A Library of Reusable Model Components for Visual Simulation of the NCSTRL System

Osman Balci, Cengiz Ulusaraç, Poorav Shah, and Edward A. Fox
Department of Computer Science
Virginia Tech
Blacksburg, VA 24061, USA

E-mail: balci, culusara, poorav, fox@vt.edu

ABSTRACT
This paper presents a library of reusable model components for visual simulation of the Networked Computer Science Technical Report Library (NCSTRL) and illustrates 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 model components can be created for visual simulation of any digital library for the purpose of performance evaluation and tuning and conducting what-if analyses for different system design configurations.

KEYWORDS: Component-based development, NCSTRL, performance evaluation, reusability, visual simulation

INTRODUCTION

Large-scale digital library (DL) designers and engineers face significant technical challenges in providing adequate response time to users [5, 6]. Substantial advances in hardware technology make the CPU faster and faster every year. However, I/O hardware technology has not kept up. I/O speed continues to be the main hardware performance bottleneck. Better designs and better tuned configurations may help meet this performance challenge.

Building digital libraries with excellent performance requires design evaluation and what-if analysis for which simulation is a very powerful tool. Although developing a simulation model of a complex DL is an onerous task, recent advances in object-oriented visual simulation enable component-based model development with almost no programming.

Component-based development is becoming increasingly important [1, 3]. The NIST advanced technology program on component-based software lists numerous benefits of component-based development [http://www.atp.nist.gov/atp/focus/cbs.htm]. Such benefits can be realized in visual simulation of digital libraries by way of using the Visual Simulation Environment (VSE).

The VSE technology has been developed over the last decade at Virginia Tech and Orca Computer, Inc. under $1.2 million research funding from the U.S. Navy. VSE enables discrete-event, domain-independent, object-oriented, picture-based, component-based, visual simulation model development and execution for solving complex problems [1, 2].

This paper reports on the creation and use of a repository of reusable model components for visual simulation of the NCSTRL system. NCSTRL (pronounced “ancestral”) is a distributed digital library of computer science technical reports from universities and institutions throughout the world. For a description of NCSTRL, please see [4, 7].

The next section describes the characterization of some workload components. Then we present the development of the library of reusable model components for visual simulation of the NCSTRL system. Before concluding remarks, we illustrate component-based visual simulation modeling of the NCSTRL system.

WORKLOAD CHARACTERIZATION

Log data collected at three NCSTRL servers, namely, Cornell, Virginia Tech and UCLA, were used to probabilistically characterize the query inter-generation times (i.e., the time elapsed between the time a server receives a query and the time it receives the next one) for a user population. Data on query inter-generation times was collected over a period of three days. We used the ExpertFit [8] software product to fit a probability distribution to the collected data. Table 1 describes the probability distributions identified as the best fit by the software.

The log data collected at the three servers also was studied in an effort to probabilistically characterize the server response times to queries. However, log entries did not contain sufficiently precise data that could be used for this characterization. Therefore, the triangular probability distribution was used to characterize this stochastic phenomena in the absence of sufficient data. Minimum, mode and maximum values of the response times were obtained from the server log files as shown in Table 2.

Table 1: Probabilistic characterization of query inter-generation times

Dienst Server	Probability Distribution	Location Parameter	Scale Parameter	Shape Parameter
http://cs-tr.cornell.edu:80	Lognormal	0	4.0019	1.2725
http://ei.cs.vt.edu:8090	Lognormal	0	4.10995	1.30315
http://www.cs.ucla.edu:80	Lognormal	0	4.02783	1.15034

The transmission time of requests from one server to another was determined by examining log entries. By collecting performance data at different times of the day, we characterized the network delay at different servers. Broadly, these delays were classified into two categories: *normal* (uniformly distributed between 2 and 5 seconds) and *heavy* (uniformly distributed between 8 and 12 seconds).

Table 2: Probabilistic characterization of server response times

Dienst Server	Probability Distribution	Minimum	Mode	Maximum
http://cs-tr.cornell.edu:80	Triangular	0.768	1.678	4.234
http://ei.cs.vt.edu:8090	Triangular	0.918	2.153	3.623
http://www.cs.ucla.edu:80	Triangular	1.645	2.907	6.342

DEVELOPMENT OF THE LIBRARY OF REUSABLE MODEL COMPONENTS

The library contains nine reusable model components as shown in Figure 1. Each component is described below.

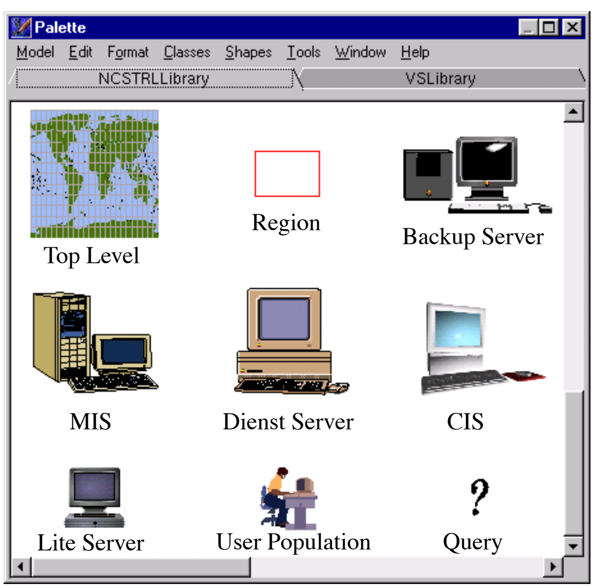

Figure 1: NCSTRL reusable model components

Each instantiated component or an object in VSE is given a unique identification called *object reference*. This reference is used for message passing.

Top Level

This component models the interactions among the NCSTRL regions in the world. It belongs to the VSE class GlobalSystem and exhibits the following behavior (methods).

- *iAmRegion:* is sent by all the regions within the system. It contains a reference to the region as well as all the MIS (Merged Index Server) servers within the region.
- *CISServerIs:* The region containing the CIS (Central Index Server) server sends this message to the Top Level so that its identity can be sent to all other servers in the system by the Top Level.
- *allMISServersAre:* calls the "MISServersAre" method in each region and passes a list of references to all MIS servers within a system.
- *replicationStarted:* performs some initialization for the simulation start-up.
- *Initialization*: sends the references (identities) of the CIS server and MIS servers to the regions in the system at simulation start-up.

Region

This component models the interactions among the servers and user populations in a region. It belongs to the VSE class Region and exhibits the following behavior (methods).

- *iAmDienstServer:* is sent by each Dienst Server within the region. The region composes a list of all Dienst or Lite servers within that region and sends the list along with the MIS and Backup server references to all the servers within the region. It also sends all the MIS and CIS server references to the Top Level along with its own reference.
- *iAmLiteSite:* is sent by each Lite server in the region.
- *iAmMISServer:* is sent by each MIS server in the region.
- *iAmCISServer:* is sent by the CIS server in the region.
- *iAmBUPServer:* is sent by the Backup server in the region.
- *otherServersAre*: is sent by the Top Level to all regions within the system. It provides a list of references of all MIS and CIS servers in the system.

- *dynObjAscended*: is sent either when a query is sent from user to Dienst server or when a reply is sent from a Dienst server to a user. Ascending dynamic object (query or reply) is routed to its proper destination.
- *replicationStarted*: performs some initialization for the simulation start-up.
- *isBUPAvailable*:checks if there is a backup server in the region.
- *revealIdentity*:sends the references (identities) of the Region and its MIS server(s) to the Top Level it is contained in.
- *sendLocalInformation*: is sent by the region to all servers within the region and it contains a list of references of all Dienst servers, MIS server, Backup server in the region and the CIS server (if any).
- *sendMISInformation*: is sent by the region to MIS server(s) in the region and it contains a list of all MIS server references in the system.

Dienst Server

This component models the behavior of a Dienst server that receives a user query, searches its own repository of technical reports, and simultaneously sends the query to all other Dienst and MIS servers in its region. If some server does not respond within the timeout limit, the Dienst server then sends the query to its region's backup server, if any. When all contacted servers respond with query results, the Dienst server returns the merged results to the user.

This component belongs to the VSE class DienstServer and exhibits the following behavior (methods).

- *replicationStarted:* sends the reference (identity) of the Dienst server to the region it is contained in at simulation start-up. The region collects all the Dienst server references within it and sends the list to each server.
- *receiveServerList:* returns a list of references of all Dienst servers within a particular region.
- *dynObjArrived:* indicates the arrival of a query submitted by the user. If the server status in the model is found to be operational, a "searchFor" message is sent to all other Dienst servers, the MIS server within the region and the system CIS server (if any). The server also sets the number of replies expected to the query. It schedules a "timeOut" event for that query, whereupon it checks if all expected responses for that query are received.
- *timeOut:* checks if all the expected responses to a particular query are received. If not, it accesses the Backup server of the region (if any). It schedules another "timeOut", whereupon a check is made to see if the Backup server has responded within the time limit.
- *backupServerTimeOut:* checks if the Backup server of the region responded to the query. If not, the Dienst server sends all the responses it has been able to collect so far to the user and informs the user of the servers that failed to respond within the time limit.
- *searchFor:* is sent to a Dienst server by another Dienst or Lite server within the region as part of a distributed search for the query received by the other server. If the server receiving the message is up, it schedules the arrival of a response to the query at the sending server after a time which is the sum of the search time and the network delay between the two servers.
- *replyToQueryArrived:* is sent to a Dienst server by all other Dienst servers, the MIS server within the region and the system CIS server (if any) in response to a search request made by the server. It indicates the arrival of a reply to the search from the remote server.

Lite Server

This component models all of the Dienst server's behavior except that it does not perform a search on its own since it does not possess a repository of technical reports. It belongs to the VSE class LiteSite and exhibits the following behavior (methods).

- *replicationStarted:* sends the reference (identity) of the Lite server to the region it is contained in. The region collects all the Lite server references within itself and sends them to each server.
- *serversWithinRegionAre:* returns a list of references of all Dienst, MIS, Backup (if any), and CIS (if any) servers within a particular region.
- *dynObjArrived:* indicates the arrival of a query submitted by the user. The server checks if it is operational by generating a uniform random number between 0 and 1 and comparing it with the probability of its going down, and if so, it then sends a "searchFor" message to all Dienst servers, the MIS server within the region and the system CIS server. It also sets the number of replies expected to the query. It schedules a "timeOut" event for that query, whereupon it checks if all expected responses for that query have been received.
- *timeOut:* checks if all the expected responses to a particular query have been received. If not, it accesses the backup server of the region (if any). It schedules another "timeOut", whereupon a check is made to see if the backup server has responded within the time limit.
- *backupServerTimeOut:* checks if the backup server of the region has responded to the query. If not, the Lite server sends all the responses it has been able to collect so far to the user and informs the user of the servers that failed to respond within the time limit.
- *replyToQueryArrived:* This message is sent to a Lite server by all the Dienst servers, the MIS server within the region and the system CIS server (if any) in response to a search request made by the Lite server. It indicates the arrival of a reply to the search from the remote server.

Merged Index Server (MIS)

This component models a Merged Index server which possesses the indices of the Dienst servers of the NCSTRL regions other than its own. It is accessed by the Dienst and Lite servers in its region. It performs a search on its indices and returns the result to the requesting server. It belongs to the VSE class MISServer and exhibits the following behavior (methods).

- *replicationStarted:* sends the reference (identity) of the MIS server to the region it is contained in. The region then sends the MIS server reference to all other servers within the region.
- *serversWithinRegionAre:* provides a list of references of all Dienst servers within a particular region.
- *allMISServersAre:* provides a list of references of all MIS servers in the system.
- *searchFor:* is sent to the MIS server by a Dienst or Lite server within the region as part of a distributed search to a query received by the Dienst or Lite server. If the MIS server is up, it schedules the arrival of a response to the query at the sending server after a time, which is the sum of the search time and the network delay between the two servers.

Central Index Server (CIS)

This component models a Central Index server which possesses the indices of all the Lite servers. It is accessed by the Dienst and Lite servers. It belongs to the VSE class BackUp and exhibits the following behavior (methods).

- *replicationStarted*: sends the reference (identity) of the CIS to the region it is contained in at simulation start-up.
- *searchFor*: is sent to the CIS server by a Dienst or Lite server within the region as part of a distributed search to a query received by the Dienst or Lite server. If the CIS server is up, it schedules the arrival of a response to the query at the sending server after a time, which is the sum of the search time and the network delay between the two servers.

Backup Server

This component models the behavior of a Backup server which possesses the indices of all the Dienst and MIS servers in its region. It is accessed by a Dienst or Lite server when the server times out on a query. The Backup server conducts the search over its indices and returns the results to the requesting server. It belongs to the VSE class BackUp and exhibits the following behavior (methods).

- *replicationStarted:* sends the reference (identity) of the Backup server to the region it is contained in. The region then sends the Backup server reference to all other servers within the region.
- *serversWithinRegionAre:* provides a list of references of all Dienst servers within a particular region.
- *searchFor:* is sent to the backup server by a Dienst or Lite server within the region if it times out on a response to a search from one or more Dienst or MIS or CIS server in the system. If the backup server is up, it schedules the arrival of a response to the query at the sending server after a time, which is the sum of the search time and the network delay between the two servers.

User Population

This component models the submission of queries from a user group to a particular Dienst or Lite server. It belongs to the VSE class UserGroup and exhibits the following behavior (methods).

- *dynObjectEnteredModel:* schedules the random submission of the next query to the server.
- *dynObjArrived:* indicates the arrival of a response from the Dienst server to the query submitted by the user. It retrieves the query submission time and calculates the total time spent in responding to the query.
- *replicationStarted:* schedules the submission of the first query from the user group upon simulation start-up.

Query

This component models the characteristics and behavior of a user query in the form of either a dynamic object or an object-oriented message. For the purpose of animating the movements of queries in the system, the dynamic object representation is used. For the purpose of conducting long experiments without animation, the message form is used.

This component belongs to the VSE class Query and exhibits the following behavior (methods).

- *setNumberOfRepliesExpected:* sets the number of replies expected for each query sent by the user.
- *returnNumberOfRepliesExpected:* is used by the Dienst server originally receiving the query to check how many replies are received for each query submitted to that Dienst server from the user population.
- *setLocalServer:* sets the destination server of the query to the local server of the user.
- *returnLocalServer:* is called by the region to determine the destination of a query and to send the query to that destination.
- *setOriginator:* is called by the user originating the query to set the user's object reference in the query so that a server can distinguish between queries sent by multiple users.
- *returnOriginator:* is called by the region in order to determine the destination of a query and to send the query to the proper destination.
- *decrementRepliesExpected:* is called by the Dienst server which originally received the query. Each time the above mentioned server receives a response from another

server, the number of replies expected for that particular query is decreased by 1.

- *recordItsSubmissionTime:* records the query submission time to a Dienst or Lite server which is used to determine the average query response time in the NCSTRL system.
- *getSubmissionTime:* returns the query submission time.

COMPONENT-BASED SIMULATION MODELING OF THE NCSTRL SYSTEM

In this section, we illustrate the development of a visual simulation model of a configuration of the NCSTRL system with no programming by way of reuse of components from the library (repository) described in the previous section.

Using the VSE Editor tool, we create a new VSE model and make it depend on the NCSTRL library model by using VSE's Dependency window [1]. All reusable components appear under the "NCSTRL Library" tab as shown in Figure 1. We instantiate a top level representation by drag and drop of the Top Level reusable component in the Palette (Figure 1). Then we drag and drop the Region component from the Palette and instantiate three regions on the world map as shown in Figure 2.

The browser on top of the window in Figure 2 shows the model's top-down hierarchical decomposition. We browse through the model hierarchy by either using the browser or by double-clicking the deep components in the hierarchy. Clicking "North America Western Region" in the browser displays the corresponding component layout. We instantiate, by way of drag and drop from the Palette, the servers and user populations as shown in Figure 3.

After dragging and dropping a component from the Palette, the instantiated component is selected and the Inspector tool is invoked. Using the Inspector, the characteristics (instance variables) of the component are specified by entering values and graphically connecting object references.

Similarly, we instantiate the servers and user populations in the "North America Eastern Region" and "Southern and Central Europe" components as depicted in Figure 4 and Figure 5, respectively.

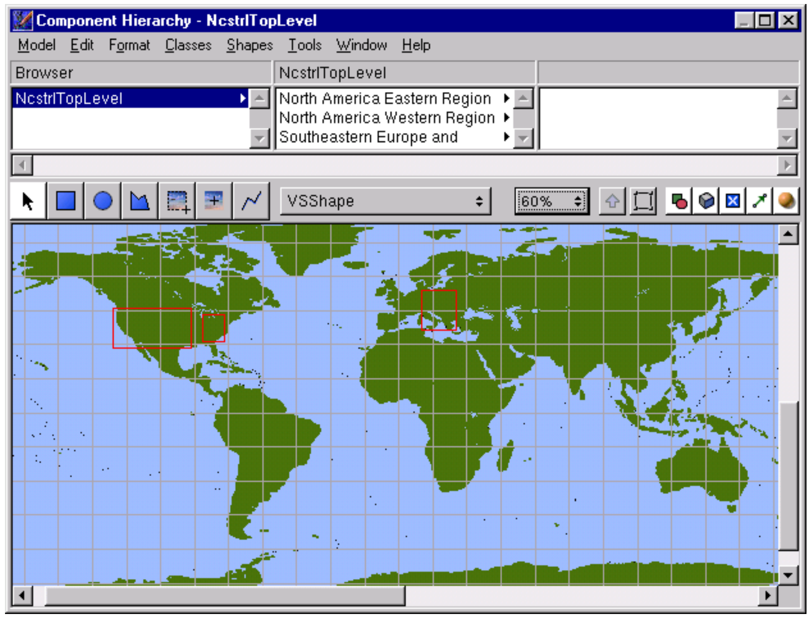

Figure 2: Top level decomposition

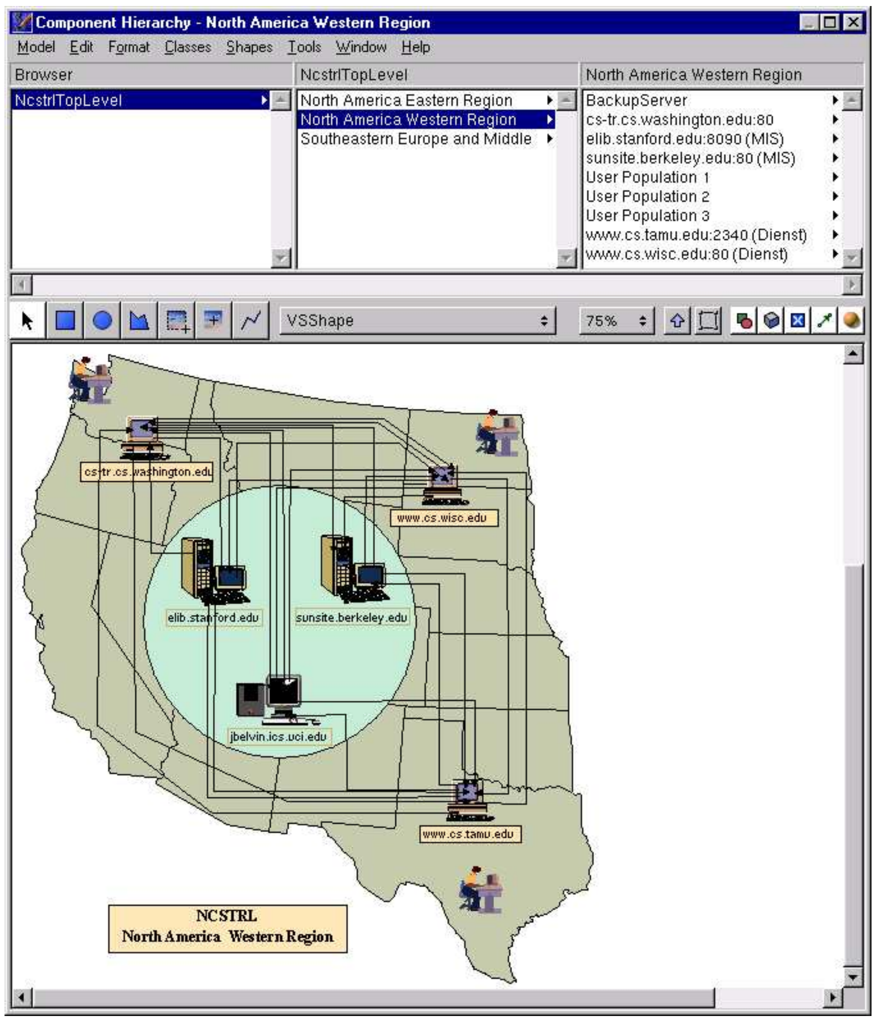

Figure 3: North America western region

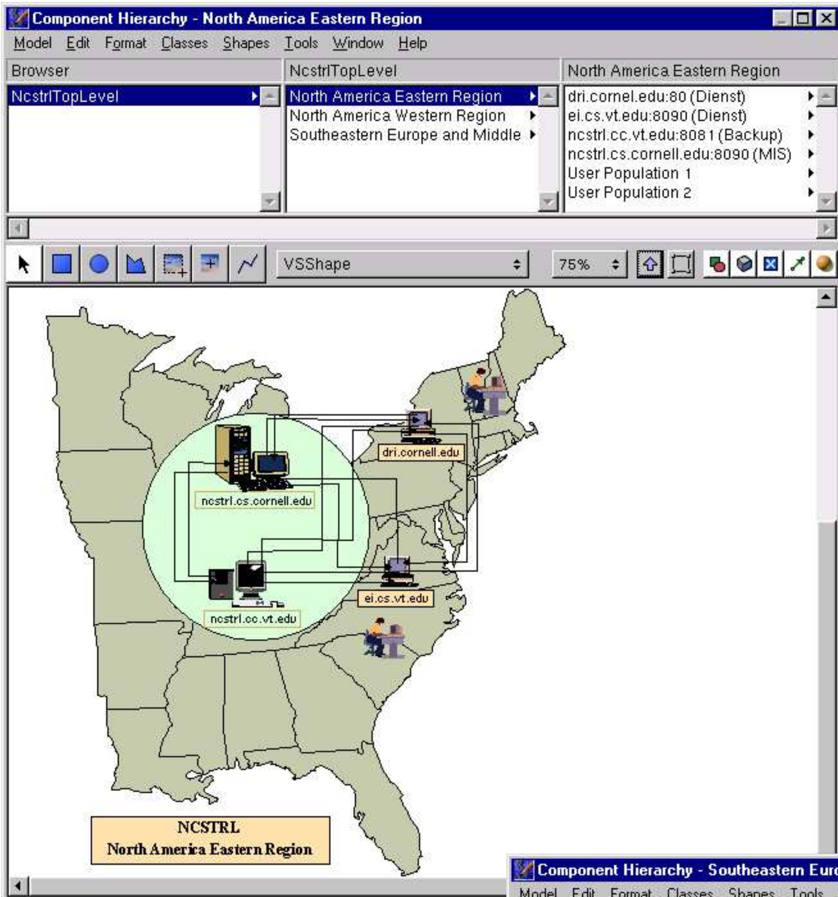

Figure 4: North America eastern region

After the model specification through reuse is completed in the VSE Editor, the "Prepare for simulation" menu option is invoked to generate the executable version of the model automatically. The VSE Simulator tool is launched to provide the run-time environment. The animation of the NCSTRL model is shown in Figure 6. The animation speed can be adjusted by using the slider of the Simulation Control window shown in Figure 6. The animation can be turned on or off throughout the model execution by selecting or deselecting the Animation button.

The Simulation Control window enables the specification of a statistical experiment with the model. The experiment specification shown in Figure 6 indicates that the simulation model is replicated 20 times, the model is warmed-up for 1000 queries in each replication, and the model is run in steady state for 10,000 queries.

Statistical data collected during model execution is written either to files within the model structure or to text files outside of the model structure. VSE Output Analyzer is used to construct confidence intervals for the model's response variables or performance measures. Other statistical analysis or graphics (plotting) software can be used to statistically and graphically analyze the simulation output data.

The drag-and-drop reuse enables the creation of complex models of different configurations of the NCSTRL system with no programming whatsoever. A domain expert can easily build complex NCSTRL models and experiment with them for a particular purpose such as (a) comparing different NCSTRL operating policies, (b) evaluating the NCSTRL performance under a particular configuration, (c) predicting the NCSTRL performance under varying levels of workload, (d) sensitivity analysis to determine the parameters and variables which significantly influence the NCSTRL performance, (e) tuning the NCSTRL performance, and (f) performing "what-if" analysis.

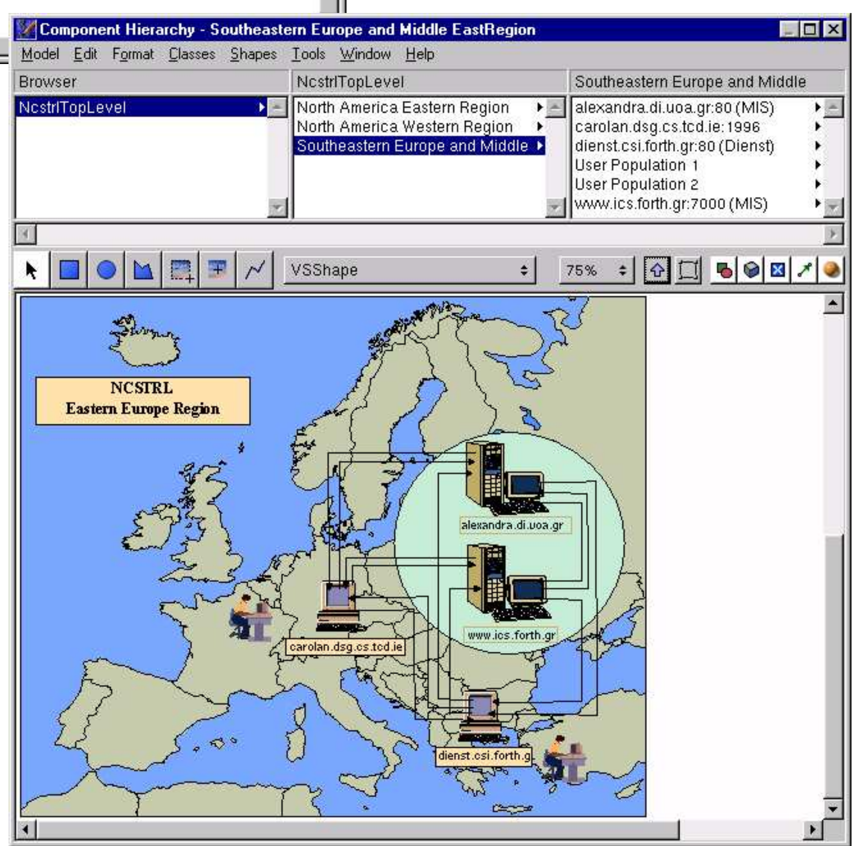

Figure 5: Southeastern and central Europe region

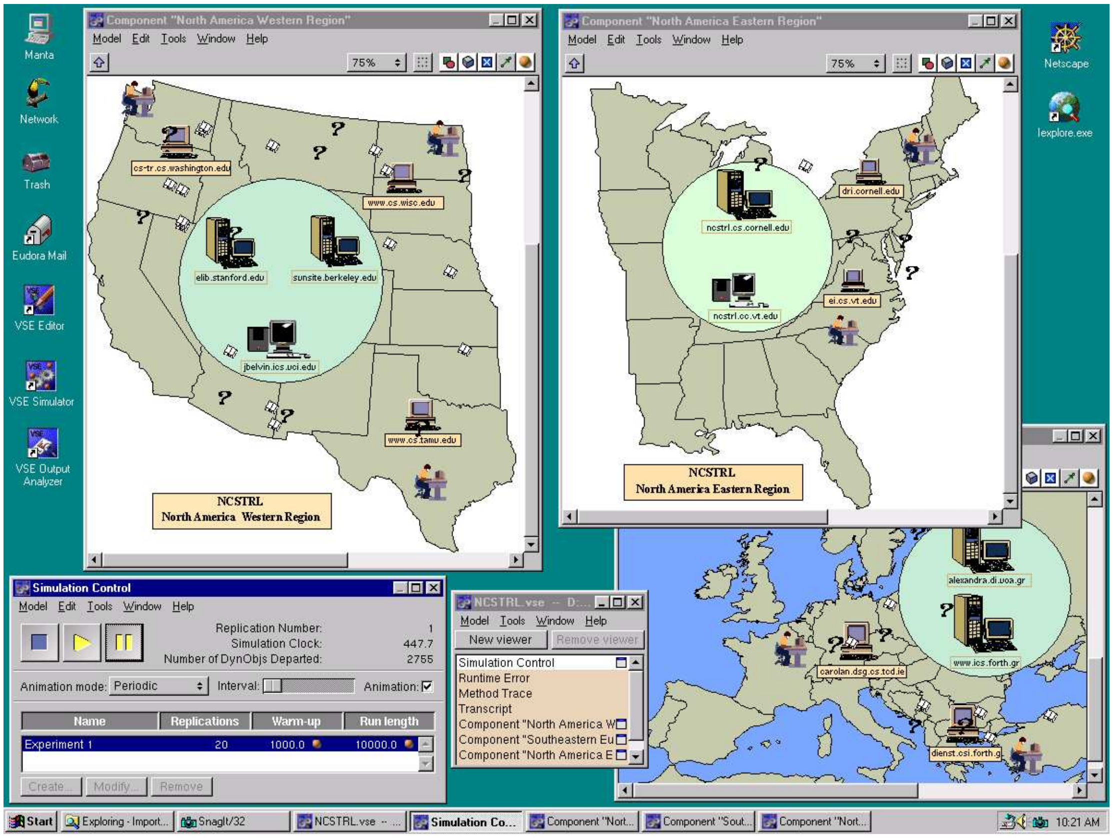

Figure 6: Animation

CONCLUDING REMARKS

Component-based visual simulation model development of the NCSTRL system is illustrated by using a library of reusable model components. Reuse at the component level brings complex visual simulation modeling to the domain experts who are not necessarily simulation analysts.

More work is needed to better characterize different parts of the workload and random phenomena. In order to collect the low-level data required for such characterization, NCSTRL must be instrumented internally for data collection. Upon collection and analysis of much more data, we will be in a position to characterize the stochastic elements within the NCSTRL system with sufficient credibility.

ACKNOWLEDGMENT

The authors thank Naomi Dushay from Cornell University for providing technical information about the NCSTRL system.

REFERENCES

1. Balci, O., Bertelrud, A.I., Esterbrook, C.M. and Nance, R.E. Developing a Library of Reusable Model Components by Using the Visual Simulation Environment, in *Proc. 1997 Summer Computer Simulation Conference* (Arlington, VA, July 13-17,1997), SCS, San Diego, CA, pp. 253-258.
2. Balci, O., Bertelrud, A.I., Esterbrook, C.M. and Nance, R.E. Introduction to the Visual Simulation Environment, in *Proc. 1997 Winter Simulation Conference* (Atlanta, GA, Dec. 7-10,1997), IEEE, Piscataway, NJ, pp. 698-705.
3. Brown, A.W., Ed. *Component-Based Software Engineering*. IEEE Computer Society Press, Los Alamitos, CA, 1996.
4. Davis, J.R. and Lagoze, C. The Networked Computer Science Technical Report Library. Technical Report 96-1595, July 1996, Department of Computer Science, Cornell University, Ithaca, NY.
5. Fox, E.A., Akscyn, R.M., Furuta, R.K. and Leggett, J.J., Guest Editors. Introduction to the Special Issue on Digital Libraries. *Communications of the ACM 38*, 4 (Apr. 1995), 23-28.
6. French, J.C. and Viles, C.L. Ensuring Retrieval Effectiveness in Distributed Digital Libraries. *Journal of Visual Communication and Image Representation 7*, 1 (Mar. 1996), 61-73.
7. Lagoze, C. and Davis, J.R. Dienst - An Architecture for Distributed Document Libraries. *Communications of the ACM 38*, 4 (Apr. 1995), 47.
8. Law, A.M. and Vincent, S.G. *ExpertFit User's Guide*. Averill Law & Associates, Tucson, AZ, 1997.