The Simulation Model Development Environment: An Overview

Osman Balci and Richard E. Nance

TR 92-32

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

June 3, 1992

Technical Report TR-92-32†

THE SIMULATION MODEL DEVELOPMENT ENVIRONMENT: AN OVERVIEW

by

Osman Balci and **Richard E. Nance**

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

1 June 1992

† Cross-referenced as Technical Report SRC-92-004, Systems Research Center, VPI&SU.

ABSTRACT

The purpose of this paper is to provide an overview of the Simulation Model Development Environment (SMDE) that has been under development since 1983. The SMDE architecture is composed of four layers: (0) Hardware and Operating System, (1) Kernel SMDE, (2) Minimal SMDE, and (3) SMDEs. Following the incremental development software engineering life cycle, SMDE software components are identified. Guided by the principles enunciated by the Conical Methodology, evolutionary prototyping and rapid prototyping approaches have been used to develop the following minimal SMDE tools: Premodels Manager, Assistance Manager, Model Generator, Model Analyzer, Model Translator, and Model Verifier. The Model Generator has been the most critically important tool, and five prototypes have been developed. The automation-based software paradigm has been achieved to a large extent with the development of the Visual Simulation Support Environment based on the DOMINO (multifaceteD cOnceptual fraMework for vIsual simulatioN mOdeling) and the VSMSL (Visual Simulation Model Specification Language).

CR Categories and Subject Descriptors: I.6.7 [**Simulation and Modeling**]: Simulation Support Systems—*Environments*; I.6.8 [**Simulation and Modeling**]: Types of Simulation—*Discrete event, Visual*, D.2.2 [**Software Engineering**]: Tools and Techniques—*Computer-aided software engineering*

Additional Key Words and Phrases: Automation-based software paradigm, simulation software engineering, software development environments, visual simulation.

TABLE OF CONTENTS

1. INTRODUCTION

The ever-increasing complexity of simulation model (software) development is undeniable. A simulation programming language supports only the programming process—one of ten processes in the life cycle of a simulation study [Balci 1990]. There is a need for automated support throughout the *entire* model development life cycle. This support can be provided in the form of an environment composed of integrated software tools providing computer-aided assistance in the development of a simulation model.

The authors have pursued research in building a discrete event Simulation Model Development Environment (SMDE) since 1983. The SMDE project has addressed a complex research problem: prototyping a domain-independent discrete-event SMDE to provide an *integrated* and comprehensive collection of computer-based tools to [Balci and Nance 1987a]:

① offer cost-effective, integrated and automated support of model development throughout the entire 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 [1986b], incremental development, evolutionary prototyping, and rapid prototyping approaches have been used to develop the prototypes of SMDE tools on a Sun computer workstation. The object-oriented paradigm, enunciated by the Conical Methodology [Nance 1981, 1987], has furnished the underpinnings of the SMDE research environment (The collection of tool prototypes).

Section 2 presents the SMDE architecture. The minimal SMDE tools are described in Section 3. Concluding remarks and future research are given in Section 4.

2. SMDE ARCHITECTURE

Figure 1 depicts the architecture of the SMDE in four layers: (0) Hardware and Operating System, (1) Kernel SMDE, (2) Minimal SMDE, and (3) SMDEs.

2.1 Layer 0: Hardware and Operating System

A Sun computer workstation with 8 megabytes of main memory, 380 megabytes of disk subsystem, a 1/4-inch cartridge tape drive, and a 19-inch color monitor with 1152x900 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.

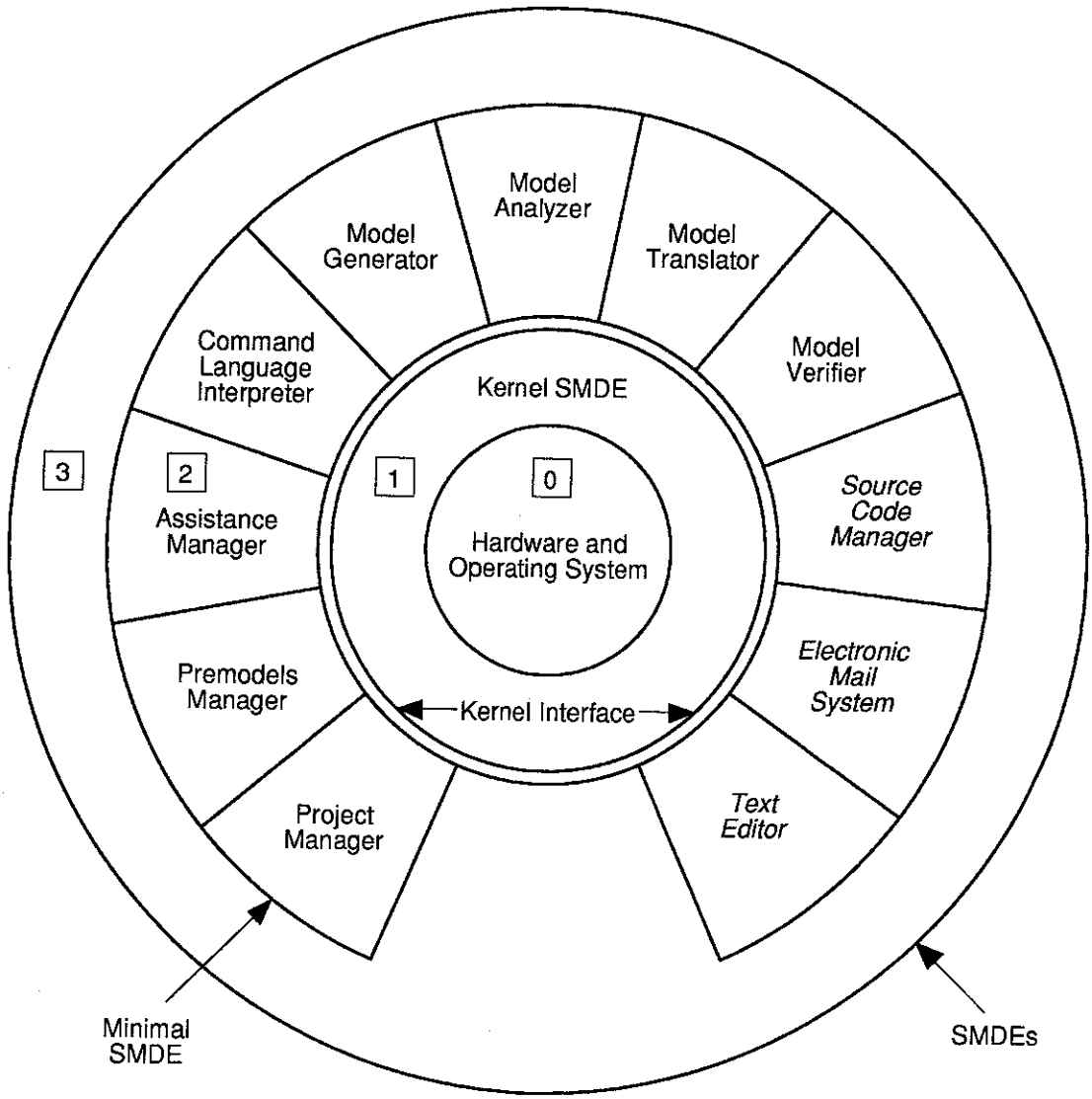

Figure 1. Simulation Model Development Environment Architecture

The UNIX SunOS 4.0 operating system and utilities, a graphical human-computer interface (SunView), device independent graphics library (SunCore), computer graphics interface (SunCGI), Sun programming environment (SunPro), and INGRES relational database management system constitute the software environment upon which the SMDE is built. Nance et al. [1984] evaluate the UNIX operating system as a foundation for building a simulation model development environment.

2.2 Layer 1: Kernel Simulation Model Development Environment

Primarily, this layer integrates all SMDE tools into the software environment described above. It provides INGRES databases, communication and run-time support functions, and a kernel interface. Three INGRES databases occupy 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 unauthorized use of tools or data.

2.3 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 credibility assessment stages. "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. Generality is claimed in the sense that the toolset is generically applicable to various simulation modeling tasks.

Minimal SMDE tools are classified into two categories. The first category contains tools specific to simulation modeling: Project Manager, Premodels Manager, Assistance Manager, Command Language Interpreter, Model Generator, Model Analyzer, 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. The prototypes of the minimal SMDE tools are described in Section 3.

2.4 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 are needed either within a particular project or by an individual modeler. If no other tools are added to a minimal 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 customizing 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 anticipated as available due to use in several other areas of application. A tool for statistical analysis of simulation output data, a tool for designing simulation experiments, a tool for documentation and credibility assessment [Kranowski 1988], a graphics tool, a tool for animation, and a tool for input data modeling are some example tools in layer 3. A Visual Simulator tool has been developed under the Visual Simulation Support Environment prototype based on a multifaceted conceptual framework [Derrick 1992].

An 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. A new tool can easily be added to the toolset by making the tool conform to the communication protocol of the kernel interface.

3. MINIMAL SMDE TOOLS

Each minimal SMDE tool is briefly described below with references containing detailed information about the development of the tool.

3.1 Project Manager

Project Manager software tool: (1) administers the storage and retrieval of items in the Project Database; (2) keeps a recorded history of the progress of the simulation modeling project; (3) triggers messages and reminders (especially about due dates); and (4) responds to queries in a prescribed form concerning project status. Other than the preliminary design decisions, no work has been done on the Project Manager. Its development is very dependent on how the other minimal SMDE tools are constructed; therefore, its development is being delayed until sufficient progress is achieved on the other tools.

3.2 Premodels Manager

The overall goal of the Premodels Manager (PM) software tool is to enable the user to: (a) locate and reuse components of successfully completed simulation studies, and (b) learn from past experience. The following design objectives are identified to meet this overall goal [Beams 1991; Beams and Balci 1992]:

① Provide easy methods of installing and maintaining documentation of successfully completed simulation studies [Nance 1977, 1979] in the Premodels Database.

② Provide appropriate methods of access to documentation of successfully completed simulation studies in the Premodels Database. Initiative, mechanisms, and complexity of access should vary according to task and type of user.

③ Provide a stratified display, capabilities for copying and pasting, capability for storing in a user-created file, and printing of the information located by a user in the Premodels Database.

④ Provide a user interface that satisfies the nine usability principles for interfaces: enable simple and natural dialogue, speak the user's language, minimize the user's memory load, promote consistency, provide feedback, provide clearly marked exits, provide shortcuts, supply good error messages, and prevent errors.

⑤ Provide context-sensitive help that is always available in a consistent manner. The system should use all available information on the user's state and avoid placing the burden on the user.

The PM consists of a collection of windows, as shown in Figure 2, which work together to allow different types of interactions between users and the Premodels Database. Three types of windows are used in the PM: (1) working windows (Browser, Searcher, Installer, and Maintainer), (2) access windows (Driver, Retriever, and Administrator), and (3) support windows (File Viewer, Describer, and Helper).

The PM has been evaluated with respect to the five design objectives stated above and has been found to provide effective reusability and learning support within the SMDE [Beams 1991; Beams and Balci 1992]. The design objectives altogether contribute to enabling the user to locate and reuse components of successfully completed simulation studies and learn from past experience.

The rapid prototyping software engineering approach has been used in developing the PM. The first PM prototype, developed by Box [1984], focused on the terminology problem in searching model components in the database. Subsequent PM prototypes have been developed, evaluated, and discarded prior to the current version described above. Knowledge gained by experimenting with one prototype PM has been used in developing the next improved PM prototype.

3.3 Assistance Manager

The overall goal of the Assistance Manager (AM) software tool is to provide effective and efficient transfer of assistance information to an SMDE user. "Effective" means accurate information is provided that is relevant to the user's needs. "Efficient" implies that if the user is involved in interaction with the SMDE, it is not necessary to switch tasks or modes in the process of seeking help. The following objectives are identified to meet this overall goal [Frankel 1987; Frankel and Balci 1989]:

(1) Provide general information for beginning system users. Such information would serve to acquaint new users with the environment, and establish a context for subsequent learning.

(2) Provide detailed and specific help on the use of an SMDE tool.

(3) Provide definitions and example usages of technical terms encountered in documentation and communication within the environment.

(4) Provide tutorial assistance for SMDE users. The tutorial should give the user a protected arena for limited experimentation with a tool's features.

(5) Provide help that is constantly available and immediately accessible. Methods should be available to suspend temporarily interaction with the AM, or save the current display for future reference. The user should not be required to step through an artificial protocol or syntax to access immediate assistance.

(6) Provide help that is unobtrusive; i.e., messages or prompts that are only visible when required or asked for.

(7) Provide a help system that is flexible enough to accommodate experienced users as well as novice or casual users.

Figure 2. Premodels Manager

(8) Provide context-sensitive help wherever possible. The system should use all available information on the user's state and avoid placing the burden on the user.

(9) Provide appropriate methods of access to the help information. Initiative, mechanisms, and complexity of access should vary according to task and type of user.

(10) Provide a straightforward and systematic method for tool developers (application programmers) to build help into tools which may be added to the environment.

(11) Provide help that is available in a consistent manner from any tool within the environment.

(12) Administer the Assistance Database by serving as an interface between the user or programmer and the database contents.

(13) Provide easy methods for update and expansion of the AM database. This is critical in order to accommodate the tailoring and updates that are inevitable in a large software environment. Updates should be enforced in a manner which helps enforce database integrity and consistency.

The AM has four components providing: (1) information on how to use an SMDE tool; (2) a glossary of technical terms; (3) introductory information about the SMDE; and (4) assistance for tool developers for integrating help information. Evaluation of the AM with respect to the 13 design objectives stated above has found the tool able to provide effective and efficient transfer of assistance information to an SMDE user [Frankel 1987; Frankel and Balci 1989].

3.4 Command Language Interpreter

The Command Language Interpreter (CLI) is the language through which a user invokes an SMDE tool. Early in the SMDE project, the CLI was prototyped based on the proposal of Moose [1983], and was fully described by Humphrey [1985]. Later, following the acquisition of the Sun computer workstation, the CLI was replaced by the SunView graphical user interface.

3.5 Model Generator

The Model Generator (the simulation model specification and documentation generator) is a software 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 [Nance 1977, 1979], and (3) accommodating model qualification. The Model Generator (MG) has been the most critically important SMDE tool. Five MG research prototypes have been developed.

3.5.1 Model Generator Prototype 1

The first MG prototype [Hansen 1984] implements the definition stage of the Conical Methodology [Nance 1981, 1987] and is implemented in C programming language on a VAX 11/780 computer system running a Unix emulator. The paradigm assumed for this prototype is a general tree for the purpose of simplicity. The hierarchical tree representation of a model facilitates the top-down definition and bottom-up specification approach of the Conical Methodology.

3.5.2 *Model Generator Prototype 2*

The second MG prototype focuses on the specification stage of the Conical Methodology and is implemented in C programming language on a VAX 11/780 running System V Unix [Barger 1986; Barger and Nance 1986]. The prototype: (1) provides a more structured approach to the model development process, especially the model specification phase; (2) utilizes, observes, and enforces the Conical Methodology principles, and (3) creates a relational database representation of a model specification from which a Condition Specification [Overstreet and Nance 1985, 1986; Overstreet et al. 1986] can be generated.

3.5.3 *Model Generator Prototype 3*

The third MG prototype builds on the capabilities of the first two and is developed on the Sun computer workstation described in Section 2.1 [Page 1990]. It supports both the definition and specification phases of the Conical Methodology. Under the conceptual framework of the Conical Methodology, the prototype enables a modeler to create a model specification in the Condition Specification language which lends itself for formal analysis.

3.5.4 *Model Generator Prototype 4*

The fourth is based on a conceptual framework developed by O. Balci and implemented by J. L. Bishop [Bishop 1989; Bishop and Balci 1990]. Under the title of General Purpose Visual Simulation System (GPVSS), this prototype includes the visualization/animation capability in the automatic generation of simulation models. It consists of over 11,000 lines of documented code developed on the Sun computer workstation described in Section 2.1. The GPVSS prototype assists a simulationist to: (1) graphically design a simulation model and its visualization, (2) interactively specify the model's logic, and (3) automatically generate the executable version of the model, while maintaining domain independence. This prototype provided the first experience in achieving the automation-based software paradigm [Balci and Nance 1987b].

3.5.5 *Model Generator Prototype 5*

The fifth MG prototype builds on the fourth one and has been a major prototype with full functionality. The prototype is a part of the Visual Simulation Support Environment (VSSE) containing the following software tools: Model Generator, Model Analyzer, Model Verifier, Model Translator, and Visual Simulator as shown in Figure 3 [Derrick 1992]. The VSSE has been developed on the Sun computer workstation described in Section 2.1 and consists of over 50,000 lines of documented code.

Figure 3. The Visual Simulation Support Environment

All VSSE tools have been developed under a new conceptual framework, called DOMINO (multifaceteD cOnceptual fraMework for vIsual simulatioN mOdeling). The DOMINO has been under development since 1984 and required tremendous amount of experimentation with many discarded undocumented prototypes. Only through the development of a fully functional environment (i.e., VSSE) and through a lot of experimentation, it was possible to test the feasibility and utility of the DOMINO conceptual framework.

The VSSE MG enables a modeler to provide a graphical, pictorial representation of a simulation model under the DOMINO conceptual framework as shown in Figure 4. After the model is graphically architectured, the modeler identifies the submodels and performs an object-oriented specification. Using the VSMSL (Visual Simulation Model Specification Language), which resembles the HyperTalk language, the modeler specifies the logic of a submodel. Once the specification is completed, the model can be translated automatically and fully into executable C code after it is successfully analyzed. By activating the Visual Simulator, a visual execution of the model can be obtained.

The prototyping research mentioned above, [Balci 1988; Balci and Nance 1989; Balci et al. 1990; Derrick 1988; Derrick et al. 1989; Nance 1984, 1988; Nance and Arthur 1988; Nance and Balci 1984, 1987; Nance et al. 1981], and Derrick's Ph.D. research [Derrick 1992] have brought significant contributions to achieving the automation-based software paradigm in the simulation modeling domain [Balci and Nance 1987b].

3.6 Model Analyzer

The Model Analyzer diagnoses the model specification created by the Model Generator and effectively assists the modeler in communicative model verification. Overstreet and Nance [1983] identify the following as the purposes of model analysis: (1) to assist in the identification of conceptual errors (misperceptions) or descriptive errors (misrepresentations) as early as possible in the model development process, (2) to suggest alternatives that might be less prone to errors or might offer more efficient model development and experimentation, and (3) to provide guidance and checks on the model development process. Four Model Analyzer prototypes have been developed.

The first prototype [Moose and Nance 1987a,b] enabled the recognition of three forms of diagnostic analysis: analytical, comparative, and informative diagnostic, all using the Condition Specification form of model representation. Attribute utilization, attribute initialization, and attribute classification testing have been fully implemented. This Model Analyzer prototype: (1) takes a model in the Condition Specification format as input, (2) parses the specification and displays syntax errors, (3) compiles model diagnostic information, (4) enables the modeler, through menu selections,

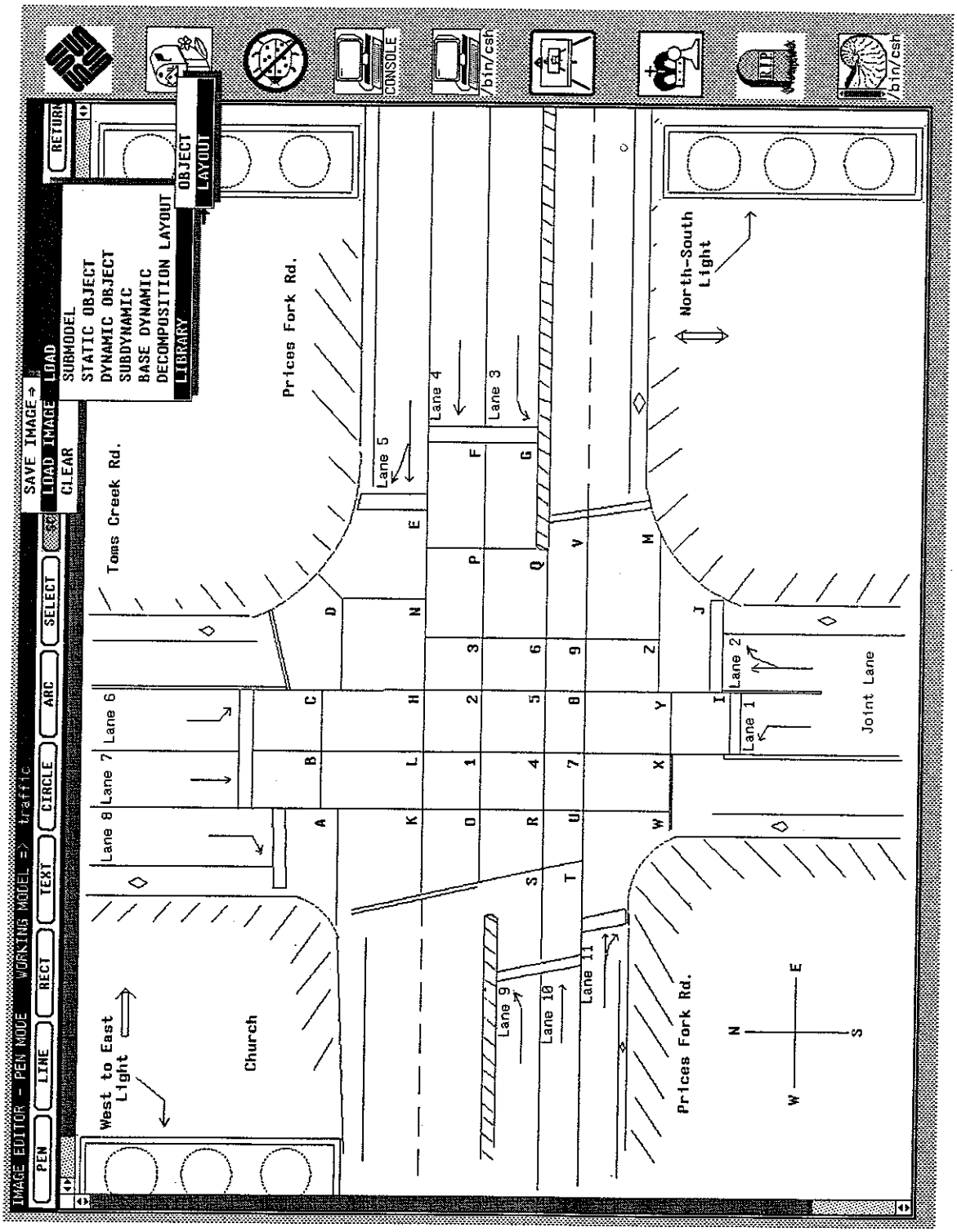

Figure 4. Graphical Model Construction under the Visual Simulation Support Environment

to view components of the model representation and pieces of the diagnostic information, and (5) produces and displays an action cluster incidence graph representation of the model. The prototype was built using the lex and yacc tools of the Unix operating system.

The second prototype implements a control and transformation metric for measuring model complexity [Wallace 1985, 1987; Wallace and Nance 1985], and provides diagnostic assistance using digraph representations of simulation model specifications [Nance and Overstreet 1987a,b; Overstreet and Nance 1983, 1986].

The third Model Analyzer prototype [Puthoff 1991] provides automated and semi-automated graph-based model diagnostic testing for model representations in the Condition Specification format. A file containing a Condition Specification of a model is submitted to the Model Analyzer. This Condition Specification is parsed using lex and yacc, and all analysis information is stored in an INGRES relational database. Based on the action cluster incidence graph and action cluster attribute graph, diagnostics are performed on the model specification. The Model Analyzer can output various aspects of the model, graphically display four different representations of the model, and perform the following diagnostic testing techniques: (1) analytical techniques (attribute utilization, attribute initialization, attribute reference, action cluster determinacy, statement order dependency, connectedness, accessibility, and out-complete), (2) comparative techniques (attribute cohesion, action cluster cohesion, and complexity), and (3) informative techniques (immediate precedence structure and decomposition) [Overstreet and Nance 1983; Nance and Overstreet 1987a,b].

The prototype also includes a rule-based expert system component, developed in Prolog on a VAX 8600, to produce a simplified action cluster incidence graph using knowledge generic to simulation and knowledge specific to the application domain [Overstreet and Nance 1986].

The fourth Model Analyzer prototype has been created under the Visual Simulation Support Environment [Derrick 1992]. This prototype accesses a relational representation of a simulation model in an INGRES database and performs completeness and consistency testing on the model specification created by the VSSE Model Generator under the DOMINO conceptual framework.

3.7 Model Translator

The Model Translator translates the model specification into an executable code after the quality of the specification is assured by the Model Analyzer. The first prototype Model Translator has been developed under the General Purpose Visual Simulation System [Bishop 1989; Bishop and Balci 1990]. The second is a tool in the Visual Simulation Support Environment [Derrick 1992]. Both prototypes extract the model specification from an INGRES relational database and fully convert it into a C programming language code. The use of C and INGRES are completely hidden

from the SMDE user. If translation errors occur, those errors are mapped back to the specification and the user is informed about the locations of the errors within the model specification. The simulator code is created by using object-oriented programming constructs under the Process Interaction conceptual framework [Balci 1988].

3.8 Model Verifier

The Model Verifier is intended for programmed model verification [Whitner and Balci 1989]. Applied to the executable representations, it provides assistance in substantiating that the simulation model is programmed from its specification with sufficient accuracy. Extensive groundwork and earlier research [Balci 1986a, 1987a, 1987b, 1989, 1990; Balci and Nance 1985; Whitner 1988] have contributed to the development of the Model Verifier. A prototype Model Verifier has been developed as part of the Visual Simulation Support Environment [Derrick 1992]. This prototype performs instrumentation-based dynamic testing on the executable representation of the simulation model.

3.9 Other Tools

Source Code manager, Electronic Mail System, and Text Editor are the other tools expected to be provided by the programming environment of the operating system used.

Source Code Manager is a software tool which configures the run-time system for execution of the programmed model, providing the requisite input and output devices, files and utilities. Its development is being delayed until a standard executable model representation is adopted for the SMDE.

Electronic Mail System facilitates the necessary communication among people involved in the project. Primarily, it performs the task of sending and receiving mail through (local or large) computer networks. The Sun workstation's MailTool is currently being used as the Electronic Mail System of the SMDE. The Sun computer workstation is a node on the Internet computer network with the node name of "mdesun.cs.vt.edu".

Text Editor is used for preparing technical reports, user manuals, system documentation, correspondence, and personal documents. Currently, the Sun text editor serves as the text editor of the SMDE.

4. CONCLUDING REMARKS AND FUTURE RESEARCH

The SMDE described in this paper can also be labeled "Computer-Aided Simulation Software Engineering Environment" or "Simulation Support Environment." We recognize that the complete set of requirements for building a SMDE poses a significant technical challenge under the objective of problem domain independence within the discrete event simulation area. Nevertheless, we have overcome the challenge by way of an evolutionary development of SMDE tool prototypes. In the creation of the Visual Simulation Support Environment prototype, we have not only achieved the automation-based software paradigm to a large extent, but also provided the capability of animating the simulation model.

One of the most challenging tasks has been the development of a conceptual framework for visual simulation modeling. The DOMINO—multifaceteD cOnceptual fraMework for vIsual simulatioN mOdeling—has been developed to provide the underpinnings of the Visual Simulation Support Environment.

The future research will deal with completing the prototyping of some SMDE tools and building a production version of the SMDE, on a hardware/software platform to be determined, based on the experience we have gained over the last nine years.

ACKNOWLEDGEMENTS

The SMDE research project has been sponsored in part by the U.S. Navy and IBM through the Systems Research Center at VPI&SU. The contributions of all those people listed as authors in the References section below are gratefully acknowledged.

REFERENCES

Balci, O. (1986a), "Credibility Assessment of Simulation Results," In *Proceedings of the 1986 Winter Simulation Conference*, J.R. Wilson, J.O. Henriksen, and S.D. Roberts, Eds. IEEE, Piscataway, NJ, pp. 38-43.

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

Balci, O., Ed. (1987a), *Proceedings of the Conference on Methodology and Validation* (1987 Eastern Simulation Conferences, Orlando, FL, April 6-9). Published as *Simulation Series 19*, 1 (Jan. 1988), SCS, San Diego, CA.

Balci, O. (1987b), "Credibility Assessment of Simulation Results: The State of the Art," In *Proceedings of the Conference on Methodology and Validation*, O. Balci, Ed. Published as *Simulation Series 19*, 1 (Jan. 1988), 19-25. SCS, San Diego, CA.

Balci, O. (1988), "The Implementation of Four Conceptual Frameworks for Simulation Modeling in High-Level Languages," In *Proceedings of the 1988 Winter Simulation Conference*, M.A. Abrams, P.L. Haigh, and J.C. Comfort, Eds. IEEE, Piscataway, NJ, pp. 287-295.

Balci, O. (1989), "How to Assess the Acceptability and Credibility of Simulation Results," In *Proceedings of the 1989 Winter Simulation Conference*, E.A. MacNair, K.J. Musselman, and P. Heidelberger, Eds. IEEE, Piscataway, NJ, pp. 62-71.

Balci, O. (1990), "Guidelines for Successful Simulation Studies," In *Proceedings of the 1990 Winter Simulation Conference*, O. Balci, R.P. Sadowski, and R.E. Nance, Eds. IEEE, Piscataway, NJ, pp. 25-32.

Balci, O. and R.E. Nance (1985), "Formulated Problem Verification as an Explicit Requirement of Model Credibility," *Simulation 45*, 2 (Aug.), 76-86.

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

Balci, O. and R.E. Nance (1987b), "Simulation Support: Prototyping the Automation-Based Paradigm," In *Proceedings of the 1987 Winter Simulation Conference*, A. Thesen, H. Grant, and W.D. Kelton, Eds. IEEE, Piscataway, NJ, pp. 495-502.

Balci, O. and R.E. Nance (1989), "Simulation Model Development: The Multidimensionality of the Computing Technology Pull," In *Impacts of Recent Computer Advances on Operations Research*, R. Sharda et al., Eds. Elsevier Science Publishing, New York, NY, pp. 385-395.

Balci, O., R.E. Nance, E.J. Derrick, E.H. Page, and J.L. Bishop (1990), "Model Generation Issues in a Simulation Support Environment" In *Proceedings of the 1990 Winter Simulation Conference*, O. Balci, R.P. Sadowski, and R.E. Nance, Eds. IEEE, Piscataway, NJ, pp. 257-263.

Barger, L.F. (1986), "The Model Generator: A Tool for Simulation Model Definition, Specification, and Documentation," M.S. Thesis, Department of Computer Science, VPI&SU, Blacksburg, VA, Aug.

Barger, L.F. and R.E. Nance (1986), "Simulation Model Development: System Specification Techniques," Technical Report SRC-86-005, Systems Research Center, VPI&SU, Blacksburg, VA, Aug.

Beams, J.D. (1991), "A Premodels Manager for the Simulation Model Development Environment" M.S. Thesis, Department of Computer Science, VPI&SU, Blacksburg, VA, Sept.

Beams, J.D. and O. Balci (1992), "Providing Reusability and Learning Support in the Simulation Model Development Environment," Technical Report TR-92-03, Department of Computer Science, VPI&SU, Blacksburg, VA, Mar.

Bishop, J.L. (1989), "General Purpose Visual Simulation System," M.S. Thesis, Department of Computer Science, VPI&SU, Blacksburg, VA, June.

Bishop, J.L. and O. Balci (1990), "General Purpose Visual Simulation System: A Functional Description," In *Proceedings of the 1990 Winter Simulation Conference*, O. Balci, R.P. Sadowski, and R.E. Nance, Eds. IEEE, Piscataway, NJ, pp. 504-512.

Box, C.W. (1984), "A Prototype of the Premodels Manager," MDE Project Memorandum, Department of Computer Science, VPI&SU, Blacksburg, VA.

Derrick, E.J. (1988), "Conceptual Frameworks for Discrete Event Simulation Modeling," M.S. Thesis, Department of Computer Science, VPI&SU, Blacksburg, VA, Aug.

Derrick, E.J. (1992), "A Visual Simulation Support Environment Based on a Multifaceted Conceptual Framework," Ph.D. Dissertation, Department of Computer Science, VPI&SU, Blacksburg, VA, Apr.

Derrick, E.J., O. Balci, and R.E. Nance (1989), "A Comparison of Selected Conceptual Frameworks for Simulation Modeling," In *Proceedings of the 1989 Winter Simulation Conference*, E.A. MacNair, K.J. Musselman, and P. Heidelberger, Eds. IEEE, Piscataway, NJ, pp. 711-718.

Frankel, V.L. (1987), "A Prototype Assistance Manager for the Simulation Model Development Environment," M.S. Thesis, Department of Computer Science, VPI&SU, Blacksburg, VA, July.

Frankel, V.L. and O. Balci (1989), "An On-Line Assistance System for the Simulation Model Development Environment," *International Journal of Man-Machine Studies 31*, 699-716.

Hansen, R.H. (1984), "The Model Generator: A Crucial Element of the Model Development Environment," Technical Report SRC-85-004, Systems Research Center, VPI&SU, Blacksburg, VA, Aug.

Humphrey, M.C. (1985), "The Command Language Interpreter for the Model Development Environment: Design and Implementation," Technical Report SRC-85-011, Systems Research Center, VPI&SU, Blacksburg, VA, Mar.

Kranowski, M. (1988), "CADCAS: A Tool for Computer-Aided Documentation and Credibility Assessment of a Simulation Study," M.I.S. Project Report, Department of Computer Science, VPI&SU, Blacksburg, VA, July.

Moose, R.L., Jr. (1983), "Proposal for a Model Development Environment Command Language Interpreter," Technical Report SRC-85-012, Systems Research Center, VPI&SU, Blacksburg, VA, Dec.

Moose, R.L., Jr. and R.E. Nance (1987a), "Model Analysis in a Model Development Environment," Technical Report SRC-87-010, Systems Research Center, VPI&SU, Blacksburg, VA, July.

Moose, R.L., Jr. and R.E. Nance (1987b), "The Design and Development of an Analyzer for Discrete Event Model Specifications," In *Impacts of Recent Computer Advances on Operations Research*, R. Sharda et al., Eds. Elsevier Science Publishing, New York, NY, pp. 407-421.

Nance, R.E. (1977), "The Feasibility of and Methodology for Developing Federal Documentation Standards for Simulation Models," Final Report to the National Bureau of Standards, Department of Computer Science, VPI&SU, Blacksburg, VA, June

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 CS81003-R, Department of Computer Science, VPI&SU, Blacksburg, VA, Mar.

Nance, R.E. (1984), "Model Development Revisited," In *Proceedings of the 1984 Winter Simulation Conference*, S. Sheppard, U.W. Pooch, and C.D. Pegden, Eds. IEEE, Piscataway, NJ, pp. 75-80.

Nance, R.E. (1987), "The Conical Methodology: A Framework for Simulation Model Development," In *Proceedings of the Conference on Methodology and Validation*, O. Balci, Ed. Published as *Simulation Series 19*, 1 (Jan. 1988), 38-43. SCS, San Diego, CA.

Nance, R.E. (1988), "Contemplations of a Simulated Navel or Recognizing the Seers among the Peers," Technical Report SRC-88-004, Systems Research Center, VPI&SU, Blacksburg, VA, Jan.

Nance, R.E. and J.D. Arthur (1988), "The Methodology Roles in the Realization of a Model Development Environment," In *Proceedings of the 1988 Winter Simulation Conference*, M.A. Abrams, P.L. Haigh, and J.C. Comfort, Eds. IEEE, Piscataway, NJ, pp. 220-225.

Nance, R.E. and O. Balci (1984), "A Model Development Environment for Combat Systems Experimentation," Final Report for FY 84 to the Naval Sea Systems Command and the Office of Naval Research, VPI&SU, Blacksburg, VA, Oct.

Nance, R.E. and O. Balci (1987), "Simulation Model Management Objectives and Requirements" In *Systems and Control Encyclopedia: Theory, Technology, Applications*, M.G. Singh, Ed. Pergamon Press, Oxford, pp. 4328-4333.

Nance, R.E., O. Balci, and R.L. Moose, Jr. (1984), "Evaluation of the UNIX Host for a Model Development Environment," In *Proceedings of the 1984 Winter Simulation Conference*, S. Sheppard, U.W. Pooch, and C.D. Pegden, Eds. IEEE, Piscataway, NJ, pp. 577-584.

Nance, R.E., A.L. Mezaache, and C.M. Overstreet (1981), "Simulation Model Management: Resolving the Technological Gaps," In *Proceedings of the 1981Winter Simulation Conference*, T.I. Ören, C.M. Delfosse, and C.M. Shub, Eds. IEEE, Piscataway, NJ, pp. 173-180.

Nance, R.E. and C.M. Overstreet (1987a), "Diagnostic Assistance Using Digraph Representations of Discrete Event Simulation Model Specifications," *Transactions of the Society for Computer Simulation 4*, 1 (Jan.), 33-57.

Nance, R.E. and C.M. Overstreet (1987b), "Exploring the Forms of Model Diagnosis in a Simulation Support Environment," In *Proceedings of the 1987 Winter Simulation Conference*, A. Thesen, H. Grant, W.D. Kelton, Eds. IEEE, Piscataway, NJ, pp. 590-596.

Overstreet, C.M. and R.E. Nance (1983), "Graph-Based Diagnosis of Discrete Event Model Specifications," Technical Report TR-83-28, Department of Computer Science, VPI&SU, Blacksburg, VA, June.

Overstreet, C.M. and R.E. Nance (1985), "A Specification Language to Assist in Analysis of Discrete Event Simulation Models," *Communications of the ACM 28*, 2 (Feb.), 190-201.

Overstreet, C.M. and R.E. Nance (1986), "World View Based Discrete Event Model Simplification," In *Modelling and Simulation Methodology in the Artificial Intelligence Era*, M.S. Elzas, T.I. Ören, and B.P. Zeigler, Eds. North-Holland, Amsterdam, pp. 165-179.

Overstreet, C.M., R.E. Nance, O. Balci, and L.F. Barger (1986), "Specification Languages: Understanding Their Role in Simulation Model Development," Technical Report SRC-87-001, Systems Research Center, VPI&SU, Blacksburg, VA, Dec.

Page, E.H., Jr. (1990), "Model Generators: Prototyping Simulation Model Definition, Specification, and Documentation under the Conical Methodology," M.S. Thesis, Department of Computer Science, VPI&SU, Blacksburg, VA, Aug.

Puthoff, F.A. (1991), "The Model Analyzer: Prototyping the Diagnosis of Discrete-Event Simulation Model Specifications," M.S. Thesis, Department of Computer Science, VPI&SU, Blacksburg, VA, Sept.

Wallace, J.C. (1985), "The Control and Transformation Metric: A Basis for Measuring Model Complexity," M.S. Thesis, Department of Computer Science, VPI&SU, Blacksburg, VA, Mar.

Wallace, J.C. (1987), "The Control and Transformation Metric: Toward the Measurement of Simulation Model Complexity," In *Proceedings of the 1987 Winter Simulation Conference*, A. Thesen, H. Grant, W.D. Kelton, Eds. IEEE, Piscataway, NJ, pp. 597-603.

Wallace, J.C. and R.E. Nance (1985), "The Control and Transformation Metric: A Basis for Measuring Model Complexity," Technical Report SRC-85-007, Systems Research Center, VPI&SU, Blacksburg, VA, Mar.

Whitner, R.B. (1988), "A Taxonomical Review of Software Verification Techniques: An Illustration Using Discrete-Event Simulation," M.S. Thesis, Department of Computer Science, VPI&SU, Blacksburg, VA, Oct.

Whitner, R.B. and O. Balci (1989), "Guidelines for Selecting and Using Simulation Model Verification Techniques," In *Proceedings of the 1989 Winter Simulation Conference*, E.A. MacNair, K.J. Musselman, and P. Heidelberger, Eds. IEEE, Piscataway, NJ, pp. 559-568.