Proceedings of the 1998 Winter Simulation Conference
ed. D. J. Medeiros, E. Watson, J. Carson, and M. S. Manivannan

VISUAL SIMULATION ENVIRONMENT

Osman Balci
Anders I. Bertelrud
Chuck M. Esterbrook
Richard E. Nance

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

ABSTRACT

This paper describes the Visual Simulation Environment (VSE) software product. VSE has been developed under $1.3 million research funding, primarily from the U.S. Navy, for over a decade. It enables discrete-event, general-purpose, object-oriented, picture-based, component-based, visual simulation model development and execution. This advanced environment can be used for solving complex problems in areas such as air traffic control and space systems, business process reengineering and workflows, complex system design evaluation, computer and communication networks, computer performance evaluation, education and training, health care systems, manufacturing systems, military/combat systems, satellite and wireless communications systems, service systems, supply chain management, and transportation systems.

1 INTRODUCTION

The research in defining and creating a Simulation Model Development Environment (SMDE) began in June 1983 at Virginia Tech (Balci and Nance 1987). Guided by the fundamental requirements identified by Balci (1986), incremental development, evolutionary prototyping, and rapid prototyping approaches were used to develop prototypes of the 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 in April 1992 (Derrick and Balci 1995, 1997).

Based on the experience gained from the use of the SMDE and VSSE prototypes, development of VSE started in August 1992 under the object-oriented software development environment of the Unix-based NEXTSTEP operating system. A fully functional research prototype of VSE was developed at Virginia Tech in July 1995 (Balci et al. 1995). Technology transfer, enabled by the establishment of Orca Computer, Inc. (Balci et al. 1997d), produced the first commercial version of VSE in November 1996 for the NEXTSTEP operating system. Subsequently, VSE was released for OPENSTEP Mach Unix, Windows NT 4.0, Windows 95, and Windows 98. VSE will be released for the new Macintosh operating system, Mac OS X, as soon as it is available.

For more information about the VSE technology, please see Orca Computer's website at http://www.Orca-Computer.com and the VSE articles at http://www.Orca-Computer.com/VSE/Articles/VSEArticles.html.

The purpose of this paper is to describe the VSE software product developed as a result of in-depth and extensive experimental research. Section 2 presents VSE's technical characteristics. Section 3 provides some examples. Concluding remarks are given in Section 4.

2 VSE CHARACTERISTICS

2.1 Toolset

VSE consists of four software tools as described below (Balci et al. 1997c):

- **VSE Editor:** Lets you build your simulation model graphically using the object-oriented paradigm, with inheritance, message passing, and encapsulation.
- **VSE Simulator:** Provides animation and lets you run experiments with your simulation model. Also sold separately so that you can sell/distribute your model for others to use.
- **VSE Output Analyzer:** Lets you statistically analyze your simulation output data.

- **VSE Teacher:** Uses your default web browser and lets you learn VSE by watching video clips and by browsing through hypermedia technical, tutorial, and help information.

2.2 General Purpose and Domain Independent

VSE can be used for solving any problem as long as the problem is solvable using discrete-event simulation. Using VSE, you can eliminate purchasing and mastering another expensive simulation product every time your problem domain changes. You can use VSE for solving complex problems in areas such as: air traffic control and space systems, business process reengineering and workflows, complex system design evaluation, computer and communication networks, computer performance evaluation, education and training, health care systems, manufacturing systems, military/combat systems, satellite and wireless communications systems, service systems, supply chain management, and transportation systems.

2.3 Fully and Truly Object-Oriented

VSE provides all major characteristics of the object-oriented paradigm: objects, classes, instantiation, class and instance variables, class and instance methods, inheritance, message passing, encapsulation, polymorphism, dynamic binding, and association. The VSE Editor provides a built-in class library and an inheritance hierarchy, which you can extend by creating your own subclasses.

2.4 Libraries of Reusable Model Components

You can create your own library of reusable model components or use a library created by someone else. Once you make your model depend on the library, the library objects become available for reuse by drag-and-drop.

Libraries can be developed for particular application domains such as manufacturing, health care, transportation, business process reengineering, and networks. A model can be constructed by reuse with no programming. For more information, see (Balci et al. 1997b, 1998).

2.5 Model Maintainability

You can build highly maintainable simulation models using VSE. You can create a class and instantiate objects belonging to that class. Each instantiated object inherits all the characteristics and behavior defined in its class. You can instantiate thousands of objects but specify their behavior only once in their class. Later, if you want to change their behavior, you make the changes only in their class and all instantiated objects automatically inherit the new behavior. Thus, model maintainability is greatly facilitated.

VSE's drag-and-drop operation from a library is equivalent to instantiation of an object and enables ease of change unlike the copy-paste-and-change operation provided by other products. Copy-paste-and-change is *not* considered true reuse in software engineering. True reuse requires an inheritance hierarchy and instantiation capability under the object-oriented paradigm.

For example, suppose that you instantiated hundreds of cities (objects) from the City class in your model at design time and later, you wanted each city object to provide a new behavior, e.g., return its population. To implement this change, you simply add a new method to the City class to provide the new behavior. Thereafter, all city objects automatically inherit the new behavior.

If a change is needed for an object being reused from a library, that change is performed using inheritance. You can create a new class by subclassing from the class hierarchy of the library your model depends on and inherit all the characteristics and behavior of the superclass (parent class). Then, you can override a method of the superclass or create a new method, defining new characteristics and behavior under the new class.

2.6 What You See is What You Represent

Twisting of logic and unnatural modeling just to conform to the conceptual framework (CF) of a simulation software product significantly increases the model complexity, inhibits model validation, and degrades model maintainability, reusability, understandability, and reliability. The CF should allow for direct and natural model representation. VSE's CF provides the modeling paradigm *What You See is What You Represent* and lets you avoid twisting of logic in model representation (Balci et al. 1997a).

2.7 Multifaceted Conceptual Framework

VSE's CF possesses many facets including:

- **Graphical:** a model is structured graphically.
- **Hierarchical:** model static architecture (components) and model dynamic architecture (dynamic objects) are decomposed hierarchically.
- **Object-oriented:** fully and truly object-oriented.
- **Component-based:** a model can be developed by reusing model components from a library.
- **Picture-based:** you can photograph the system, scan the photograph, then drag, drop and resize the image in the VSE Editor as your component layout on top of which animation can take place.

- **Visual:** model execution provides automatic animation.
- **Multi-view:** you can use the machine-oriented view, material-oriented view or a combination of the two views in building complex visual simulation models.

2.8 Simplified Model Logic Specification

A VSE model is built in three steps. In Step 1, the model static and dynamic structures are created in a graphical and hierarchical manner. In Step 2, the new classes are named. In Step 3, the logic of each method of each class is specified using VSE's *high-level English-like object-oriented scripting language*. Only in Step 3 and only in a method is model logic specified using the scripting language. The localization of logic specification to a particular method significantly reduces the model complexity, simplifies model development, and increases model maintainability.

2.9 Hierarchical Model Architecture

Graphical and hierarchical decomposition of a simulation model in a top-down manner is a very powerful approach advocated by many authors in the published literature. VSE lets you decompose your model static architecture as well as your dynamic objects graphically and hierarchically in a top-down manner.

The VSE model architecture consists of static and dynamic parts. The *model static architecture* is composed of hierarchically decomposed *components*. The *model dynamic architecture* is made up of dynamic objects. A *dynamic object* is an entity of interest which physically or logically moves from one point to another in a model. For example, bus, ship, aircraft, passenger, train, and computer job might be represented as dynamic objects. A dynamic object may be decomposed into a hierarchy of components similar to the model static architecture decomposition.

2.10 Dynamic Object Decomposition

A distinctive capability of VSE is the flexibility given to modeling with dynamic objects. For example, a dynamic object representing an aircraft can be decomposed into its inside view component, which can be further decomposed into components representing cockpit, service area, passenger area, and cargo area. The passenger area component can be decomposed into seats. Pilots, attendants, and passengers can be represented as dynamic objects. A passenger dynamic object can enter into an aircraft dynamic object, move within its component hierarchy, and occupy a seat component. All dynamic objects which enter into an aircraft dynamic object then move together with the aircraft to an airport component of the model. In essence, their collective behavior is represented by the enclosing dynamic object (aircraft) but each passenger object retains its individual representation. (Balci et al. 1997a)

2.11 Graphical and Picture-Based

A VSE model is graphically structured in a hierarchical manner. You can drag and drop a graphical image in the VSE Editor in many different formats including EPS and TIFF. The image can represent your model component layout, or a portion of it, on top of which objects can be animated. You can resize and reposition the image after dropping it in the VSE Editor.

Graphical representation of a VSE model component can be constructed by using any of the following techniques (Balci and Nance 1998):

Diagram: A diagrammatic representation of a logical process can be used as the layout of a model component on top of which the logic execution can be animated. The logical process represented could be the execution of a computer program, information flow in an organization, or the activities in a business process. Many types of diagrams can be used (Balci and Nance 1998). Diagrammatic visualization enables the modeler to represent complex activities and procedures easily, and to communicate the associated logic to others in a concise, precise, and understandable manner.

Drawing: A model component layout can be composed of drawings created by using professional drawing software such as Adobe Illustrator, CorelDraw, and FreeHand. Such a layout relies on the knowledge and talent of the creator in producing a meaningful visual.

Icon: Icon-based layout composition is commonly used by problem domain-specific (e.g., manufacturing, health care, business processes) simulation software products. Different objects and tasks in a particular problem domain are represented by icons. The modeler builds a model component layout by selecting icons and connecting them to show potential interactions. The same functionality can easily be provided in VSE. Such iconic objects can be created in the VSE Editor's Templates window as reusable model components and you can reuse them by dragging and dropping from the Palette window.

Map: A geographic, military, or other kind of map can be used as a model component layout on top of which animation can be conducted. Maps enable us to represent an area in correct forms, sizes, and relationships. (See the gulf war visual simulation in Figure 1.)

Painting: A model component layout can be composed of paintings created by using professional painting software such as Adobe Photoshop, Paint Shop Pro, and Painter.

Photograph: A photograph can be much better than a thousand words because it is concise, precise, and clear in

representing a complex system. A complex system component can be photographed, the photograph can be scanned, the scanned image can be cleaned, and the photographic image can be used as the model component layout on top of which animation can be conducted. You can also use a digital camera to eliminate scanning. (See the traffic visual simulation in Figure 2.)

The photographic image can also be a frame of a movie or a video clip. You can video tape the system operation and use the video frames in creating the graphical layout of a VSE model component.

Schema: A model component layout can be composed of blueprints created by using professional computer-aided design (CAD) software such as AutoCAD and DesignCAD 97. The blueprint of a system design can be used as the model component layout on top of which animation can be conducted.

2.12 Input Data Specification

You can create the model input data elements (i.e., input variables, input parameters, and input data files) in a separate interface to your model (Input Data window). The input data elements can be referred to in the VSE Editor's Attributes Inspector panel for an object (graphically) and in the script of a method (textually).

In the VSE Simulator, you can enter values for the model input data elements under a convenient graphical user interface and create as many value sets as desired. You can then conduct an experiment under a selected value set. Thus, you can distribute or sell your model to others where the user will only need the VSE Simulator to conduct experiments with your model under different input data element values. The VSE Simulator is a stand-alone software tool providing VSE's runtime environment and is also sold separately for this purpose.

VSE provides exceptional flexibility in specifying input data values. You can create an input variable belonging to the VSInputStream class and enable the user of your model to specify a value for that variable from a variety of different sources. Using the VSE Simulator, the user can specify that the values of the variable come from a (a) random variate stream with a probability distribution, (b) trace data file, or (c) text file. Such specification does not require any model logic change.

2.13 Multiple Animation Viewers

You can create as many viewers as you want to watch the animations in the components of your model. You are limited only by the screen space. However, you can attach multiple monitors to your PC providing continuous screen space from one monitor to another. With more screen space, you can simultaneously view the animations of many components in your model. If you are limited by screen space, you can use a magnification level for each viewer between 25% and 400%.

2.14 Automation-Based Software Paradigm

You can use the VSE's state-of-the-art automation-based software engineering paradigm in building complex visual simulation models. You can focus on creating and maintaining the model specification graphically, hierarchically, and under the object-oriented paradigm and let the VSE automatically generate the executable model.

You can easily understand runtime errors within VSE. They are mapped to the model specification and not an intermediate programming language. When a runtime error occurs, double-clicking on a method name displayed by the VSE Simulator in the method trace window launches the VSE Editor and shows the method, and highlights the statement causing the error.

2.15 Zoom In and Zoom Out

Both VSE Editor and VSE Simulator let you increase or decrease the magnification of a component layout. The magnification level can be set to any value between 25% and 400%.

2.16 File Input and File Output

You can read text files into your model or write to them from your model.

2.17 External Functions

You can call functions in Dynamic Link Libraries (DLLs). Such external functions can be written in any programming language (e.g., C, Pascal), as long as they conform to the standard Microsoft Windows function calling convention.

2.18 Design of Experiments

Using the powerful message passing mechanism of the object-oriented paradigm provided by VSE, you can incorporate any statistical experimental design by instrumenting your simulation model.

VSE provides built-in random variate generators for 27 probability distributions. Each generator can be initialized with a seed for creating identical experimental conditions. You can perform self-driven simulation, trace-driven simulation, or a combination of the two.

2.19 Statistical Output Analysis

You can select *method of replications*, *method of batch means* or any other technique by instrumenting your model for collecting data. The collected data can be written to output data files. Using VSE Output Analyzer, you can open the output data files and statistically analyze the simulation output data.

3 EXAMPLES

3.1 Gulf War

Animation is conducted on top of a military map. The aircraft carrier dynamic object is decomposed into a layout showing its flight deck. Combat aircraft dynamic objects enter into the carrier dynamic object and move on its flight deck for landing and takeoff. Tank dynamic objects are created to be "intelligent", i.e., each has its own logic and acts on its own. Tanks fire at each other and are destroyed upon receiving too much damage. Aircraft fire missiles at certain targets and return to the carrier for refueling before taking off for the next attack. The dynamic object images are not created to the scale of the map; however, they could be created if so desired by zooming in the map as was done in the next example.

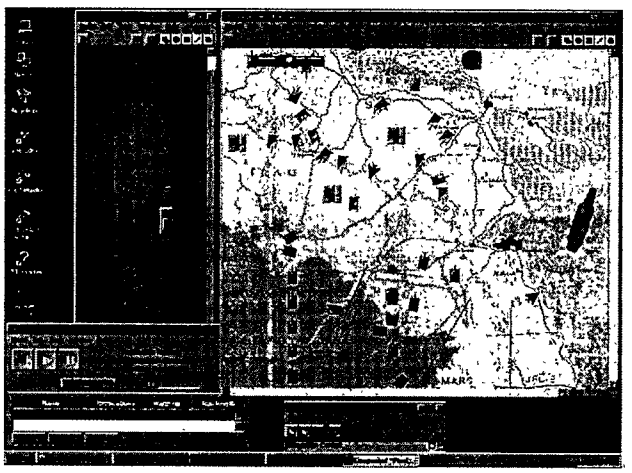

Figure 1: Gulf War Visual Simulation

3.2 Blacksburg Downtown Traffic

An aerial photograph of some of the Blacksburg downtown traffic intersections, obtained from the Town of Blacksburg, is scanned. The scanned image is cleaned and brought into the VSE Editor by dragging and dropping. Then, the photographic image is decomposed into components for representing vehicle movement. Vehicles are modeled as dynamic objects and are instantiated at runtime. The traffic light for each lane is depicted by a line that changes color during animation. Some intersections operate with stop signs. All of the model was built by reusing, with no programming, from the traffic library of reusable model components developed earlier. With the availability of the traffic library, such a model can be constructed by drag and drop with no programming.

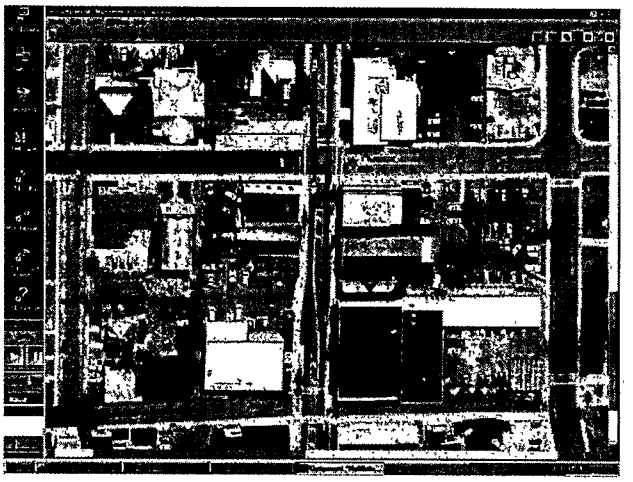

Figure 2: Traffic Visual Simulation

3.3 GPS Satellite Communications

The global positioning system (GPS), air traffic, satellite communications, air traffic control centers (ATCs), and GPS stations are all represented in this model. The world map, representing the model top level component, is decomposed into two components as shown in Figure 3. The white circles on the map represent the cities among which aircraft fly. Each city contains one ATC. When an aircraft enters into a satellite's air space, it broadcasts its tail number to the satellite which in turn relays the information to the closest ATC or GPS station.

3.4 Global Air Traffic

An input data file is created from the Federal Aviation Association Official Airline Guide March 1996 data file to contain longitudes and latitudes of 200 cities and 1000 flight schedules among the 200 cities worldwide. At the start-up of simulation, the input data file is read in and the 200 cities are created and placed on the map as deep components, each containing a graphical city layout. Based on the flight information, dynamic objects representing the aircraft are instantiated at runtime. Double-clicking on an aircraft image during animation displays its inside top-level

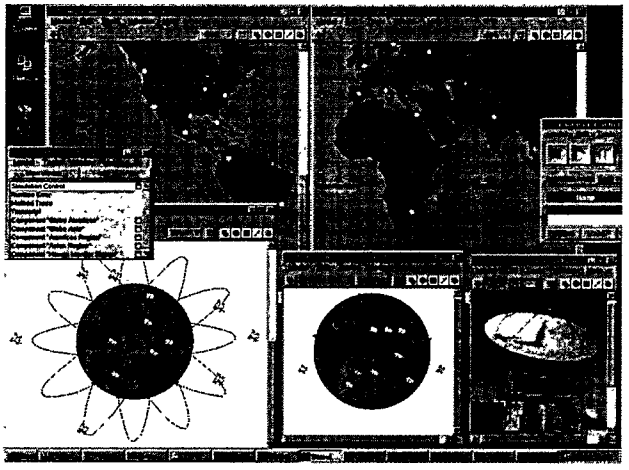

Figure 3: Global Positioning System Visual Simulation

view. Double-clicking on a city image similarly displays its inside top-level view.

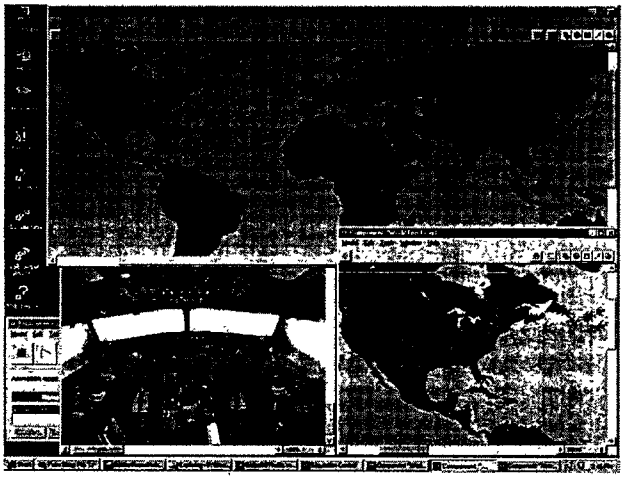

Figure 4: Global Air Traffic Visual Simulation

3.5 Clinic Health Care System

The Clinic model (Figure 5) represents a receptionist, a check-in room, a medical staff room, five examination rooms, a waiting room, and an inside waiting area. It includes physician, physician assistant, nurse practitioner, and nurse, all represented as dynamic objects. The quantity of each medical staff member is given as input. Patients, represented as dynamic objects, arrive with random interarrival times. After seeing the receptionist, they are scheduled to go to the check-in room. A randomly determined medical staff member sees the patient in the check-in room. The patient may be sent to an examination room after check-in. After examination, the patient waits for further examination or checks out and leaves the clinic. A much more complex version of this model is described by Swisher et al. (1997).

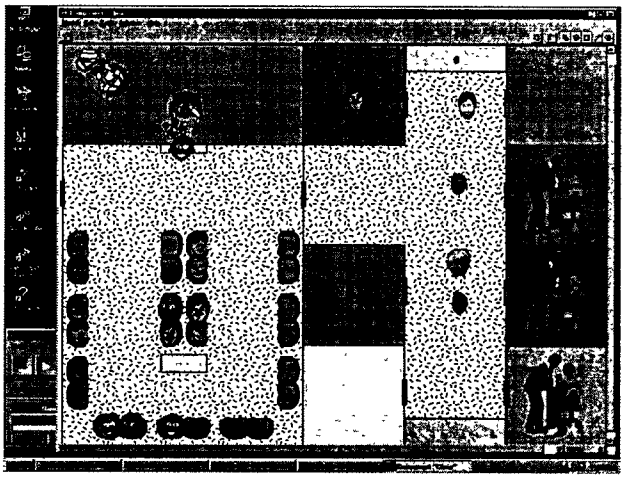

Figure 5: Clinic Visual Simulation

3.6 NCSTRL System

Balci et al. (1998) describe a VSE model of the Networked Computer Science Technical Report Library (NCSTRL) and illustrate how a visual simulation model can be developed for a new NCSTRL configuration by way of component reuse with no programming. Such a repository of reusable components can be created for visual simulation of any digital library for performance evaluation and tuning, and for conducting what-if analyses for different system design configurations.

3.7 Airport

A graphical layout of an airport in Phoenix, Arizona represents the top-level model component. The layout is decomposed into 64 gates and three terminal buildings. Each terminal building is further decomposed into a graphical representation showing the hallways and security check points. This hierarchical graphical decomposition represents the model static architecture.

A Boeing 767 aircraft image is decomposed into its layout, partially shown in Figure 7. The layout is further decomposed into 36 seats for the first class and 190 seats for the economy class. An aircraft is instantiated at runtime to fly from one city airport to another worldwide.

Passenger dynamic objects are instantiated at runtime to arrive in a terminal building with random interarrival

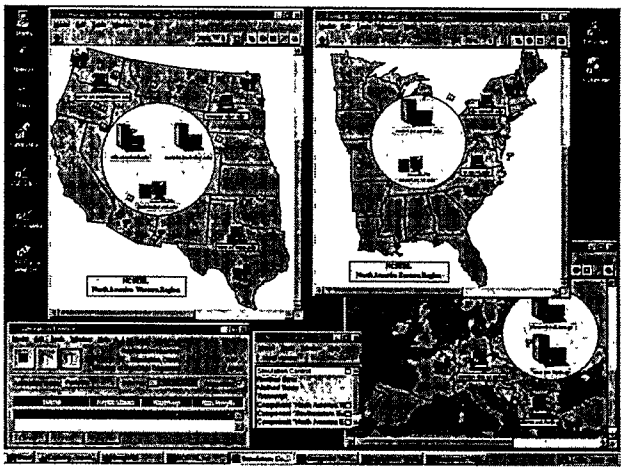

Figure 6: NCSTRL System Visual Simulation

times. Their movements inside the terminals are also modeled. They move into the aircraft dynamic object, navigate through its shallow components, and occupy a seat component. The aircraft moves to the destination airport with all passenger dynamic objects in its layout. The passengers can move within the aircraft layout while the aircraft dynamic object is in motion. The passengers depart the aircraft at the destination airport and move inside a terminal building before exiting.

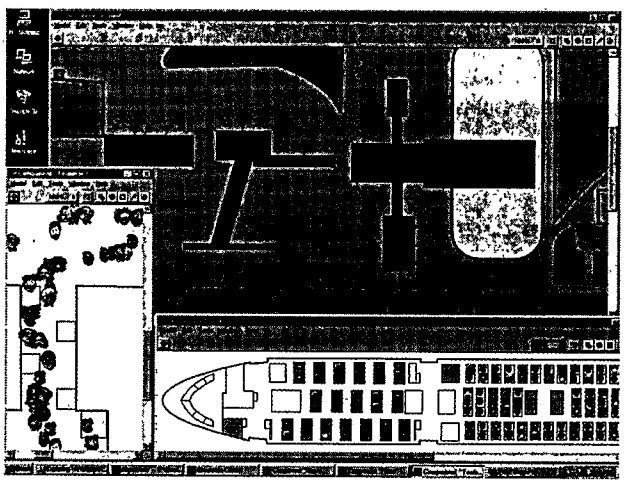

Figure 7: Airport Visual Simulation

4 CONCLUDING REMARKS

The VSE technology enables component-based visual simulation model development with its fully and truly object-oriented capability inherent to its conceptual framework. Model factories can be established for manufacturing model components that can be reused by others in the development of VSE models. (Balci et al. 1997b)

The VSE technology enables the establishment of a component-based simulation modeling marketplace so that the customers of the simulation industry can observe large economic benefits such as reduced costs, increased quality, and enhanced interoperability.

The VSE technology increases automation and productivity in simulation model development by enabling: (a) quality and reliability improvements in simulation models, (b) reduced time to develop and test simulation models, and (c) cost amortization through simulation model component reuse.

The VSE technology increases the productivity of simulation modelers by enabling increased quality through specialization and improved focus on problem solving instead of simulation programming. Expert domain knowledge can be used for creating libraries of reusable model components in specialized areas (e.g., air traffic control, satellite communication, transportation). Expert knowledge use increases model quality and the modelers can reuse high-quality model components rather than trying to create them on their own.

The VSE technology broadens simulation markets for model producers by enabling: (a) creation of systematically reusable simulation model components, (b) increased software interoperability in all phases of model creation and experimentation, and (c) easy adaptation for use in international markets by reusing model components representing systems that employ international standards. (Balci et al. 1997b)

ACKNOWLEDGMENTS

The research that has led to the creation of the Visual Simulation Environment technology since 1983 has been sponsored primarily by the U.S. Navy under research grants totaling $1.3 million. Visual Simulation Environment is a registered trademark of Orca Computer, Inc. (http://www.OrcaComputer.com)

REFERENCES

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

Balci, O., A. I. Bertelrud, C. M. Esterbrook, and R. E. Nance. 1995. A picture-based object-oriented visual simulation environment. In *Proceedings of the 1995 Winter Simulation Conference*, ed. W. R. Lilegdon,

D. Goldsman, C. Alexopoulos, and K. Kang, 1333-1340. IEEE, Piscataway, NJ.

Balci, O., A. I. Bertelrud, C. M. Esterbrook, and R. E. Nance. 1997a. Dynamic object decomposition in the visual simulation environment. In *Proceedings of the 11th European Simulation Multiconference*, ed. A. R. Kaylan and A. Lehman, 69-73. SCS, San Diego, CA.

Balci, O., A. I. Bertelrud, C. M. Esterbrook, and R. E. Nance. 1997b. Developing a library of reusable model components by using the visual simulation environment. In *Proceedings of the 1997 Summer Computer Simulation Conference*, 253-258. SCS, San Diego, CA.

Balci, O., A. I. Bertelrud, C. M. Esterbrook, and R. E. Nance. 1997c. Introduction to the visual simulation environment. In *Proceedings of the 1997 Winter Simulation Conference*, ed. S. Andradóttir, K. J. Healy, D. Withers, and B. L. Nelson, 698-705. IEEE, Piscataway, NJ.

Balci, O., A. I. Bertelrud, C. M. Esterbrook, and R. E. Nance. 1997d. The visual simulation environment technology transfer. In *Proceedings of the 1997 Winter Simulation Conference*, ed. S. Andradóttir, K. J. Healy, D. Withers, and B. L. Nelson, 1323-1329. IEEE, Piscataway, NJ.

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. R. C. Crain, J. R. Wilson, J. J. Swain, and D. Goldsman, 726-736. IEEE, Piscataway, NJ.

Balci, O., and R. E. Nance. 1998. A taxonomy of layout composition techniques for visual simulation. In *Proceedings of the 1998 Summer Computer Simulation Conference*. SCS, San Diego, CA.

Balci, O., C. Ulusaraç, P. Shah, and E. A. Fox. 1998. A library of reusable model components for visual simulation of the NCSTRL system. In *Proceedings of the 1998 Winter Simulation Conference*, ed. D. J. Medeiros, E. Watson, J. Carson, and M. S. Manivannan. IEEE, Piscataway, NJ.

Derrick, E. J., and O. Balci. 1995. A visual simulation support environment based on the DOMINO conceptual framework. *Journal of Systems and Software* 31: 215-237.

Derrick, E. J., and O. Balci. 1997. DOMINO: a multifaceted conceptual framework for visual simulation modeling. *INFOR – Canadian Journal of Operational Research and Information Processing* 35.

Swisher, J. R., B. Jun, S. H. Jacobson, and O. Balci. 1997. Simulation of the Queston Physician Network. In *Proceedings of the 1997 Winter Simulation Conference*, ed. S. Andradóttir, K. J. Healy, D. Withers, and B. L. Nelson, 1146-1154. IEEE, Piscataway, NJ.

AUTHOR BIOGRAPHIES

OSMAN BALCI is President of Orca Computer, Inc., developer of VSE, and an Associate Professor of Computer Science at Virginia Tech. He received B.S. and M.S. degrees from Bogazici University 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 two international journals: *Annals of Software Engineering* and *World Wide Web*; Verification, Validation and Accreditation Area Editor of *ACM Transactions on Modeling and Computer Simulation;* Simulation and Modeling Category Editor of *ACM Computing Reviews;* and serves on five other editorial boards. He is a Director at Large of the Society for Computer Simulation (SCS). He will be a member of the WSC Board of Directors representing SCS in Jan. 1999. Dr. Balci has been a PI or Co-PI on research grants and contracts primarily sponsored by the U.S. Navy with a total funding of $1.3 million. His current research interests center on software engineering, visual simulation and modeling, and world wide web. Dr. Balci is a member of Alpha Pi Mu, Sigma Xi, Upsilon Pi Epsilon, ACM, IEEE CS, INFORMS, and SCS.

ANDERS I. BERTELRUD is a Vice President of Orca Computer, Inc., developer of the Visual Simulation Environment (VSE). 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 and Upsilon Pi Epsilon. He has been working on the development of VSE since September 1992.

CHUCK M. ESTERBROOK is a Vice President of Orca Computer, Inc., developer of the Visual Simulation Environment (VSE). He received a B.S. degree in Computer Science from Virginia Tech in 1996. He has been working on the development of VSE since September 1992.

RICHARD E. NANCE is Chairman of the Board of Orca Computer, Inc. He is also 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. Dr. Nance was the founding Editor-in-Chief of the *ACM Transactions on Modeling and Computer Simulation* (1990-95) He has been the U.S.A. representative to IFIP TC7 on System Modeling and Optimization since 1993. He has served on the editorial boards of numerous journals publishing papers in discrete event 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 and a Distinguished Service Award from ACM Special Interest Group on Simulation in 1995. He was selected as an ACM Fellow in 1996. His current research interests center on computer simulation, including simulation model development, model representation, and diagnosis; software engineering; computer networks; and computer performance evaluation. Dr. Nance is a member of Alpha Pi Mu, Sigma Xi, Upsilon Pi Epsilon, ACM, IIE, and INFORMS.