Proceedings of the 1999 Winter Simulation Conference
P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, eds.

MODULAR SIMULATION ENVIRONMENTS: AN OBJECT MANAGER BASED ARCHITECTURE

Charles R. Standridge

Padnos School of Engineering
Grand Valley State University
301 West Fulton
Grand Rapids, MI 49504-6495, U.S.A.

ABSTRACT

To perform a simulation project, simulationists employ simulation specific software tools, general purpose software tools, and perhaps software developed to meet the needs of a particular project. Ideally, these divergent tools would work together in a seamless simulation environment. Modular simulation environments are one way of meeting this goal. Software tools can be added to or deleted from a modular simulation as needed. Thus, the simulation environment can be configured on a project by project basis or even dynamically during the course of a project. The flow of data between the tools in the environment is also a primary concern. An object manager based architecture provides the capabilities to add and delete software tools as necessary as well as to control the flow of data between the software tools. Each software tool and each data set can be viewed as an object with certain attributes. The object manager controls the invocation of the software tools as well as meeting input data requirements and organizing and managing the results of each operation. The design of such an object manager is presented. An example modular simulation environment is given and its configuration and operation illustrated.

1 INTRODUCTION

Ideally, each simulationist would be able to select the set of software tools to use on each simulation project (Standridge and Centeno 1994). The selection would be based on the particular requirements of that project. The tool set would contain both simulation specific tools such as model builders and simulation engines as well as tools with wide applicability such as word processors, statistical analysis packages, and spreadsheets.

Modular simulation environment concepts seek to provide the standard by which ad hoc collections of software tools can be used together to perform a simulation project. These concepts specify how simulation related data flows between tools in a general way so that heterogeneous software can work together (Standridge, et al., 1996).

Some benefits of modular simulation environments are as follows:

1. High flexibility for end user selection of simulation software. This supports simultaneous use of simulation software from multiple software providers as well as locally developed tools.
2. Use of widely applicable software with which the simulationist is already familiar.
3. Inclusion of tools such as spreadsheets, word processors and presentation graphics generators that have not traditionally been a part of simulation environments.
4. Management and selective use of simulation inputs and results.
5. Definition of a standard for simulation environment structure. Because of its flexibility, simulationist may generally adopt such a standard and tool builders may find developing software compatible with the requirements of the standard helpful.

The specification of an implementation architecture that supports the dynamic inclusion of software tools and the management of simulation data is one significant issue in the development of modular simulation environments. Object management is one approach that shows promise in successfully addressing this issue. This paper presents an object manager based architecture for modular simulation environments. An example simulation enviroment is presented and the operations that it can perform illustrated.

2 SIMULATION ENVIRONMENT OVERVIEW

Traditional simulation environments have three major components: (1) a fixed set of software tools prescribed by the environment designers and implementers, (2) a

database management system that controls the flow of data between the software tools transparently to the environment users, and (3) a user interface that gives access to all environment capabilities. Each software tool uses existing information in the database and adds the results of its own operations to the database. Furthermore, these simulation environments may be divided into two major categories: those that emphasize model building tools and those that emphasize experimentation and tools for handling simulation related data.

TESS (Standridge 1985; Standridge and Pritsker 1987) was an early, pre-graphical user-interface, simulation environment emphasizing data handling and experimentation. Basic modeling tools provided for building SLAM II network models and editing sets of SLAM II control statements. The database manager organized simulation inputs and outputs. Simulation results could be collected automatically from SLAM II, GPSS/H, and MAP/1 simulations. Software tools supported constructing animations, graphing data, and reporting data. A command language served as the user interface. Selective querying of the database was supported. A database subprogram library allowed computer programs to store and retrieve data from the TESS database.

Balci and Nance (1987, 1992) designed and implemented a simulation model development environment. This work has lead to the visual simulation environment (Balci et al. 1995). A visual user interface supports the development and simulation of visual models that may be hierarchical and object oriented. The collection of tools includes a visual editor, a library of existing component models, a static model analyzer, a model dynamics tester, a simulator, an analyzer for simulation results, a multi-media learning support system, and an evaluator for accessing the credibility of a simulation study.

Fritz, Sargent, and Daum (1995) report the development of a similar simulation environment for building and executing hierarchical control flow graph models.

Modular simulation environments are distinct from these existing environments in that the set of software tools are not pre-determined, the use of widely applicable software is encouraged, and the user-interface relies on the standards set by existing graphical user interfaces. Data and software tool management must accommodate the open-ended nature of the environment.

3 THE OBJECT MANAGER APPROACH

A simulation environment consists of a set of tools as well as the data needed for input to the tools and the data resulting from the use of the tools. In an object manager-based architecture, each tool as well as each input data set and output data set is considered to be an object. Each object may be characterized by attributes that can be given values. The AVS/Express visualization software environment employs such a object manager architecture. (Cunningham 1998).

The object manager controls the definition of each object and the assignment of values to object attributes as well as the invocation of tools. Object manager control includes making sure that object attribute values are of the proper type and within the proper range as well as assuring that all inputs to a tool object are acceptable before the tool is invoked.

In a modular simulation environment with an object manager-based architecture there are two basic classes: tools and data sets. Tools may be simulation-specific such as a graphical model builder or even an entire simulation environment. Alternatively, tools may be general purpose such as a spreadsheet or a graphics package. Furthermore, tools may be from another domain such as an optimization package. A tool may be invoked independently or from within another tool. For example, a simulation may stop periodically to invoke an optimization to determine how much product to produce and inventory to ship over the next four weeks.

Data sets result from the operation of tools or are constructed as input to tools. A data set object may be output from one tool and input to another. Data sets may be generic sets of values or have a simulation specific structure such as a time series of time persistent value observed during a simulation run, e.g. the volume of work in progress. Standridge et al. (1996) characterize the types of data set subclasses associated with simulation inputs and outputs.

A data set object may have a specific structure as required by a particular tool for input or resulting from the use of a tool. For example, the Proof animation package requires a data set of instructions to create an animation.

4 AN OBJECT MANAGER BASED SIMULATION ENVIRONMENT

Figure 1 shows the object manager based architecture for the modular simulation environment under development at the Padnos School of Engineering, Grand Valley State University. This environment will be used in teaching both undergraduate and graduate manufacturing engineering students as well as supporting simulation projects in partnership with local industry.

Some tool objects in the environment are simulation specific in nature (AweSim, Proof, ExpertFit), some are general purpose software (EXCEL, Text Editor), some are from the optimization domain (GAMS) (Brooke, et al. 1998), and some are developed specifically to support the operation of the environment (List Builder, Data Collection, and Data Set Operations).

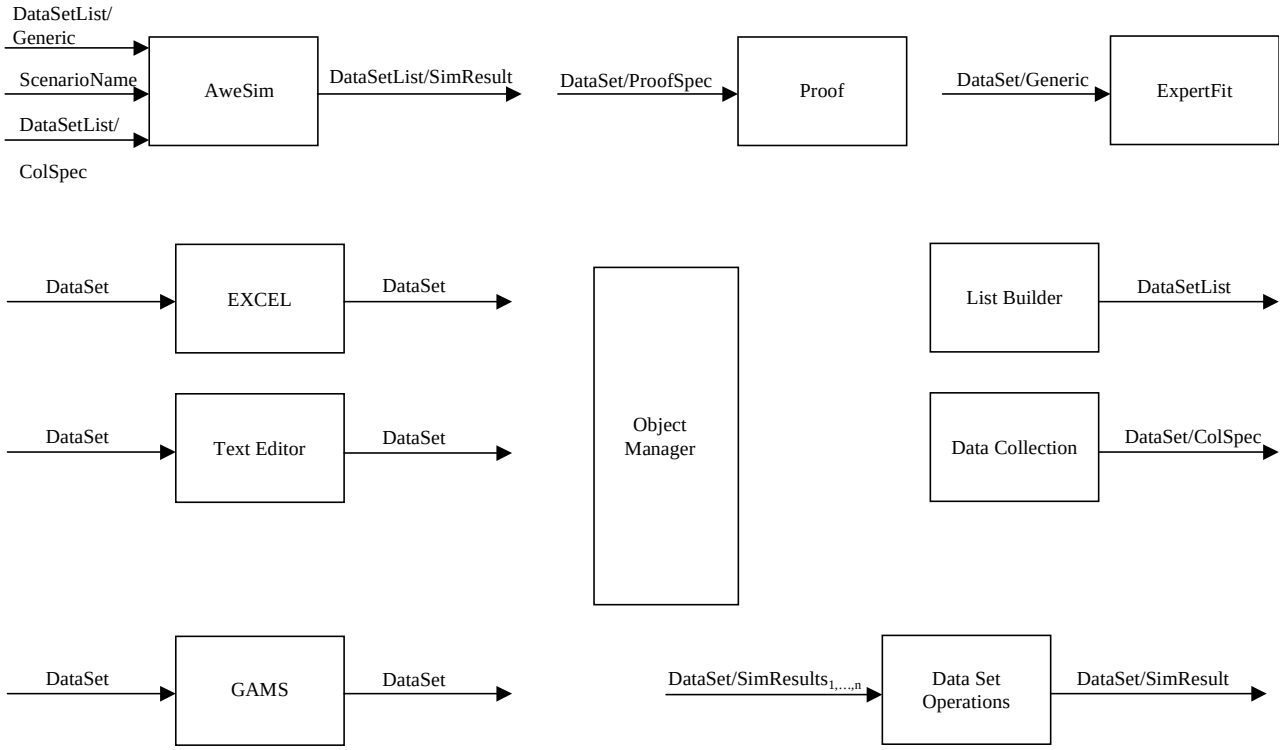

Figure 1: Example Object Manager Based Architecture for a Modular Simulation Environment

The modular simulation environment can include other simulation environments as objects. AweSim (Pritsker, O'Reilly, and Laval 1997) is itself a simulation environment. It is used within the modular simulation environment as it would be in a stand alone mode. However, the data sets resulting from AweSim simulations are organized and processed in a systematic manor within the modular environment instead of on an ad hoc basis. Each is labeled with the name of a scenario that identifies the specific experimental conditions for a particular simulation. In addition, input data sets setting specific experimental conditions and model parameter values are organized and labeled systematically.

The modular simulation environment can link heterogeneous commercial software products. AweSim, Proof (Henriksen 1996), and ExpertFit (Law and McComas 1996) are all commercial simulation software products but are not integrated with each other. The modular simulation environment provides the structure for the flow of data between these tools. AweSim simulations can produce data sets for animation by Proof. Distributions fit to data by ExpertFit provide the basis for random sampling within AweSim simulations.

The modular simulation environment can link simulation specific software and general purpose software. AweSim simulation results can be processed by EXCEL to produce statistical summaries, histograms, and graphs. AweSim inputs could be prepared using a text editor. In this way, various versions of each input data set could be prepared and managed by the simulation environment. Each version corresponds to a unique set of simulation parameters defining a different scenario to evaluate. Version support is not provided by AweSim.

The modular simulation environment can support the joint use of simulation and other analysis techniques such as optimization. Optimization models can be employed to make decisions periodically within simulation runs or to set the parameter values of a simulation run. The modular environment provides the data links between the simulation and optimization as well as the mechanism for invoking the optimization from the simulation.

The modular simulation environment contains its own tools for managing data. One tool, data collection, provides the mechanism for specifying which simulation results are collected and how to integrate them into the modular environment. Another tool, list builder, produces lists of data sets to input to a simulation as well as lists of data sets resulting from a simulation. A third tool, data operations, provides merge, extract, and elementary computation operations on data sets of simulation results. Thus, additional data sets are created.

The modular simulation environment provides alternative mechanisms for performing the same tasks. A user is free to choose among the mechanisms at will. For example, simulation input data could be prepared using either the text editor or the EXCEL spreadsheet.

The modular simulation environment provides for the organization and management of data set objects that can be further classified into various types. Data set objects may be generic or of a specific type such as simulation results, Proof specifications, and collection specifications. Simulation results and Proof specifications may be further identified by the simulation scenario and replication from which they were generated. Each generic data set may have multiple versions representing alternative simulation input values.

The object manager can perform the following operations:

1. Define new tool – A new tool enters the modular simulation environment.
2. Remove tool – A tool departs the simulation environment.
3. Run tool – A tool in invoked.
4. Define data set – A new data set enters the simulation environment.
5. Remove data set – A data set leaves the simulation environment.

The operations may be performed from within a tool via an application program interface or from the user interface of the object manager.

Typical object attribute types are numeric, string, data set, and data set list.

5 EXAMPLE OPERATION OF A MODULAR SIMULATION ENVIRONMENT

Consider the following example application of the modular simulation environment shown in Figure 1. A manufacturing firm supplies goods sold by a large retailer. A simulation is required to determine the re-order point for the retailer along with the manufacturer's inventory level that would allow production to be suspended for one day.

The simulation analyst assigned to the project determines that two types of input data sets are required. One contains values of the parameters of the inventory system whose values are changed from run to run: the retailer's reorder point and the manufacturers production cut-off point and is called EXPERIMENT. The other contains values for the remaining operating parameters of this simple supply chain and is called PARAMETERS. There is one version of the PARAMETERS data set called BASELINE and as many versions of the EXPERIMENT data set as scenarios to be evaluated by the simulation. Two versions of the latter called LOWREORDER and HIGHREORDER define alternative scenarios.

The two data sets EXPERIMENT and PARAMETERS are defined in the simulation environment using object manager commands.

In the following, all tools are invoked using the object manager commands.

Values for PARAMETERS version BASELINE and EXPERIMENT versions LOWREORDER and HIGHREORDER are entered using the text editor. Note that alternatively an EXCEL spreadsheet could have been used for this task.

The list builder is used to create two lists of input data sets:

1 LOW containing PARAMETERS version BASELINE and REORDER version LOWREORDER.
2 HIGH continuing PARAMETERS version BASELINE and REORDER version HIGHREORDER.

The list builder is used to specify that a data set containing two time series: the inventory at the retailer and the inventory at the manufacturer is a simulation output. This list is called INVENTORY.

Two scenarios are simulated. The name of the scenario is the same of the name of the input data set list used in the simulation, LOWREORDER and HIGHREORDER. One simulation result data set is produced for each scenario using standard AweSim capabilities: INVENTORY version LOWREORDER and INVENTORY version HIGHREORDER.

Graphs of the inventory times are desired. EXCEL is invoked and the data set INVENTORY version LOWREORDER loaded. The graph is created using EXCEL commands. This process is repeated for the data set INVENTORY version HIGHREORDER.

In each scenario, the inventory levels are observed and recorded at the same points in time. It is of interest to have a data set containing the inventory at the retailer in each scenario as well as a time series of the difference between the scenarios. The data set operations tool can be used to merge the data sets INVENTORY version LOWREORDER and INVENTORY version HIGHREORDER, extract only the time series of inventory levels, and compute the difference. This creates a new data set named by the user: DIFFERENCE version LOWHIGH.

GraphiC is a C programming language based subroutine library for generating high quality graphs. A GraphiC application program is written for generating graphs of the inventory levels. This program with attributes data set list is added to the modular simulation environment shown in Figure 1 using the define new tool capability of the object manager. The data set list contains the names of the data sets with information to graph.

6 SUMMARY

A modular simulation environment meets the requirement to select the software tools needed to perform each simulation project. Simulation specific tools and general purpose tools as well as tools developed particularly for the project can be included. The data flow between the tools can be defined. New tools may be added as needed during the course of the project.

An object manager architecture is an effective implementation strategy for a modular simulation environment. Each tool and data set in the environment is viewed as an object under the control of the object manager. The object manager organizes the data sets and invokes the tools as needed, either independently or from within other tools. It ensures that the inputs to tools and the results from tools are complete and acceptable as well as integrating them into the environment.

A demonstration modular simulation environment is under development in the Padnos School of Engineering at Grand Valley State University. This environment will support teaching undergraduate and graduate simulation courses as well as performing projects with local industry. The environment will include simulation specific software such as a traditional simulation environment, fitting distributions to data, and animation; general purpose software such as spread sheets and text editors; optimization software; and data management software developed specifically for the environment.

An example simulation project performed using the demonstration environment has been presented.

REFERENCES

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

Balci, O. and R. E. Nance. 1992. Simulation model development environment. In *Proceedings of the 1992 Winter Simulation Conference*, ed. J. J. Swain, D. Goldsman, R. C. Crain, and J. R. Wilson, 726-735. IEEE, Piscataway, New Jersey.

Balci, O., A. I. Bertelrun, 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. C. Alexopoulos, K. Kang, W. R. Lilegdon, and D. Goldsman, 1333-1340. IEEE, Piscataway, N.J.

Brooke, A., D. Kendrick, A. Meeraus, and R. Raman. 1998. *GAMS: A User's Guide*. GAMS Development Corporation, Washington, D.C.

Cunningham, I. 1998. Creating Engineering Visualization Applications with AVS/Express. Technology White Paper. Advanced Visual Systems, Inc., Waltham, M.A.

Fritz, D. G., R. G. Sargent, and T. Daum. 1995. An overview of HI_MASS (Hierarchical Modular and Simulation System). In *Proceedings of the 1995 Winter Simulation Conference*, ed. C. Alexopoulos, K. Kang, W. R. Lilegdon, and D. Goldsman, 1356-1363. IEEE, Piscataway, N.J.

Henriksen, J. O. 1996. The power and performance of Proof animations. In *Proceedings of the 1996 Winter Simulation Conference*, ed. J. M. Charnes, D. M. Morrice, D. T. Brunner, and J. J. Swain, 460-467. IEEE, Piscataway, N.J.

Law, A.M. and M. G. McComas. 1996. ExpertFit: Total support for simulation input modeling. In *Proceedings of the 1996 Winter Simulation Conference*, ed. J. M. Charnes, D. M. Morrice, D. T. Brunner, and J. J. Swain, 588-593. IEEE, Piscataway, N.J.

Pritsker, A. A. B., J. J. O'Reilly, and D. K. LaVal. 1997. *Simulation with Visual SLAM and AweSim***.** New York: Halsted Press.

Standridge, C. R. 1985. Performing simulation projects with the extended simulation system (TESS). *Simulation* 45: 283-291.

Standridge, C. R. and A. A. B. Pritsker. 1987. *TESS: The Extended Simulation Support System.* New York: Halsted Press.

Standridge, C. R. and M. A. Centeno. 1994. Concepts for modular simulation environments. In *Proceedings of the 1994 Winter Simulation Conference*, ed. J. D. Tew, S. Manivannan, D. A. Sadowski, and A. F. Seila, 657-663. IEEE, Piscataway, N.J.

Standridge, C. R., J. F. Kelly, T. Kelley, and J. Walther. 1996. Progress in modular simulation environments. In *Proceedings of the 1996 Winter Simulation Conference*, ed. J. M. Charnes, D. M. Morrice, D. T. Brunner, and J. J. Swain, 714-720. IEEE, Piscataway, N.J.

AUTHOR BIOGRAPHY

CHARLES R. STANDRIDGE is an associate professor in the Padnos School of Engineering at Grand Valley State University. He led the development of the Simulation Data Language (SDL) and of The Extended Simulation Support System (TESS) for Pritsker Corporation. His current interests lie in the development of teaching approaches for simulation including a modular simulation environment based teaching laboratory. He is working with industry on simulation-based approaches to problems in inventory management and supply chain operations.