Simulation Model Development Environments: A Research Prototype

Osman Balci
Richard E. Nance

TR 86-20

Technical Report SRC-86-004†

SIMULATION MODEL DEVELOPMENT ENVIRONMENTS: A RESEARCH PROTOTYPE‡

Osman Balci
Richard E. Nance

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

20 August 1986

† Also cross referenced as TR-86-20, Department of Computer Science.

‡ Submitted to the *Journal of the Operational Research Society* (published in England) for publication. Spelling conforms with British English.

ABSTRACT

The needs for automated assistance in the simulation task are undeniable. The size and complexity of current modelling efforts far exceed the bounds considered challenging only a decade ago. A research prototype of a simulation model development environment (SMDE) has been developed to provide an integrated and comprehensive collection of computer-based tools to (1) offer cost-effective, integrated, and automated support of model development throughout its entire 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 prototype SMDE is composed of four layers: (0) hardware and operating system, (1) kernel SMDE, (2) minimal SMDE, and (3) SMDEs. Currently, a SUN 3/160 colour computer workstation and the UNIX operating system constitute layer 0 upon which an advanced prototype SMDE is being developed. The objectives of this paper are to: (1) describe the current state of research on simulation support environments, (2) use the SMDE research prototype as an example for describing concepts and principles employed, experiences gained, and guidelines (potential principles) derived in the design and creation of the prototype, and (3) identify and speculate on future research directions.

Key words: computer systems, modelling, simulation

INTRODUCTION

The advent of digital computers has made simulation a powerful problem-solving technique in many disciplines. Advancements in computing technology (both hardware and software) have played an important role in advancing the state-of-the-art of simulation. A meaningful chronology of simulation software[1] traces the evolution through five periods: (1) the early era of custom programs — 1955-60, (2) the period of emergence of simulation programming languages (SPLs) — 1960-65, (3) the second generation of SPLs — 1966-70, (4) the era of extended features — 1971-78, and (5) the current period.

Progress in software support for a simulation study has unfortunately not kept up with the incredible pace in computing technology. Despite phenomenal advances in computer hardware and software over recent years, simulation still remains a labour intensive, error prone, and costly technique especially for large complex simulation studies. Automated support of a simulation study *throughout its entire life cycle* is undeniably needed to confront the current modelling problems identified by Balci.[2] The automated support for modelling and simulation is characterised in the form of a simulation model development environment (SMDE).

The life cycle of a simulation study[3,4] is characterised in terms of 10 phases, 10 processes, and 13 credibility assessment stages (CASs). The work described herein focuses on automated support over the phases, processes, and CASs of the life cycle corresponding to model development. Automated support for the remaining portions is related to a simulation management system[5,6] which is envisioned to cover the entire life cycle.

The remainder of this paper is organised to meet the following objectives: (1) to describe the current state of research on simulation support environments, (2) to use the SMDE research prototype as an example for describing concepts and principles employed, experiences gained, and guidelines (potential principles) derived in the design and creation

of the prototype, and (3) to speculate on future research directions.

REVIEW OF RELATED WORK

Currently, simulation software development is in a significant transition period. While several factors characterise the transition, Nance[1] points out the following three as most obvious: (1) a shift from the *program* to the *model* view of the simulation process, (2) interest in and commitment to the development of support tools, and (3) the legacy of a concept/language impedance from the insularity promoted by SPLs. Several current research projects represent differing perceptions of the goal terminating this transition period or varying paths toward a similar goal.

The Computer Aided Simulation Modelling (CASM) project at the London School of Economics is investigating ways of making the process of simulation modelling more efficient. Balmer and Paul[7] provide an overview of the CASM environment in which the problem formulator, interactive simulation program generator (ISPG), and output analyser are the essential components. The CASM relationship with artificial intelligence (AI) and some work on the output analyser are described by Balmer.[8] Doukidis and Paul[9] and Paul and Doukidis[10] report on early and more recent attempts at automating simulation problem formulation using expert systems and natural language understanding systems. An ISPG called LANGEN has been developed based on the extended Lancaster Simulation Environment[11] and is described by Chew *et al.*[12]

Reddy *et al.*[13] describe a Knowledge-Based Simulation system (KBS) which provides facilities for interactive model creation and alteration, simulation monitoring and control, graphics display, and selective instrumentation. KBS uses a schema representation language[14] — an AI-based knowledge representation system for modelling — and combined object- and rule-based programming to represent simulation knowledge and behaviour.

Baskaran and Reddy[15] present an introspective environment for knowledge based simulation to learn/understand the dynamics and the underlying causality of the system being modelled so that the information gained can be used in automatic model verification and post analysis.

McArthur *et al.*[16] present an overview of the Rand Object-oriented Simulation System (ROSS) which provides a programming environment in which users can design, test, and modify large knowledge-based simulations of complex mechanisms. ROSS is an English-like, interactive, object-oriented language implemented in Lisp. A simulation in ROSS can be interrupted while it is running, the state can be queried or the code modified, and the simulation can then be resumed. Simulating Warfare In the ROSS Language[17] (SWIRL) and Tactical Warfare In the ROSS Language[18] (TWIRL) are the two major applications of ROSS.

Unger *et al.*[19] describe a distributed software prototyping and simulation environment called JADE. JADE is structured in four levels: hardware, kernel, programming, and prototyping. It provides tools for the design, implementation, debugging, testing, maintenance, and performance analysis of distributed, concurrent programs.

Birtwistle *et al.*[20] describe an animated discrete event simulation environment called ANDES. ANDES is being constructed as a part of JADE. It provides animated pictures of the state of a simulation, a wide spectrum of perspectives ranging from gross model logic to views of individual processes, and assistance in establishing model credibility.

Standridge *et al.*[21] present an overview of The Extended Simulation System (TESS) which integrates simulation, data management and graphics capabilities. TESS has evolved through the integration and enhancement of, and providing a common interface for, four previously developed packages: SLAM II — a Simulation Language for Alternative Modelling; SDL — Simulation Data Language; AID for input data analysis; and SIMCHART for

output analysis.

Reese and Sheppard[22] give an overview of a simulation software development environment (SSDE) which is composed of a series of data bases and a language. SSDE's framework provides the means of hiding the file system and development support tools and automating the transition between the different phases of development. The prototype SSDE includes several tools developed specifically to support simulation programming.

Zeigler and De Wael[23] describe the current state of a project to implement a support environment for the multifaceted modelling methodology. Objects and relations defined in the methodology are represented by a knowledge base to guide the modeller in the use of tools to manipulate and validate models, store and retrieve them from model bases.

Ören and Aytaç[24] describe the architecture of a knowledge-based modelling and simulation system called MAGEST.

Robertson[25] reports on the design of an intelligent, rule based simulation environment which consists of an inference engine, simulation knowledge base, simulated world state database, knowledge editor, simulation toolbox, simulation browser, and graphical presenter. A number of sources, although not as comprehensive as the above, describe related topics. Wales and Luker[26], following the paradigm of program generators, use Ada-implemented tools to produce simulation programs from activity-like diagrams. Lehmann *et al.*[27] propose a general approach to a multi-stage, hierarchical modelling concept. Henriksen[28] speculates that the simulation practitioners of the future will work in an environment comprised of well-integrated software tools. Ruiz-Mier *et al.*[29] speculate on the use of a network simulation language in an AI-based programming environment.

SMDE RESEARCH PROTOTYPE

Since June 1983 the MDE project has addressed a complex research problem: prototyping of a discrete event SMDE 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. Guided by the fundamental requirements identified by Balci[2], the Conical Methodology[4] (CM) has furnished the architectural underpinnings of the SMDE research prototype. Prototypes of SMDE tools have been developed using the rapid prototyping technique.

Conical Methodology

The CM, developed specifically for simulation modelling tasks, prescribes a top-down model definition followed by a bottom-up model specification. The model definition phase requires the modeler to perform an object decomposition, assigning attributes to objects based on the system being described and the objectives of the simulation study. Attributes are strongly typed, and the CM advocates extraction of the maximum information from the modeler during the definition phase, e.g., attribute dimensions and value range definition in addition to the attribute typing.

The model specification phase utilises the basically static representation generated in the definition phase to produce a dynamic representation. Information extracted from the modeler is used to construct a bottom-up specification of the changes in attribute values (and, consequently, object states). The intent of the methodology is to structure an iterative construction process that admits diagnosis of model representations early in the model development life cycle.[30] The advantages of early error detection and correction, prior to the implementation in a SPL, are obvious.

Documentation occupies a central role in the CM. The iterative representational versions of the model form increasingly more procedural and less abstract descriptions of the target of the modelling team. Hierarchical decomposition encourages, even forces, a stratification of object descriptions that address the differing informational concerns of project manager, simulation analyst, and programmer (see reference 6).

Rapid Prototyping

Rapid prototyping is a technique for rapid construction of comparatively inexpensive and incomplete experimental versions of a target system. Lessons learned from a predecessor prototype enable improvements in the successors.[31] Brooks[32] originated the idea and advocated to "plan to throw one away; you will, anyhow." When ideas or requirements cannot be specified completely and accurately, a prototype is developed rapidly and cheaply (without much attention to efficiency or polish) for the purpose of testing the ideas or requirements and accelerating the learning process. In light of the knowledge gained, the prototype may be thrown-away or refined. (Even if the product is discarded, the knowledge retained is the actual benefit.) Based on this knowledge, a new prototype is constructed and experimented with, and compared against target specifications. The process is repeated until a satisfactory prototype is obtained.

The prototype SMDE is architectured in four layers as depicted in Figure 1: hardware and operating system, kernel SMDE, minimal SMDE, and SMDEs. Each layer is described in some detail below.

Layer 0: Hardware and Operating System

A SUN-3/160† colour computer workstation running under MC68020 CPU with 4 megabytes of main memory, 380 megabytes of disk subsystem, a 1/4-inch cartridge tape

† SUN-3/160 is a trademark of Sun Microsystems, Inc.

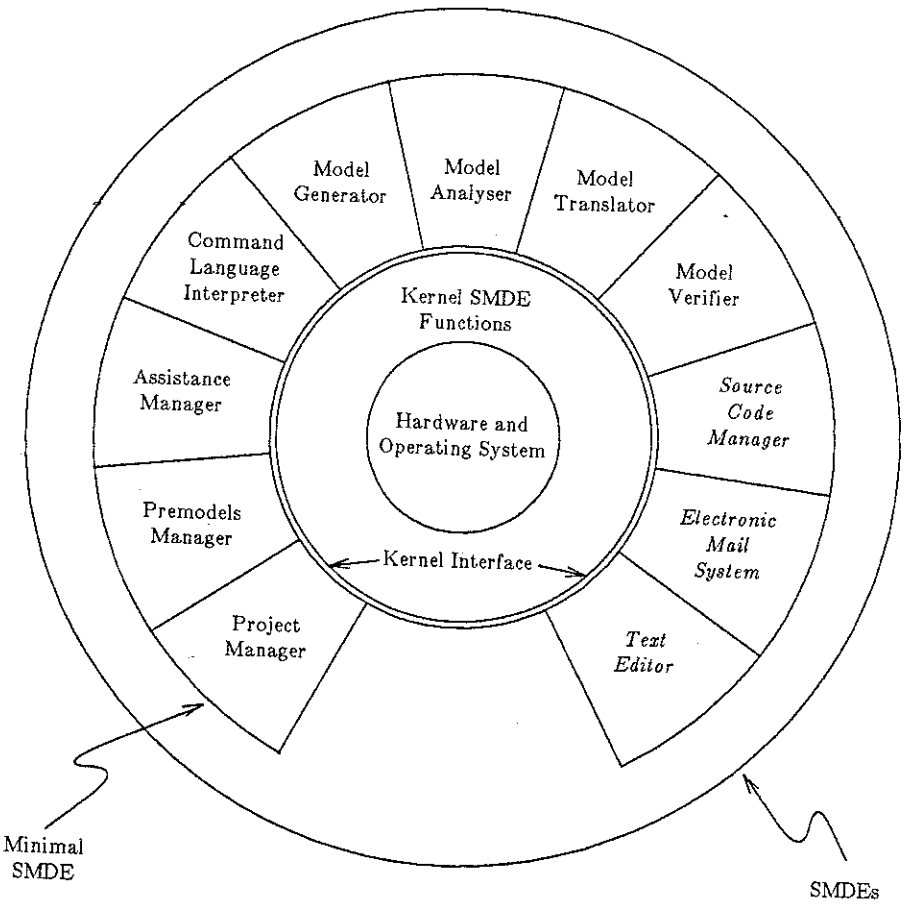

Fig. 1. *The architecture of the SMDE research prototype.*

drive, and a 19-inch monitor with 1152×900 pixel resolution constitute the hardware of the prototype SMDE. A laser printer and a line printer over the Ethernet local area network serve the SMDE for producing high quality documents and hard copies of SUN screens and files.

The UNIX‡ 4.2BSD operating system and utilities, multiwindow display manager (SunWindows), device independent graphics library (SunCore), computer graphics interface (SunCGI), visual integrated environment (SunView), Sun programming environment (SunPro), and INGRES relational database management system (SunINGRES) constitute the

‡ UNIX is a trademark of AT&T Bell Laboratories.

software environment upon which the SMDE is built. Nance *et al.* have evaluated the capabilities of UNIX for hosting an earlier prototype SMDE, noting major and minor deficiencies.[33]

The current prototype, based on the SUN workstation, eliminates certain deficiencies in the earlier version and reduces the negative effects of others. Figure 2 provides a "flavour" of the supporting technology introduced by the SUN workstation.

Layer 1: Kernel Simulation Model Development Environment

Primarily, this layer integrates all SMDE tools into the software environment described above. It provides SunINGRES databases, communication and run-time support functions, and a kernel interface. There are three SunINGRES databases at this layer labeled project, premodels, and assistance, each administered by a corresponding manager in Layer 2. All SMDE tools are required to communicate through the kernel interface. Direct communication between two tools is prevented to make the SMDE easy to maintain and expand. The kernel interface provides a standard communication protocol and a uniform set of interface definitions. Security protection is imposed by the kernel interface to prevent any unauthorised use of tools or data.

Layer 2: Minimal Simulation Model Development Environment

This layer provides a "comprehensive" set of tools which are "minimal" for the development and execution of a model. "Comprehensive" implies that the toolset is supportive of all model development phases, processes, and CASs. "Minimal" implies that the toolset is basic and general. It is basic in the sense that this set of tools enables modelers to work within the bounds of the minimal SMDE without significant inconvenience. It is general in the sense that the toolset is generically applicable to various simulation modelling

Fig. 2. *The SUN computing technology.*

tasks.

Minimal SMDE tools are classified into two categories. The first category contains tools specific to simulation modelling: Project Manager, Premodels Manager, Assistance Manager, Command Language Interpreter, Model Generator, Model Analyser, Model Translator, and Model Verifier. The second category tools (also called assumed tools or library tools) are expected to be provided by the software environment of Layer 0: Source Code Manager, Electronic Mail System, and Text Editor.

Project Manager is a tool which (1) administers the storage and retrieval of items in the project database, (2) keeps a recorded history of the progress of the project, (3) triggers messages and reminders (especially about due dates), and (4) responds to queries in a prescribed form concerning project status.

Premodels Manager is a tool which (1) implements a query language, (2) administers the premodels database, (3) provides information on previous modelling projects, and (4) provides a stratified description and several representational forms of models or model components developed in the past.

The current research prototype of the *Assistance Manager* administers the Sun-INGRES assistance database and possesses user and programmer interfaces. The user interface provides: (1) information on how to use a SMDE tool, (2) a glossary of technical terms, (3) introductory information about the SMDE, and (4) tutorial assistance for SMDE tools and methodologies, techniques, and procedures used within the SMDE. The user interface provides "help" in two modes: local (programmer-activated) and global (user-activated). The programmer interface is intended for tool developers for the storage and retrieval of assistance-related information in the database. The Assistance Manager also provides the capability for automatic hardcopy generation.

Command Language Interpreter (CLI) is the language through which a user invokes a

SMDE tool. Early in our project, the CLI was prototyped based on the proposal of Moose[34] and was fully described by Humphrey.[35] Later, after the acquisition of the SUN workstation, the CLI was replaced by SUN's window management system.

Model Generator (the simulation model specification and documentation generator) is a tool which assists the modeler in: (1) creating a model specification in a predetermined form which lends itself for formal analysis, (2) creating multi-level (stratified) model documentation, and (3) model qualification. The Model Generator is intended to support the model formulation and model representation processes, beginning with a conceptual model from the system and objectives definition and extending through several communicative models evolving from a conceptual model. Therefore, it is the most important and the most complex tool of the SMDE.

Two research prototypes of the Model Generator have been developed under the guidance of the CM.[4] The first produces a primitive model specification[36,37] that is general but very difficult to fully translate into an executable code. Hansen[38] provides a description of an early version. The second produces a specification which is less general but which lends itself to full translation.

Model Analyser diagnoses the model specification created by the Model Generator and effectively assists the modeler in communicative model verification. The first prototype is described by Moose and Nance.[39] The current research prototype implements a control and transformation metric for measuring model complexity[40] and provides diagnostic assistance using digraph representations of discrete event simulation model specifications.[30]

Model Translator translates the model specification into an executable code after the quality of the specification is assured by the Model Analyser.

Model Verifier is intended for the programmed model verification. Applied to the executable representations, it provides: (1) assistance in incorporating diagnostic measures

within the source program, (2) a cross-reference map to identify where a particular variable is referenced and where its value is changed, (3) a chart of the programmed model control topology to indicate subprogram invocation relationships, and (4) dynamic analysis procedures for snapshots, traces, breaks, statement execution monitoring, and timing analyses.

Source Code Manager is a tool which configures the run-time system for execution of the programmed model, providing the requisite input and output devices, files, and utilities.

Electronic Mail System facilitates the necessary communication among people involved in the project. Primarily, it performs the task of sending and receiving of mail through (local or large) computer networks. The SUN workstation's MailTool (icon shown in the upper right of Figure 2) is used to communicate with other nodes in our local area network from which large computer networks (e.g., ARPANET, CSNET, BITNET, etc.) can be accessed.

Text Editing is provided by the VI editor and the SUN workstation's TextEditor. Both of these tools are used for preparing technical reports, user manuals, system documentation, correspondence, and personal documents.

Layer 3: Simulation Model Development Environments

This is the highest layer of the environment, expanding on a defined minimal SMDE. In addition to the toolset of the minimal SMDE, it incorporates tools that support specific applications and needed either within a particular project or by an individual modeler. If no other tools were added to a minimal SMDE toolset, a minimal SMDE would be a SMDE.

The SMDE tools at layer 3 are also classified into two categories. The first category tools include those specific to a particular area of application. These tools might require further customising for a specific project, or additional tools may be needed to meet special requirements. The second category tools (also called assumed tools or library tools) are those

expected to be available due to their availability and use in several other areas of application. A tool for statistical analysis of simulation output data, a tool for designing simulation experiments, a graphics tool, a tool for animation, and a tool for input data modelling are some example tools of layer 3.

A SMDE tool at layer 3 is integrated with other SMDE tools and with the software environment of layer 0 through the kernel interface. The provision for this integration is indicated in Figure 1 by the opening between Project Manager and Text Editor.

CONCEPTS AND PRINCIPLES EMPLOYED

All too often in software-intensive projects, the perceived need for a specific tool leads to its creation and use within that project only. After some repeated experiences or perhaps after observing a support utility in use by another project, the motivation arises to add tools and expand programming support. At this point the serious implications of ad hoc, independent development become apparent. After all, *tool development is still software development*, and incompatibilities are even more likely to abound because of the more intimate association with the host operating system.

The SMDE prototype has benefited from the clear, overriding presence of the CM, which provides the underlying philosophical guidance necessary to achieve an *integrated* suite of tools. The concepts and principles advocated by the CM are identified below and related to similar principles in software development, engineering design, and systems analysis. These concepts and principles have been utilised during the design and creation of the SMDE research prototype.

Division of concerns

Related to the "separation of concerns" generally attributed to Dijkstra[41], the division of concerns is enforced by a partitioning of the model development task into a model definition phase followed by a model specification phase. The modeler must define an object before undertaking the specification. The definition phase focuses on the characterisation of an object through the identification and typing of attributes. In essence, the definition phase produces a *static* model description, which serves as input to the specification phase that produces a *dynamic* description. The separation of model development from model execution and the insistence on a representation devoid of specific SPL terminology is indicative of the division of concerns also.

Hierarchical decomposition

Also referred to as top-down decomposition, the principle of hierarchical decomposition imposes a stratification during the model definition phase. At the model stratum, modelling objectives, assumptions, terminology, and the initialisation and termination requirements are of interest in describing the "model object." Attributes of the "model object" are defined, and the decomposition into subobjects proceeds with the association of attributes describing them. Any form of decomposition supports the modularity principle of software design; thus, modularity is achieved in the model representation.

Progressive elaboration

No distinction is made in the software development literature between progressive elaboration and iterative refinement. Both principles are included in the latter term. However, progressive elaboration is a principle more supportive of the rapid prototyping paradigm. The difference between the two principles is illustrated in Figure 3. In Figure 3, iterative

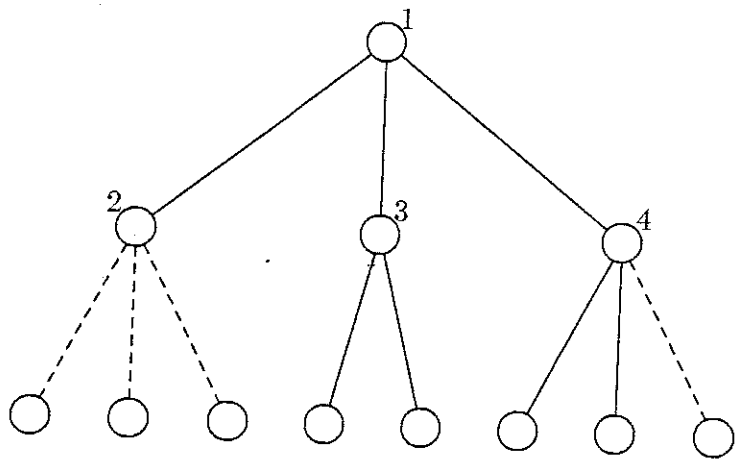

Fig. 3. *Progressive elaboration and iterative refinement.*

refinement is observed in the *expansion* of description (more detail) of object 2. Progressive elaboration is illustrated in the *addition* of an object (perhaps increasing the functional description) under object 4.

Iterative refinement

The capability for expanding the description of an object by increased details is termed *iterative refinement.* Additional information is given under the previous subsection. Essentially, the principle of iterative refinement assures the continuing capability for expanding the levels of hierarchical decomposition by the creation of objects. Progressive elaboration assures the addition of objects to an existing level (no addition of levels).

Documentation through specification

The principle of inseparability of documentation from specification is akin to the principle of *concurrent documentation* described by Tausworthe.[42] The intent is to incorporate the facilities for producing model documentation within the specification task so that documentation loses the onerous characterisation of "after the fact" and remains more current

and accurate.

Information hiding

Parnas[43] enunciated the principle of information hiding for software as a means to reduce the "ripple effect," where a change in one code component leads to "fixes" in many other components. The CM utilises this principle in the admonition to the modeler: push details to the lowest possible level. Following this principle minimises the interdependencies among modules.

Life cycle verification

The early use of diagnosis techniques and the application of verification procedures to communicative models is a cornerstone of the CM. Identical to the life cycle verification principle of software engineering, the purpose is to expose modelling errors as soon as possible and not delay all error detection/correction until execution.

EXPERIENCES GAINED AND GUIDELINES DERIVED

Early prototypes in the MDE research project relied on a VAX 11/785† running EUNICE (an emulation of UNIX under the VAX/VMS operating system) to provide layer 0 of the SMDE. Using this system through "dumb" terminals revealed inherent limitations for the prototyping efforts. The lack of a window management system restricted the effectiveness of multitasking in UNIX and the development of an acceptable modeler-computer interface. Our experience shows that the state-of-the-art computing technology as provided by the SUN workstation is crucial for rapidly prototyping a SMDE.

The appropriateness of the rapid prototyping technique has been justified. However,

† VAX 11/785 is a trademark of Digital Equipment Corporation.

emphasis should be given to knowledge production in contrast with software production as the goal of rapid prototyping. Although a prototype may be thrown-away, the knowledge gained from making it is retained. We share the caution expressed by Taylor and Standish[31] that "it is often sociologically disastrous to the morale of the designers and implementers who are forced to discard their previous work and are made to feel that their accomplishments may have little permanent value." Project participants must understand the utility of rapid prototyping. In addition, the knowledge gained from a prototype and the essential versions of a prototype must be fully documented so that later participants can continue the work with minimal difficulty.

Our experience with rapid prototyping reveals that modifiability is the most important quality characteristic of a rapid prototype. Customarily, a program is developed to be efficient and concise; however, in rapid prototyping one should relegate efficiency and conciseness to subordinate objectives. A rapid prototype should: (1) lend itself to easy modification, (2) be a working tool that can be experimented with, (3) be relatively cheap to develop, (4) be developed relatively quickly, and (5) provide only the relevant functionality and ignore unnecessary details.

FUTURE RESEARCH DIRECTIONS

An advanced prototype of a discrete event SMDE is the end-product of the current research. This prototype is expected to provide significant experience, serving as the basis for the next phase of the ongoing research. Construction of an experimental installation prototype SMDE forms the next step, involving (1) the analysis of the tradeoffs among design alternatives based on the evaluation of the advanced prototype, (2) design and implementation of the installation prototype SMDE, and (3) human engineering of the installation prototype SMDE.

AI techniques are used in the existing prototypes, particularly in the diagnostic procedures employed in the Model Analyser.[30,44] Expert systems applications are anticipated in future prototypes of other tools, with the result being more *intelligent* simulation environments/systems. Shannon[45] points out that "it is strongly believed that such [intelligent simulation] systems will be domain specific (i.e., manufacturing, telecommunications, computer networks, etc.) as opposed to generic or general." Our experience affirms this view in part, for certain tools must rely on two forms of expert knowledge: (1) the application specific and (2) the knowledge of simulation analysis, which is domain independent.

The major objective of the MDE project is to design a SMDE which is general or generic, applicable to many domains yet permitting customised applications. The complete set of requirements for developing such an environment poses a significant challenge to SMDE designers and implementers. Nevertheless, we are confident that the challenge can be met by way of an evolutionary development of SMDE prototypes.

Extending further on the research horizon is the development of a simulation management system[5,6] which embodies the SMDE for providing automated support throughout the entire life cycle of a simulation study.

CONCLUSIONS

Simulation software development is currently in a period of significant transition. Several recent research projects are investigating ways of providing an integrated and cost-effective simulation environment/system to support simulation throughout the model life cycle. Such environments/systems are likely to employ rapid prototyping, include more "intelligence," and exhibit much improved human interfaces. The MDE project experience suggests that an underlying methodology is essential to achieving an integrated suite of support tools. The concepts and principles employed, experiences gained, and guidelines

derived in the design and creation of the SMDE research prototype, reported herein, hopefully are helpful to both simulation researchers and practitioners struggling with increasingly more challenging modelling tasks.

ACKNOWLEDGEMENTS

This research was sponsored in part by the Naval Sea Systems Command and the Office of Naval Research under contract N60921-83-G-A165 through the Systems Research Center at VPI&SU.

REFERENCES

1. R.E. Nance (1983) A tutorial view of simulation model development. In *Proceedings of the 1983 Winter Simulation Conference*, Arlington, Va., December 1983, pp. 325-331.
2. O. Balci (1986) Requirements for model development environments. *Comput. & Ops. Res.* **13**, 1 (Jan.-Feb.), 53-67.
3. O. Balci (1986) Guidelines for successful simulation studies. Technical Report TR-85-2, Department of Computer Science, Virginia Tech, Blacksburg, Va., May
4. R.E. Nance (1981) Model representation in discrete event simulation: the conical methodology. Technical Report CS81003-R, Department of Computer Science, Virginia Tech, Blacksburg, Va., Mar.
5. R.E. Nance and O. Balci (1986) The objectives and requirements of model management. In *Systems and Control Encyclopedia: Theory, Technology, and Applications* (M. Singh, Ed.), Pergamon Press, Oxford. In press.
6. R.E. Nance, A.L. Mezaache and C.M. Overstreet (1981) Simulation model management: resolving the technological gaps. In *Proceedings of the 1981 Winter Simulation Conference*, Atlanta, Ga., December 1981, pp. 173-179.
7. D.W. Balmer and R.J. Paul (1986) CASM — the right environment for simulation. *J. Opl. Res. Soc.* **37**, 5 (May), 443-452.
8. D.W. Balmer (1985) An 'intelligent' environment for simulation. Paper presented at the *7th European Congress On O.R.* Bologna, Italy, June 1985.
9. G.I. Doukidis and R.J. Paul (1985) Research into expert systems to aid simulation model formulation. *J. Opl Res. Soc.* **36**, 4 (Apr.), 319-325.
10. R.J. Paul and G.I. Doukidis (1986) Further developments in the use of artificial intelligence techniques to formulate simulation problems. *J. Opl Res. Soc.* In press.
11. J.G. Crookes, D.W. Balmer, S.T. Chew and R.J. Paul (1986) A three-phase simulation

system written in Pascal. *J. Opl Res. Soc.* **37**, 6 (June), 603-618.

12. S.T. Chew, R.J. Paul and D.W. Balmer (1985) Three phase simulation modelling using an interactive simulation program generator. CASM Report, Department of Statistical and Mathematical Sciences, London School of Economics and Political Science.

13. Y.V. Reddy, M.S. Fox, N. Husain and M. McRoberts (1986) The knowledge-based simulation system. *IEEE Software* **3**, 2 (Mar.), 26-37.

14. Y.V. Reddy and M.S. Fox (1982) KBS: an artificial intelligence approach to flexible simulation. Technical Report CMU-RI-TR-82-1, Carnegie-Mellon University, Pittsburgh, Penn., Sept.

15. V. Baskaran and Y.V. Reddy (1984) An introspective environment for knowledge based simulation. In *Proceedings of the 1984 Winter Simulation Conference*, Dallas, Tex., November 1984, pp. 645-651.

16. D.J. McArthur, 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.), pp. 70-91. Addison-Wesley, Reading, Mass.

17. P. Klahr, D.J. McArthur, S. Narain and E. Best (1982) SWIRL: simulating warfare in the ROSS language. Technical Report N-1885-AF, The Rand Corporation, Santa Monica, Calif., Sept.

18. P. Klahr, J.W. Ellis, W.D. 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.), pp. 224-268. Addison-Wesley, Reading, Mass.

19. B. Unger, A. Dewar, J. Cleary and G. Birtwistle (1986) A distributed software prototyping and simulation environment: jade. In *Proceedings of the Conference on Intelligent Simulation Environments*, San Diego, Calif., January 1986, pp. 63-71. (Published as *Simulation Series* **17**(1) on Jan. 1986 by SCS, San Diego, Calif.)

20. J. Joyce, G. Birtwistle and B.L.M. Wyvill (1984) ANDES: an environment for animated discrete event simulation. United Kingdom Simulation Conference, Bath, September 1984.

21. C.R. Standridge, D.K. Vaughan and M.L. Sale (1985) A tutorial on TESS: the extended simulation system. In *Proceedings of the 1985 Winter Simulation Conference*, San Francisco, Calif., December 1985, pp. 73-79.

22. R. Reese and S. Sheppard (1983) A software development environment for simulation programming. In *Proceedings of the 1983 Winter Simulation Conference*, Arlington, Va., December 1983, pp. 419-426.

23. B.P. Zeigler and L. De Wael (1985) Towards a knowledge-based implementation of multifaceted modelling methodology. In *Proceedings of the European Conference on AI Applied to Simulation*, Ghent, Belgium, February 1985, pp. 42-51. (Published as *Simulation Series* **18**(1) on Feb. 1986 by SCS, San Diego, Calif.)

24. T.I. Ören and K.Z. Aytaç (1985) Architecture of MAGEST: a knowledge-based modelling and simulation system. In *Simulation in Research and Development* (A. Javor, Ed.), pp. 99-109. North-Holland, Amsterdam.

25. P. Robertson (1986) A rule based expert simulation environment. In *Proceedings of the*

Conference on Intelligent Simulation Environments, San Diego, Calif., January 1986, pp. 9-15. (Published as *Simulation Series* **17**(1) on Jan. 1986 by SCS, San Diego, Calif.)

26. F.J. Wales and P.A. Luker (1986) An environment for discrete event simulation. In *Proceedings of the Conference on Intelligent Simulation Environments*, San Diego, Calif., January 1986, pp. 58-62. (Published as *Simulation Series* **17**(1) on Jan. 1986 by SCS, San Diego, Calif.)

27. A. Lehmann, B. Knödler, E. Kwee and H. Szczerbicka (1985) Dialog-oriented and knowledge-based modelling in a typical PC environment. In *Proceedings of the European Conference on AI Applied to Simulation*, Ghent, Belgium, February 1985, pp. 91-96. (Published as *Simulation Series* **18**(1) on Feb. 1986 by SCS, San Diego, Calif.)

28. J.O. Henriksen (1983) The integrated simulation environment (Simulation software of the 1990s). *Opns. Res.* **31**, 6 (Nov.-Dec.), 1053-1073.

29. S. Ruiz-Mier, J. Talavage and D. Ben-Arieh (1985) Towards a knowledge-based network simulation environment. In *Proceedings of the 1985 Winter Simulation Conference*, San Francisco, Calif., December 1985, pp. 232-236.

30. R.E. Nance and C.M. Overstreet (1986) Diagnostic assistance using digraph representations of discrete event simulation model specifications. Technical Report TR-86-8, Department of Computer Science, Virginia Tech, Blacksburg, Va., Mar.

31. T. Taylor and T.A. Standish (1982) Initial thoughts on rapid prototyping techniques. *ACM SIGSOFT Software Engineering Notes* **7**, 5 (Dec.), 160-166.

32. F.P. Brooks (1975) *The Mythical Man-Month — Essays on Software Engineering.* Addison-Wesley, Reading, Mass.

33. R.E. Nance, O. Balci and R.L. Moose (1984) Evaluation of the UNIX host for a model development environment. In *Proceedings of the 1984 Winter Simulation Conference*, Dallas, Tex., December 1984, pp. 577-584.

34. R.L. Moose (1983) Proposal for a model development environment command language interpreter. Technical Report CS83032-R, Department of Computer Science, Virginia Tech, Blacksburg, Va., Dec.

35. M.C. Humphrey (1985) The command language interpreter for the model development environment: design and implementation. Technical Report TR-85-17, Department of Computer Science, Virginia Tech, Blacksburg, Va., Mar.

36. C.M. Overstreet (1982) Model specification and analysis for discrete event simulation. Ph.D. Dissertation, Virginia Tech, Blacksburg, Va., Dec.

37. C.M. Overstreet and R.E. Nance (1985) A specification language to assist in analysis of discrete event simulation models. *Commun. ACM* **28**, 2 (Feb.), 190-201.

38. R.H. Hansen (1984) The model generator: a crucial element of the model development environment. Technical Report CS84008-R, Department of Computer Science, Virginia Tech, Blacksburg, Va., Aug.

39. R.L. Moose and R.E. Nance (1985) Model analysis in a model development environment. Technical Report TR-85-27, Department of Computer Science, Virginia Tech, Blacksburg, Va.

40. J.C. Wallace and R.E. Nance (1985) The control and transformation metric: a basis for measuring model complexity. Technical Report TR-85-15, Department of Computer Science, Virginia Tech, Blacksburg, Va., Mar.

41. E.W. Dijkstra (1976) *A Discipline of Programming.* Prentice-Hall, Englewood Cliffs, N.J. (pp. 202-203)

42. R.C. Tausworthe (1977) *Standardized Development of Computer Software*, Prentice-Hall, Englewood Cliffs, N.J.

43. D.L. Parnas (1972) Information distribution aspects of design methodology. In *Proceedings of IFIP Congress 71*, North Holland Publishing Co., Amsterdam, TA-3, pp. 26-30.

44. C.M. Overstreet and R.E. Nance (1986) Discrete event simulation world views: characterization, transformation and analysis. In *Proceedings of the Workshop on Knowledge-Based Modelling and Simulation*, Wageningen, the Netherlands, North-Holland, Amsterdam. In press.

45. R.E. Shannon (1986) Intelligent simulation environments. In *Proceedings of the Conference on Intelligent Simulation Environments*, San Diego, Calif., January 1986, pp. 150-156. (Published as *Simulation Series* **17**(1) on Jan. 1986 by SCS, San Diego, Calif.)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER SRC-86-004 / TR-86-20 Dept. CS	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Simulation Model Development Environments: A Research Prototype		5. TYPE OF REPORT & PERIOD COVERED Interim Report: 16 March 86 - 15 August 86
		6. PERFORMING ORG. REPORT NUMBER SRC-86-004 / CS TR-86-20
7. AUTHOR(s) Osman Balci and Richard E. Nance		8. CONTRACT OR GRANT NUMBER(s) N60921-83-G-A165 B001-Z
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Computer Science and Systems Research Center Virginia Tech, Blacksburg, VA 24061		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Sea Systems Command SEA61E Washington, D.C. 20362		12. REPORT DATE 25 August 1986 13. NUMBER OF PAGES 24
14. MONITORING AGENCY NAME & ADDRESS(If different from Controlling Office) Naval Surface Weapons Center Dahlgren, VA 22448		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distributed to scientists and engineers at the Naval Surface Weapons Center and is made available on request to other Navy scientists. A limited number of copies is distributed for peer review.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

To Navy research and development centers and university-based laboratories.

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

computer systems, modelling, simulation

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

The needs for automated assistance in the simulation task are undeniable. The size and complexity of current modelling efforts far exceed the bounds considered challenging only a decade ago. A research prototype of a simulation model development environment (SMDE) has been developed to provide an integrated and comprehensive collection of computer-based tools to (1) offer cost-effective, integrated, and automated support of model development throughout its entire life cycle, (2) improve the model quality by effectively assisting in the quality assurance of the model, (3) signifi-

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

cantly increase the efficiency and productivity of the project team, and (4) substantially decrease the model development time. The prototype SMDE is composed of four layers: (0) hardware and operating system, (1) kernel SMDE, (2) minimal SMDE, and (3) SMDEs. Currently, a SUN 3/160 colour computer workstation and the UNIX operating system constitute layer 0 upon which an advanced prototype SMDE is being developed. The objectives of this paper are to: (1) describe the current state of research on simulation support environments, (2) use the SMDE research prototype as an example for describing concepts and principles employed, experiences gained, and guidelines (potential principles) derived in the design and creation of the prototype, and (3) identify and speculate on future research directions.