Proceedings of the 1995 Winter Simulation Conference
ed. C. Alexopoulos, K. Kang, W. R. Lilegdon, and D. Goldsman

A PICTURE-BASED OBJECT-ORIENTED VISUAL SIMULATION ENVIRONMENT

Osman Balci

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

Anders I. Bertelrud
Charles M. Esterbrook

Orca Computer, Inc.
VT Corporate Research Center
1800 Kraft Drive, Suite 111
Blacksburg, Virginia 24060, U.S.A.

Richard E. Nance

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

ABSTRACT

This paper introduces the Visual Simulation Environment (VSE)—a new technology in visual simulation created as a result of experimental research for over a decade. Computer-aided support throughout the *entire* model development life cycle is required to manage the ever-increasing complexity of (visual) simulation modeling. Ongoing research since 1983 has sought to provide such support in the form of a computer-aided simulation model development environment. Based on the experience gained from two major prototype environments, the production version has been under development since August 1992. The current status of the VSE is presented herein.

1 INTRODUCTION

A simulation programming language supports only the programming process—one of ten processes in the life cycle of a simulation study (Balci 1994)—and does not by itself provide adequate support for construction of large and complex simulation models. Computer-aided support throughout the *entire* model development life cycle is required to manage the ever-increasing complexity of (visual) simulation modeling. This support should be provided in the form of an environment: a suite of integrated software tools.

The research in building a discrete event Simulation Model Development Environment (SMDE) started in June 1983 at Virginia Tech under funding from the U.S. Navy. The SMDE project has addressed a complex research problem: prototyping a domain-independent discrete-event SMDE. The SMDE provides an *integrated* and comprehensive collection of computer-based tools to: (1) offer cost-effective, integrated and computer-aided support of simulation model development throughout the *entire* model development 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. (Balci and Nance 1987)

Guided by the fundamental requirements identified by Balci (1986), incremental development, evolutionary prototyping, and rapid prototyping approaches have been used to develop the prototypes of SMDE tools. An overview of the SMDE architecture and prototype tools is given by Balci and Nance (1992). A Visual Simulation Support Environment (VSSE) research prototype was completed on a Sun computer workstation in April 1992 (Derrick and Balci 1992, 1995).

Based on the experience gained from the use of the SMDE and VSSE prototypes, development of the production version of the environment, named the Visual Simulation Environment (VSE), started in August 1992 under the object-oriented software engineering environment of the Unix-based NEXTSTEP operating system. A beta version of the VSE has been used by the students in the CS4214 Simulation and Modeling course in Spring 1995 and Fall 1995 at Virginia Tech. Beta testers at the Naval Surface Warfare Center (Dahlgren, VA) started using the VSE in July 1995. Technology transition is enabled by the creation of Orca Computer, Inc. awarded a contract by the Naval Research Laboratory to develop an extremely complex visual simulation model of world-wide air traffic control and satellite communication using the VSE.

The purpose of this paper is to introduce the Visual Simulation Environment—a new technology in visual simulation created as a result of experimental research for over a decade. Section 2 describes the VSE toolset and major features of the VSE. Example visual simulation models are presented in Section 3 to illustrate different areas of application of the VSE and the VSE characteristics. Conclusions are given in Section 4.

2 THE VISUAL SIMULATION ENVIRONMENT

The VSE currently runs under the Unix-based NEXTSTEP operating system on the following hardware platforms: Motorola 68040, Intel 80486/Pentium, Sun SPARC, and Hewlett-Packard PA-RISC. It will be ported to OpenStep which will run under Windows 95, Windows NT, Solaris, OSF/1, and other operating systems in 1996. A VSE model can be compiled as a four-way fat-binary to execute with no change on Motorola, Intel, SPARC, and PA-RISC hardware platforms running NEXTSTEP or OpenStep.

2.1 The VSE Toolset

The VSE currently contains the following toolset:

VSE Editor: is used for building a model in three phases. In Phase I, the editor provides computer-aided assistance to the modeler in graphical picture-based specification of a model in a hierarchical manner. The top-level model view is decomposed into components. Each component is further decomposed into other components as shown in Figure 1. Horizontal or vertical decomposition or a combination of the two can be used. Top-down or bottom-up approaches or a combination of the two can be employed in model construction. Each component can be represented by a scanned picture or a drawing in TIFF, EPS, GIF, JPEG, or many other formats. In Phase II, the model classes are created by inheriting from the built-in class hierarchy. Methods are named under each user-defined class. In Phase III, the logic of each method is specified by using an object-oriented scripting language which is very high level and English-like. Logic specification is localized to a particular method and thereby the complexity is significantly reduced.

VSE Library: contains earlier developed model components for reuse. Fully tested model components are stored in the library and can be reused by the copy-and-paste mechanism. Reusability is fully provided since each VSE model is truly object-oriented. For example, a particular type of satellite can be modeled by a dynamic object encapsulating all of its behavior and services, and can be made available for reuse in the library. Model components can be stored in the library for a particular problem domain such as manufacturing systems, networks, and air traffic control. The availability of model components for reuse facilitates the development of a new model. The more model components are reused from the library, the easier it gets to develop a new model.

VSE Model Analyzer: is used to apply static analysis on the model specification. All errors found are categorized by type and are listed in a scrollable window. Clicking on an error message in the list displays the

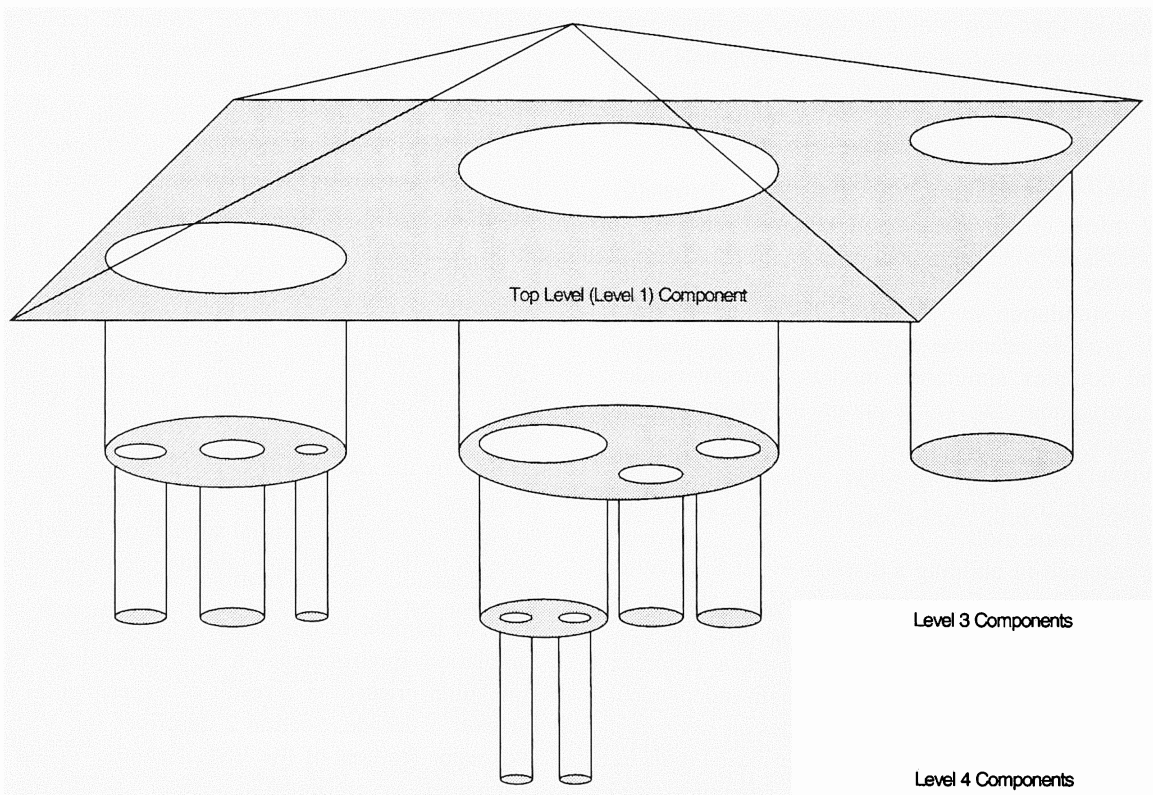

Figure 1: Decomposition of the Model Static Architecture or a Dynamic Object

location of the error as highlighted in the model specification. Completeness and consistency checking are applied among other static analysis techniques.

VSE Model Tester: is used to apply dynamic analysis on the executable version of the model. The use of extensive assertion checking is advocated. Through different modes, the Model Tester enables the modeler to apply a variety of dynamic testing techniques. For example, the life of a dynamic object can be monitored as it moves from one component to another.

VSE Simulator: is used to run the simulation model. Experiments can be conducted by using the method of replication, method of batch means, and other techniques. Model execution can be started, paused, resumed, and stopped. Visualization can be turned on and off during model execution. As many viewers as desired (as the screen space permits) can be displayed to concurrently view the visualization of different model components. Using the model browser, any model component can be selected for visualization during the course of execution.

VSE Data Analyzer: is used to perform statistical analysis of simulation output data such as confidence interval construction for an output variable or a graph creation in EPS format for output data.

VSE Learning Support System: is a multimedia system to assist the modelers in learning how to develop a VSE model and how to conduct a successful simulation study. Its hypertext capabilities enable cross linking between the elements of different documents. The LSS can play back prerecorded demonstrations on the computer with all screen motions and sound. It provides a glossary of simulation terms and a description of the life cycle of a simulation study. It contains the complete VSE Reference Manual with hypertext links.

VSE Simulation Study Evaluator: is used to assess the credibility of simulation study results.

2.2 The VSE Major Features

The VSE is a multifaceted visual simulation model development environment. Its major features are listed below in no particular order. The VSE:

- is intended for discrete-event type of visual simulation modeling for problem solving.
- is general purpose and domain independent. Several problem domains are illustrated by the sample models shown in Figures 2 through 7.
- can be used to visually simulate any complex system at any level of detail desired.
- provides the automation-based software paradigm where the central focus is on creating and maintaining the model specification and automatically generating the executable code. The VSE Editor is used for graphically building the model architecture, defining the classes by inheriting from the built-in class hierarchy, and specifying each class method logic by using an object-oriented scripting language which is very high level and English-like. Thereafter, the model is automatically translated into executable code using the built-in model translator in the VSE Editor.
- traces execution errors to the source point in the model specification. While the VSE Simulator is running the model, if an execution error occurs, it is mapped all the way back to the class method in which the error occurs, the VSE Editor is launched, and the location of the error is highlighted within the method.
- provides a picture-based approach to visual simulation modeling. Graphical representation of any model component can be an image composed of a scanned photograph/picture, a painting, or a drawing in TIFF, EPS, GIF, JPEG, or many other formats. Images in TIFF, EPS, GIF, or JPEG format can be clicked on, dragged, and dropped in the VSE Editor window. Images in other formats can be converted by using a variety of third party products and shareware software under NEXTSTEP, Macintosh, Windows, and other platforms.
- employs a multifaceted conceptual framework for guiding the modeler in model construction. The conceptual framework is based on the DOMINO conceptual framework (Derrick and Balci 1992; Derrick, Balci, and Nance 1989). Under the VSE conceptual framework, a model is viewed in terms of a static and a dynamic architecture. The static architecture is decomposed into a hierarchy of components as shown in Figure 1. A dynamic object is an object that physically or logically moves from one component to another. Similar to the static model architecture decomposition, a dynamic object can also be decomposed into a hierarchy of components as shown in Figure 1.
- enables the creation of truly object-oriented models exhibiting the properties of encapsulation, inheritance, and message passing.
- facilitates the reuse of earlier developed and fully tested model components stored in the VSE Library; thereby, significantly reducing the model development time.
- facilitates the development of the model static architecture in top-down and/or bottom-up hierarchical manner with vertical and/or horizontal decomposition.
- facilitates the development of a dynamic object in top-down and/or bottom-up hierarchical manner with vertical and/or horizontal decomposition. For

example, a ship can be represented as a dynamic object and can be decomposed into a hierarchy of components similar to the decomposition of the model static architecture as shown in Figure 1.

- facilitates the creation of "intelligent" moving objects by enabling logic specification for a dynamic object. A dynamic object (e.g., vehicle, satellite, airplane, passenger) can have its own logic and decide what action to take as opposed to a model component telling the dynamic object what actions to take.
- enables the creation of models using the machine-oriented view, material-oriented view or a combination of the two views (Derrick and Balci 1992).
- enables the modeler to follow the paradigm "What You See Is What You Represent." The VSE conceptual framework enables the modeler to represent the system elements as perceived by the modeler; thereby, significantly reducing the complexity of model specification. This capability in object-oriented terms is called the "Principle of Association."
- provides a state-of-the-art human-computer interface.
- provides a very high-level English-like object-oriented (with encapsulation, inheritance, and message passing) scripting language for method logic specification.
- reduces the amount of model logic specification and localizes it to a model segment.
- supports all phases of visual simulation model development and experimentation.
- facilitates the validation, verification, and testing of simulation models with the VSE Model Analyzer and VSE Model Tester software tools (Balci 1994).
- facilitates the credibility assessment of simulation study results with the VSE Simulation Study Evaluator software tool.
- provides a multimedia learning support system and on-line assistance with the VSE Learning Support System software tool.
- provides 27 random variate generators for the 27 probability distributions of UniFit II (Law and Vincent 1994).

3 EXAMPLES

Six sample VSE simulation models are presented in this section to illustrate the VSE capabilities for a variety of problem domains. All screen captures shown in Figures 2 through 7 are in 16-bit or 24-bit color.

Figure 2 shows the visual simulation of a Naval combat scenario. The background image was downloaded from an Internet site in GIF format and was dragged and dropped into the VSE Editor window to be the top-level model representation. The ship, aircraft, missile, and other images were selected from a Clip Art library available on the Macintosh and they were brought to NEXTSTEP and were used in model construction using the VSE Editor.

Figure 3 shows the top-level view of the visual simulation of the Washington DC metro system. The metro system map was scanned in 24-bit color on a scanner connected to a Macintosh and the TIFF image created was brought to NEXTSTEP. The image was dragged and dropped in the VSE Editor window. It was then resized in the Editor to be the top-level model view. Using the Editor's tool palette, the metro stations were defined as components (decompositions at level 2). Each decomposition (metro station) is graphically represented and decomposed further into the metro train tracks, train stop areas, passenger waiting areas, and entry and exit points. Train stop area is further decomposed. The metro train is represented as a dynamic object that is decomposed into an inside view that is further decomposed into seats and stand-up area.

The visual simulation can be viewed at model component or dynamic object level. The model hierarchy browser is shown in bottom-right corner of Figure 3. Clicking on a component name will show the visualization of that component. By clicking on the eye-shaped icon, another viewer can be displayed. As the screen space permits, many viewers can be used to concurrently watch the visual simulations of different components and dynamic objects. During model execution, one can double-click on a dynamic object to see the visual simulation of its decompositions if the dynamic object is further decomposed. In this model, the metro trains are decomposed but not the passenger dynamic objects.

Figure 4 shows three viewers concurrently displaying the visual simulations of the three model components. The globe component models the NAVSTAR Global Positioning System with 18 satellites. In addition, four geo-stationary satellites are included. The Asia and Americas components are decomposed into cities and ground control stations. Airplanes are present in the visual simulation and are decomposed further.

The model shown in Figure 5 was the Spring 1995 semester project in the CS4214 Simulation and Modeling course at Virginia Tech. An aerial photograph of the traffic intersection was obtained from the Town of Blacksburg. The photograph was scanned using a scanner connected to a Macintosh. The scanned image was brought into Adobe Photoshop on the Macintosh and was cleaned. Then, it was brought as a TIFF file into the VSE Editor to be the top-level model view. The CS4214 students collected data at the intersection and used UniFit II (Law and Vincent 1994) for input data modeling. The objective of the simulation study was to

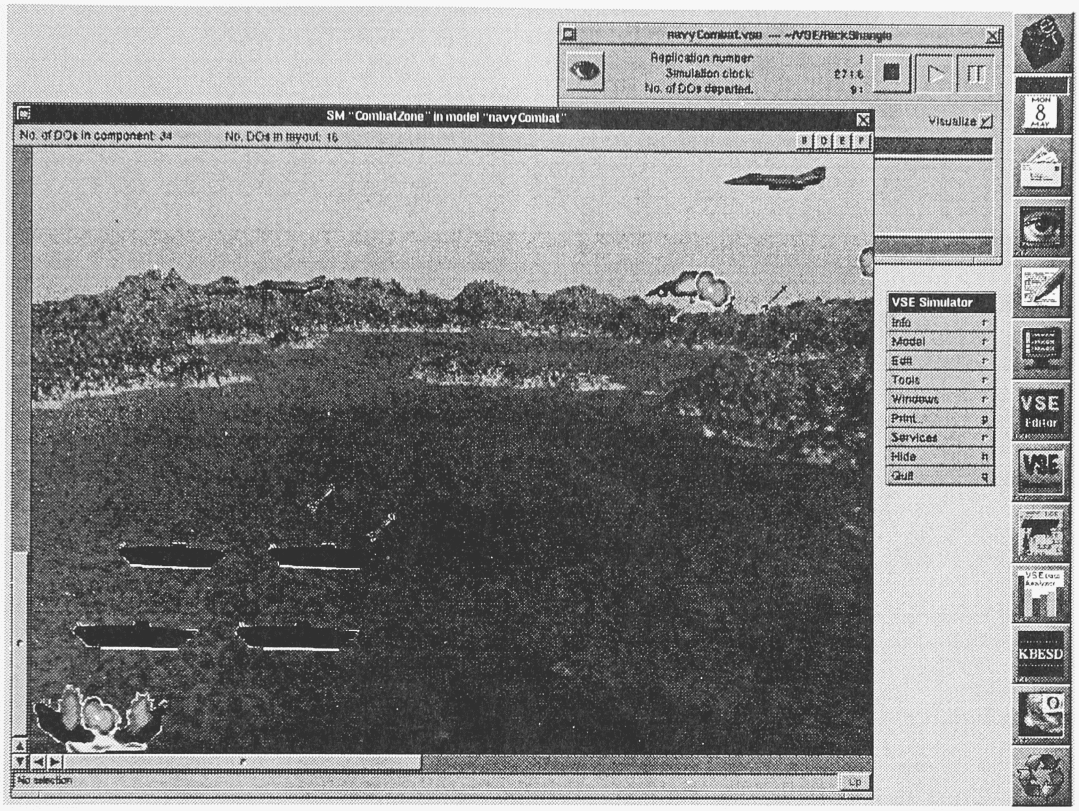

Figure 2: Visual Simulation of a Naval Combat Scenario Using the VSE

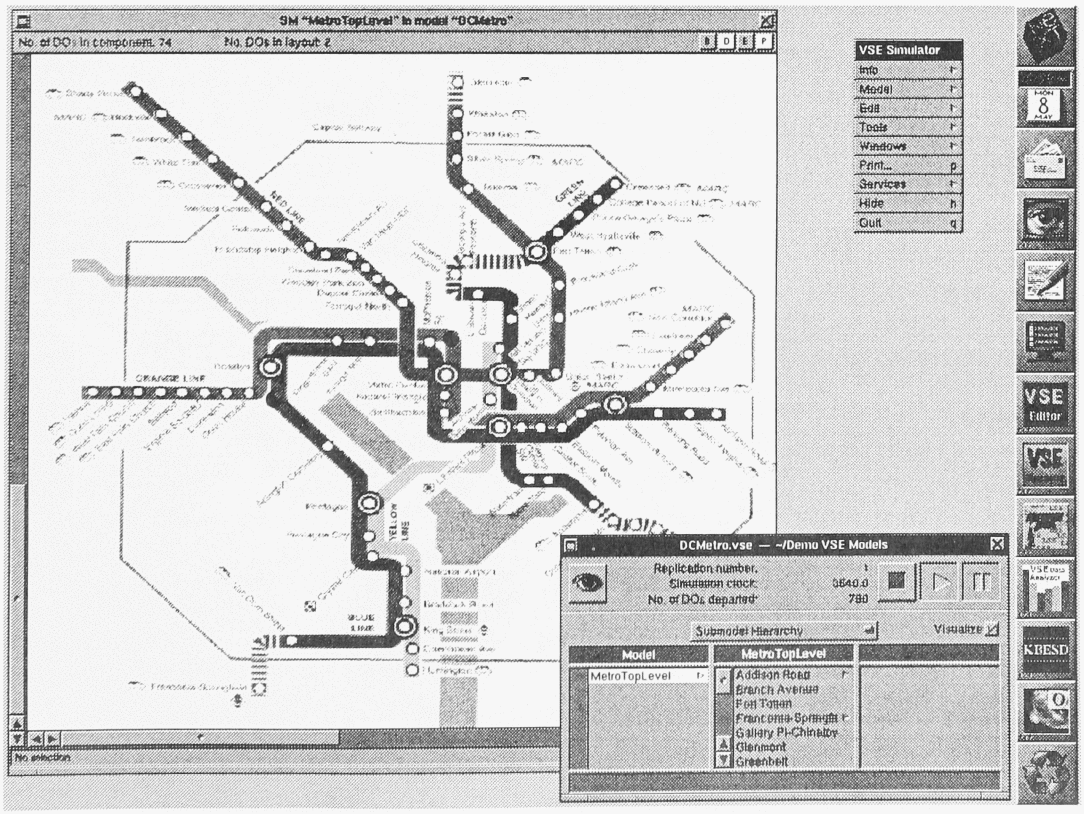

Figure 3: Visual Simulation of Washington DC Metro System Using the VSE

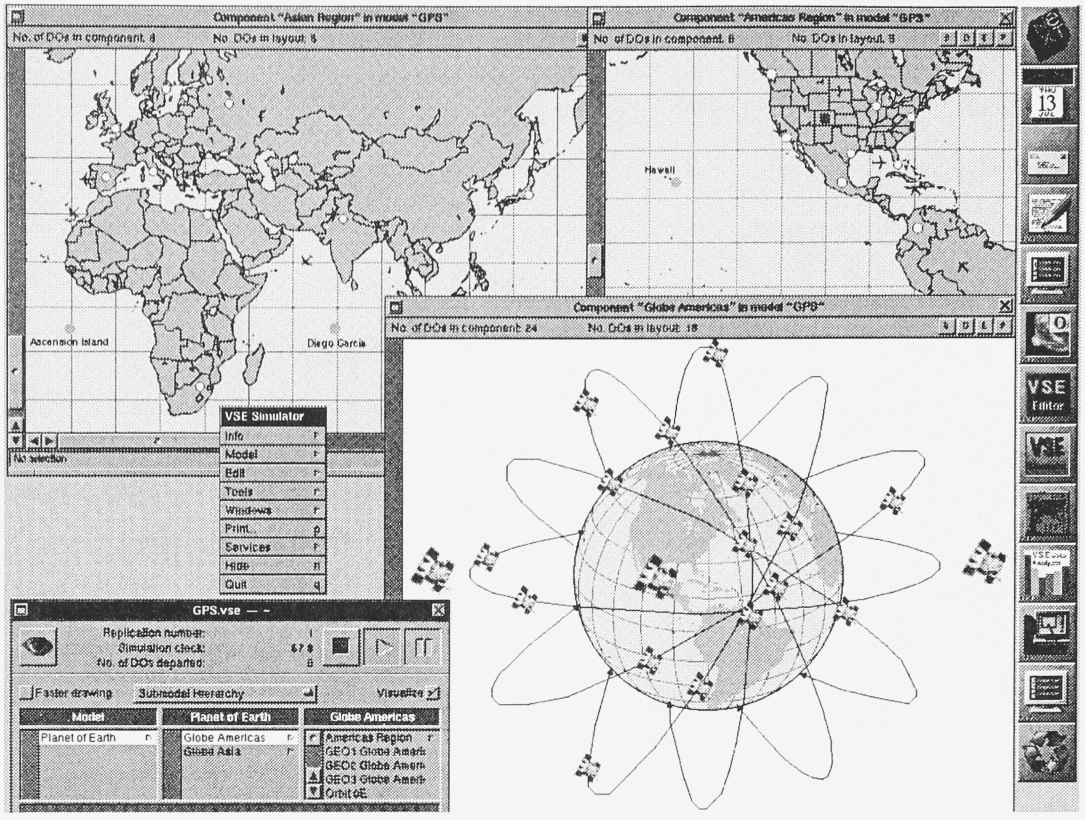

Figure 4: Visual Simulation of Satellite Communication / Global Air Traffic Control Using the VSE

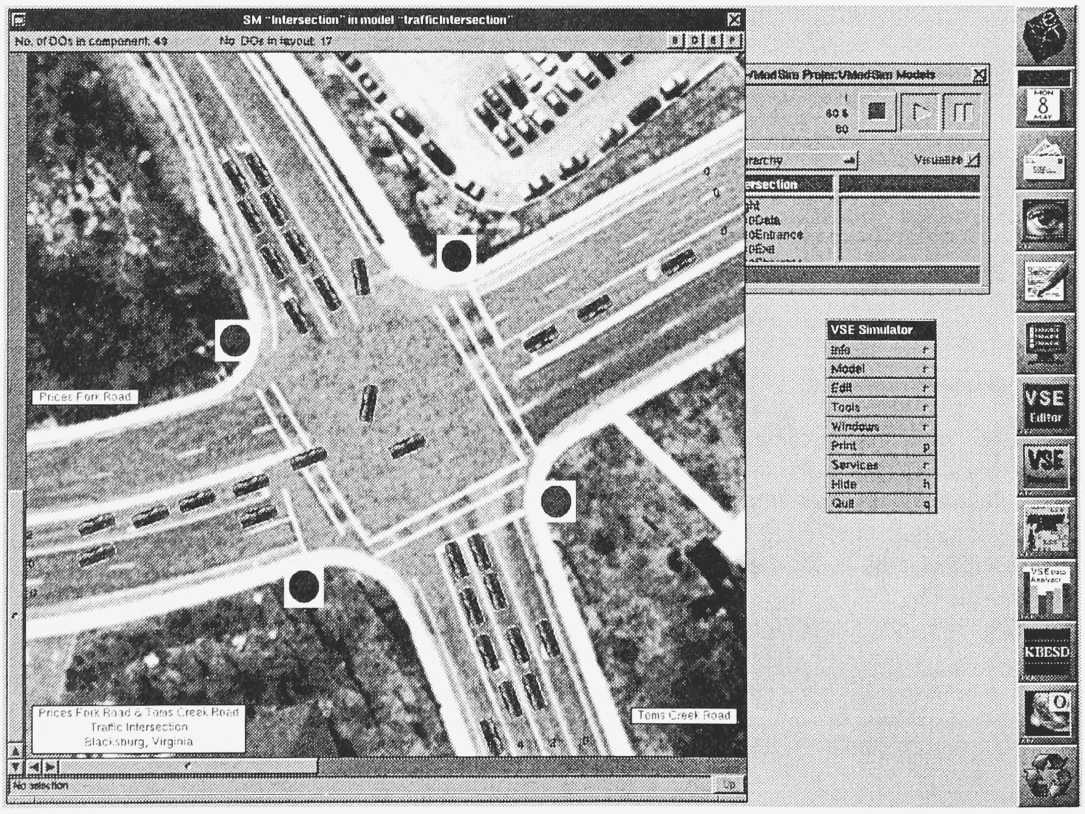

Figure 5: Visual Simulation of a Traffic Intersection in Blacksburg (Virginia) Using the VSE

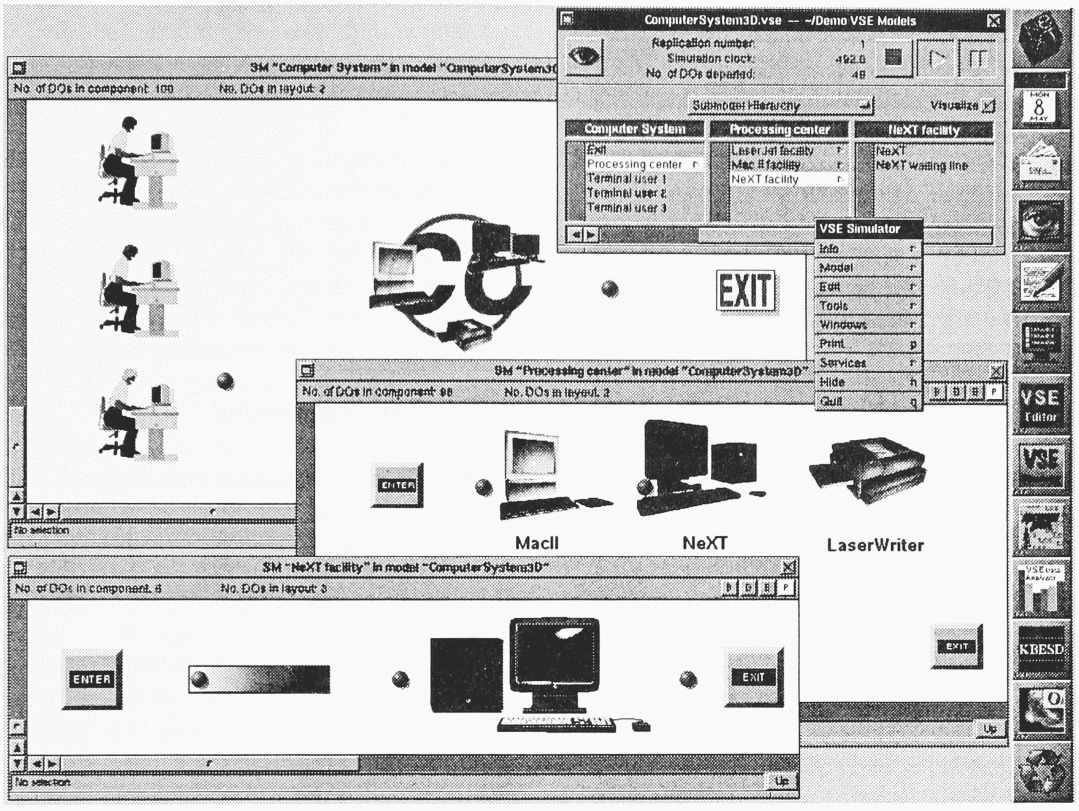

Figure 6: Visual Simulation of a Networked Computer System Using the VSE

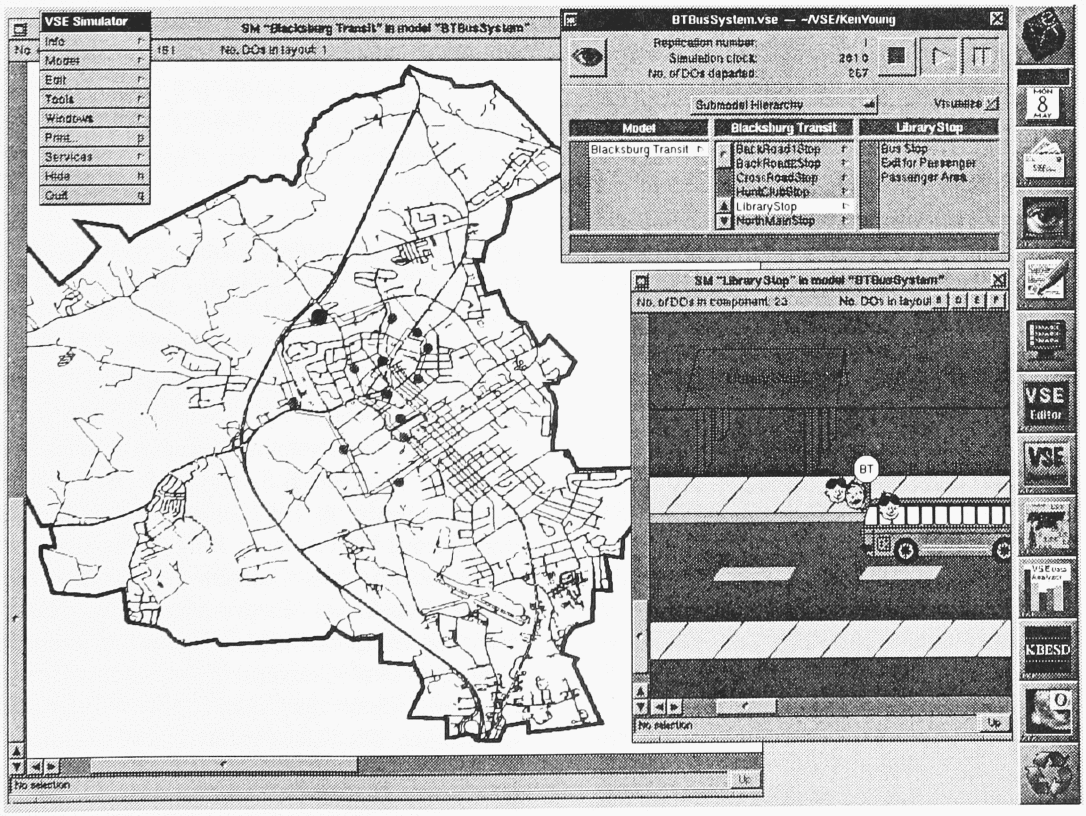

Figure 7: Visual Simulation of Blacksburg Transit Bus Transportation System Using the VSE

determine which of the three light timings (the currently used and two alternatives) is the best in terms of reducing the average waiting times of vehicles at the intersection. The VSE Data Analyzer was used to construct confidence intervals for the response variables.

Figure 6 shows the visual simulation of a computer system. Figure 7 shows the visual simulation of the Blacksburg bus transportation system. Double-clicking on the bus dynamic object displays visual simulation of the inside of the bus and the movements of passengers.

4 CONCLUSIONS

A general-purpose, picture-based, object-oriented visual simulation model development environment is presented. The VSE provides computer-aided assistance in the creation of complex visual simulation models and significantly reduces the model development time. It guides the modeler under a multifaceted conceptual framework and facilitates the pictorial and hierarchical model development. Logic specification is localized to a class method and is done by using an object-oriented, very high-level, English-like scripting language.

ACKNOWLEDGMENTS

The research that has led to the creation of the VSE since 1983 has been sponsored in part by the U.S. Navy through the Systems Research Center at Virginia Tech. Contributions of Emilio Arce, Rick Shangle, and Ken Young in developing some of the sample models are gratefully acknowledged.

REFERENCES

Balci, O. 1986. Requirements for model development environments. *Computers & Operations Research* 13:53-67.

Balci, O. 1994. Validation, verification, and testing techniques throughout the life cycle of a simulation study. *Annals of Operations Research* 53:121-173.

Balci, O. and R. E. Nance. 1987. Simulation model development environments: a research prototype. *Journal of Operational Research Society* 38:753-763.

Balci, O. and R. E. Nance. 1992. The simulation model development environment: an overview. In *Proceedings of the 1992 Winter Simulation Conference*, ed. J. J. Swain, D. Goldsman, R. C. Crain, and J. R. Wilson, 726-736. IEEE, Piscataway, New Jersey.

Derrick, E. J. and O. Balci. 1992. DOMINO: A multifaceted conceptual framework for visual simulation modeling. Technical Report TR-92-43, Department of Computer Science, Virginia Tech, Blacksburg, VA.

Derrick, E. J. and O. Balci. 1995. A visual simulation support environment based on the DOMINO conceptual framework. *The Journal of Systems and Software*, to appear.

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*, ed. E. A. MacNair, K. J. Musselman, and P. Heidelberger, 711-718. IEEE, Piscataway, New Jersey.

Law, A. M. and S. Vincent. 1994. *UniFit II user's guide*, Averill M. Law & Associates, Tucson, AZ.

AUTHOR BIOGRAPHIES

OSMAN BALCI is an Associate Professor of Computer Science at Virginia Tech. He is also President and CEO of Orca Computer, Inc., developer of the Visual Simulation Environment. He received B.S. and M.S. degrees from Boğaziçi University in Istanbul, Turkey in 1975 and 1977, and M.S. and Ph.D. degrees from Syracuse University in 1978 and 1981. Dr. Balci is the Editor-in-Chief of *Annals of Software Engineering* and serves on seven editorial boards. His current research interests center on software engineering and simulation. Dr. Balci is a member of Alpha Pi Mu, Sigma Xi, Upsilon Pi Epsilon, ACM, IEEE CS, and INFORMS.

ANDERS I. BERTELRUD is a Vice President of Orca Computer, Inc., developer of the Visual Simulation Environment. He received B.S. and M.S. degrees in Computer Science from Virginia Tech in 1993 and 1995. He is a member of Phi Beta Kappa, Upsilon Pi Epsilon, and ACM.

CHARLES M. ESTERBROOK is a Vice President of Orca Computer, Inc., developer of the Visual Simulation Environment.

RICHARD E. NANCE is the RADM John Adolphus Dahlgren Professor of Computer Science and the Director of the Systems Research Center at Virginia Tech. He received B.S. and M.S. degrees from N.C. State University in 1962 and 1966, and a Ph.D. degree from Purdue University in 1968. He is also Chairman of the Board of Orca Computer, Inc., developer of the Visual Simulation Environment. Dr. Nance is the founding Editor-in-Chief of the *ACM Transactions on Modeling and Computer Simulation*. He served as Program Chair for the 1990 Winter Simulation Conference. Dr. Nance received an Exceptional Service Award from the TIMS College on Simulation in 1987. He is a member of Alpha Pi Mu, Sigma Xi, Upsilon Pi Epsilon, ACM, IIE, and INFORMS.