (Naval Sea Systems Command), David Kaplan (Naval Research Laboratory), CAPT Karl Duff, CDR Marvin Langston, and Raghu Singh (Space and Naval Warfare Systems Command). Finally, the project has benefitted from the critical comments and advice rendered by visitors from other universities: Robert M. O'Keefe (University of Kent at Canterbury, UK) and Stephen C. Mathewson (Imperial College, UK).

WHEN AND WHERE?

The MDE Project began in 1983, although seminal research leading to the work was initiated in 1977 through the support of the National Bureau of Standards. A Science Research Council fellowship in 1980 permitted me to attend research presentations in the UK and to hold personal discussions with prominent British researchers, such as Alan T. Clementson (Birmingham), Stephen C. Mathewson, E. Fiddy (British Leyland), and John G. Crookes (Lancaster). C. M. Overstreet's dissertation research during 1977-1982 originated many of the key ideas and conjectures, resolving the correctness of a few of the most significant.

For the most part, the work has been conducted at the MDE Research Laboratory

and the Systems Research Center at Virginia Tech. The Laboratory houses a SUN-3/160* workstation, with MC68020 processor and four megabytes of main storage. The workstation uses a color monitor and 380 megabytes of peripheral storage. The UNIX** 4.2BSD operating system hosts an environment of model development tool prototypes, with database support provided by INGRES.

Early tool prototypes were developed on a VAX 11/785, and certain tasks continue to utilize these versions and to test ancillary concepts or explore alternative design decisions. The SUN workstation and the VAX communicate via an Ethernet installation that permits each to be accessed from remote sites.

WHAT?

The MDE Project has the objective of defining the tools needed by *modelers* for the development and continuing support of large, complex models. The emphasis on "modelers" is meaningful from two perspectives:

(1) the tools are not designed strictly for programmers but for the users with modeling knowledge and responsibilities, and

(2) the plural form indicates a task of sufficient size that a team or group effort is typical.

The first perspective is motivated by the need to eliminate the communications cost incurred in the problem and solution translation currently required between a modeler and a programmer. The second simply recognizes that the sharing of ideas, understanding, techniques, and eventually model components depends on *communications.*

*SUN-3/160 is a registered trademark of Sun Microsystems Inc.
**UNIX is a registered trademark of AT&T Bell Laboratories.

Our definition of "model development environment" is an integrated set of hardware and software tools that enables accomplishment of the objectives cited above. More specifically, a MDE should provide [Balci 1986, p.53]:

(1) cost-effective integrated and automated support of model development throughout the life of the model,

(2) improved model quality by assisting in the quality assurance of the model,

(3) increased efficiency and productivity of the modeling team through computer-assisted model creation and verification, and

(4) reduced development and redevelopment effort and time.

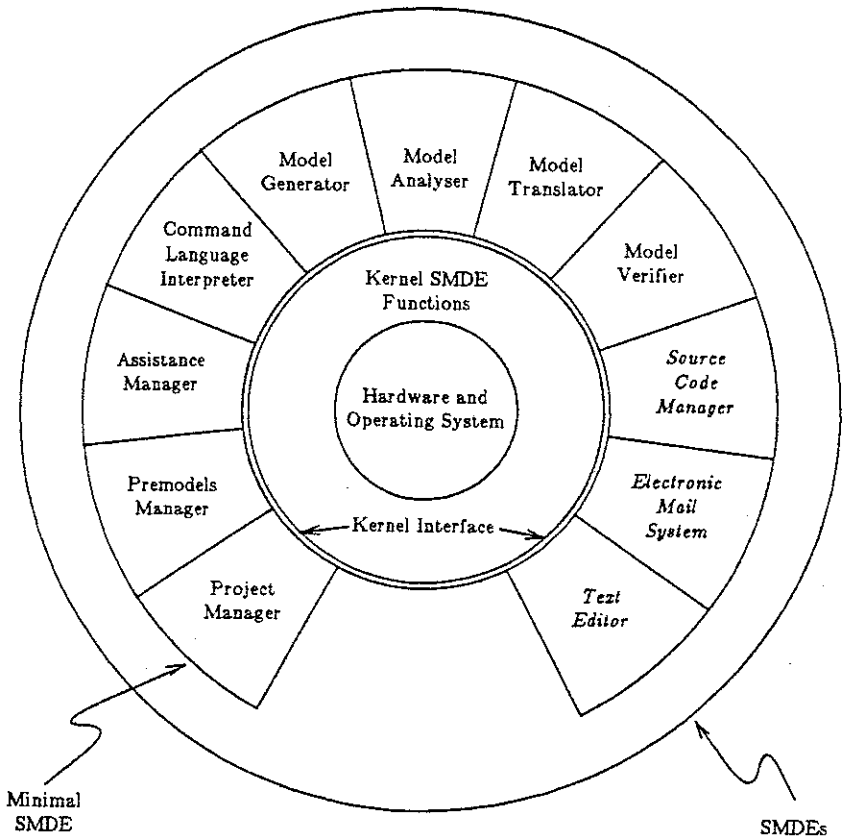

Figure 1. Simulation Model Development Environment: Prototype Architecture

Figure 1 shows the tools that comprise the minimal set for model development. This set, and the functionality represented by them, is not limited to discrete event simulation, but is intended to broadly serve the *general modeling task.* We assert that while the specific form of assistance and the procedures by which it is rendered can vary depending on problem-solving techniques, the assistance function remains invariant. For example, the form of assistance rendered by the Model Generator in the construction of a linear programming model might begin with the functions:

(1) Prompt the modeler to state and justify the assumptions underlying the specific application.

(2) Respond to the modeler with definitions of terms from the assumptions that are found in a lexicon for that environment.

(3) Request that definitions be given for terms not found in the lexicon.

(4) Lead the modeler through the definition of the variables comprising the objective function, prompting throughout for information such as the value type, permissible range of values, and dimensions (units of measurement).

The first three appear to be common to any modeling task; they are invariant. An environment for continuous simulation, discrete event simulation, mathematical programming, or any other solution technique would provide these functions. Only at the level represented by (4) do we encounter technique-specific knowledge acquisition.

I have been encouraged by recent correspondence with Arthur Geoffrion, developer of the structured modeling concepts [Geoffrion 1987], to perceive the *similarities* existing within problem-solving approaches originally stimulated by different technique domains. Both Professor Geoffrion and I were surprised by the degree of overlap in the stated

objectives for his structured modeling and my conical methodology, the former originating within mathematical programming, the latter, discrete event simulation.

An in-depth description of each tool is given in [Balci 1986]. A brief sketch along functional lines, aided by imposing an imaginary clock face over Figure 1, follows:

The three managers (from 7 to 9 o'clock) are utilities for three databases: *Project* provides data and documentation details for project management; *Premodels* permits inquiries regarding model components already developed; and *Assistance* furnishes user help for tools, methodology, and applications.

The *Command Language Interpreter* provides the user interface with the environment.

The *Model Generator*, through interactive dialog, enables the user to produce a model specification.

The model specification is examined by the *Model Analyzer* which through iteration with user and Model Generator leads to a refined specification with (hopefully) few errors.

The refined model specification is submitted to the *Model Translator* to produce an executable model implementation.

Dynamic diagnosis to assure the correctness of the model implementation is performed by the *Model Verifier.*

The three tools in the 3 to 5 o'clock position are provided for our prototype environment by the UNIX operating system and can generally be supplied by the host operating system.

HOW?

The design philosophy of *rapid prototyping* (see [Jenkins 1983] for one characteriza-

tion of the approach) has been employed in the development of the modeling tool set illustrated in Figure 1. By defining the required functionality for each tool and the means by which each tool communicates with all others, we preserve the flexibility and design independence for each tool prototype without sacrificing the all-important *integration* property. Requiring all communication among tools through the Kernel Interface promotes insularity at the highest level: a new prototype should plug into the existing environment with no discernible effect on the remainder.

WHY?

No doubt, this is the key question. Why the need to pursue an alternate and different way of creating simulation models and using them? Don't we have enough simulation programmming languages now?

The Narrow Answer

A direct, if perhaps narrow answer, is provided by the General Accounting Office report on the problems with cost overruns, inordinate delays and user dissatisfaction in a study of 57 federally funded models [USGAO 76]. Although 12 years old now, does anyone seriously believe that those problems have been eliminated? Do many persons believe that the productivity of simulation modelers and analysts has been significantly improved over the past 12 years?

We count ourselves among the group who *do* believe that the productivity of simulation modelers *has* improved significantly over this period. However, the advances have come in selected *application* domains through the creation of high level language dialects such as SIMFACTORY* or NETWORK II.5* based on SIMSCRIPT II.5*. During the

*SIMSCRIPT II.5, SIMFACTORY and NETWORK II.5 are registered trademarks of CACI, INC.

last two Winter Simulation Conferences tutorials were presented for at least six "languages or systems" for simulating manufacturing applications.

Removing the application domains of manufacturing automation and networks from consideration, have the gains in productivity been significant? Our answer, regrettably is "no". Thus, the MDE Project seeks to alleviate the effects, if not eliminate some, of the serious problems cited in the GAO report.

A Globally Philosophical Response

The emergence of a plethora of manufacturing and network dialects is a natural response of those marketing SPLs and related services. Recognizing the heavy demand from a particular market area, the vendor seeks to gain a competitive advantage. However, this strategy has its limitations. Already, we find an extension of the network dialect to particular types of network simulations -- COMNET II.5* for telecommunications networks. Does the future portend dialects for transportation networks, petroleum pipeline networks, and retail distribution networks? Where does the perceived cost of user abstraction give way to the restrictions of hard-coded interpretation and the ongoing cost of learning and maintaining *one more dialectal tool*?

Lest we seem too critical, the *natural* response above is but a consequence of a 35-year trend that might be labeled the *interface progression.* In clarification, the past 35 years in computing technology have witnessed a continuing progression of language developments that relieve the computer user from the requirement of detailed knowledge of the underlying computer system: mnemonic assemblers replacing basic machine language, in turn replaced by high level languages such as FORTRAN and COBOL, then problem solving languages such as SIMSCRIPT and ACSL, then non-procedural

*COMNET II.5 is a registered trademark of CACI, INC.

languages, and so on.

The distinctive characteristic of the MDE Project is that the vantage point is not from "the machine" (the execution environment) looking toward the user. Rather, the vantage point is from the user (and users' needs) peering toward the machine. The emphasis is on *translating the user's concepts into a communicative form* ignoring whether that form is procedural, let alone executable. Execution is a deferred goal. Computer-assisted analysis must be invoked as early in model development as possible, certainly prior to an executable representation.

The important differences in the two vantage points emerge in the implications for model development where logical representations (easily understood by modelers) must be transformed into physical forms (efficiently executed by machines). Application-specific dialects force this transformation (*binding* is the term in compilers) early; while the MDE philosophy delays the binding until much later. Of even greater consequence, the MDE philosophy allocates the binding decision (both time and form) to the user rather than the marketer.

INTROSPECTIVE SUMMARY

Through this brief, stylized description of a single project, I have sought to describe the leading edge of research in discrete event simulation model development. Emphasis is given to explaining the need for a very different form of model creation than is typical of current efforts. Such an explanation relies on an extension of concepts such as *automated diagnosis*, *interface progression*, and the distinct separation of specification from implementation. Principles such as *delayed binding*, *specification derived documentation*, and *abstraction* are important in understanding where the technology is going.

This perceived path in simulation modeling technology is not without its perils, and

here I suspect that the picture is quite similar for both the continuous and discrete event domains. The most disturbing problem arose in discussion at the 1987 Winter Simulation Conference:

> *What potential catastrophic results can be produced by users given modeling tools so sophisticated that they need know nothing about the internal workings or the underlying technology?*

Some in the audience cautioned that a "Frankenstein" could be a possibility. I am not so alarmed by this prospect.

Consider the parallel with automotive technology. For many years drivers with no knowledge of the mechanical composition of an automobile have used the tool effectively, only infrequently stalled by their lack of knowledge. Yet, in the early years of the automobile, a mechanics certification, obtained after passing a course, was a prerequisite to gaining an operator's license. Increasingly, the computer user of today knows less and less about the inner workings of either the software (the operating system) or the hardware. Sometimes the user suffers disruptions and discomfort because of this lack of knowledge, but the these difficulties are perceived as "the price imposed." Pursuing the automotive analogy further: have you noticed the difficulty in finding a systems programmer that you can trust?

REFERENCES

1. Balci, O. 1986. "Requirements for Model Development Environments", *Computers & Operations Research*, **13**, 53-67.
2. Geoffrion, A. M. 1987. "An Introduction to Structured Modeling", *Management Science*, **33**, no. 5 (May): 547-588.
3. Henriksen, J. O. 1983. "The Integrated Simulation Environment (Simulation Software of the 1990s)", *Operations Research*, **31**, no. 6 (Nov-Dec): 1053-1073.
4. Jenkins, A. M. 1983. "Prototyping: A Methodology for the Design and Development of Application Systems", Discussion Paper #227. School of Business, Indiana University, Bloomington, April.
5. U.S. General Accounting Office, 1976. "Report to the Congress: Ways to Improve Management of Federally Funded Computerized Models", LCD-75-111, Washington,

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Limited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) Systems Research Center Blacksburg, Virginia 24061 SRC 88-004	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Systems Research Center 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Naval Surface Warfare Center
6c. ADDRESS (City, State, and ZIP Code) Virginia Tech 320 Femoyer Hall Blacksburg, Virginia 24061	7b. ADDRESS (City, State, and ZIP Code) Dahlgren, Virginia 22448
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Naval Surface Warfare Center 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N60921-83-G-A165 B030
8c. ADDRESS (City, State, and ZIP Code) Dahlgren, Virginia 22448	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Contemplations of a Simulated Navel or Recognizing the Seers Among the Peeres

12. PERSONAL AUTHOR(S)
Richard E. Nance

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Interim Research Report	FROM _____ TO _____	January 1988	12

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The Model Development Environment Project has the goal of defining the software utilities and the database support needed for creating, validating, and experimenting with complex simulation models. This project review, emphasizing the needs and explaining some of the guiding concepts and principles, serves to underscore key issues extending beyond discrete event simulation. An introspective summary presents an optimistic reaction to the fear that technically naive modelers might use the more sophisticated capabilities to produce catastrophic results.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL