Proceedings of the 1994 Winter Simulation Conference
ed. J. D. Tew, S. Manivannan, D. A. Sadowski, and A. F. Seila

CONCEPTS FOR MODULAR SIMULATION ENVIRONMENTS

Charles R. Standridge

Department of Industrial Engineering
Florida A&M University / Florida State University
Joint College of Engineering
P.O. Box 2175
Tallahassee, Florida 32308

Martha A. Centeno

Industrial and Systems Engineering
Florida International University
College of Engineering and Design
University Park
Miami, Florida 33199

ABSTRACT

Ideally, each simulationist would employ a self-selected set of software tools operating within an integrated environment. Some of these tools would perform simulation specific tasks such as model building. Other tools, such as word processors and spreadsheets, are more general purpose but able to perform simulation project tasks such as graphing simulation results, entering simulation input values, and performing statistical analyses. Tailoring of these general purpose tools to simulation activities or particular problem contexts is necessary. These requirements are addressed by a modular simulation environment. A database management system provides for the organization, storage and retrieval of simulation input values, simulation results, and models. Standard information exchange mechanisms retrieve data from the database for use by software tools and store data resulting from the use of these tools. Pre-existing and newly developed tools can be attached or detached as needed from a modular simulation environment to serve different user types or different contexts. Standard user interfaces, such as those provided by a windowing environment, support a common user interface for the tools as well as other possibilities for information exchange between tools.

1 INTRODUCTION

Current simulation environments provide fixed functional capabilities and user interfaces that must serve a wide variety of application contexts. Ideally however, each simulation user would employ a self-selected set of simulation specific software tools as well as general purpose tools that provide the capabilities needed to perform any particular simulation project. Thus, a simulation software environment should be tailored to the specific needs and desires of individual users and to the particular application context currently under study.

This viewpoint is analogous to one facet of modular modeling (Cota and Sargent, 1992; Gordon, et. al, 1991; Standridge, 1986; Zeigler, 1990). Modules addressing a particular application area are developed. A model of a specific system in that area is constructed from the modules. This approach results in a shorter model development time than building the model from scratch using general purpose modeling constructs. Modular modeling helps enable technical staff familiar with the system but not fluent in general purpose simulation modeling to build models and supports the reuse of models or fragments of models.

Furthermore, the philosophy of computer operating environments such as Microsoft Windows for personal computers and X-Windows for Unix-based work station computers should be a significant influence on simulation environments. This philosophy encourages the development of a variety of simulation specific software tools such as graphical model builders as well as the use of existing general purpose software tools such as spreadsheets to meet various simulation project requirements. The windowing systems provide the mechanisms for tool integration. Windows and X-Windows provide a user interface standard for software tools as well as standard mechanisms for sharing information between tools.

Some benefits of simulation environments having these characteristics are as follows:

1. High flexibility for end user selection of simulation software. This supports simultaneous use of simulation software from multiple sources as well as locally developed tools.
2. Use of general purpose software with which the end user is already familiar from non-simulation applications such as spreadsheets or general purpose statistical packages like SAS.

3. Inclusion of tools such as word processors and presentation graphics generators that have not traditionally been a part of simulation environments.
4. Use of standard windows techniques for sharing information between tools.
5. Management and selective use of simulation inputs, results, and models.
6. Definition of a standard for simulation environment structure and user interface. Because of its flexibility, end users may generally adopt such a standard and tool builders may find developing software compatible with its requirements helpful.
7. Compatibility with the existing environment for newly developed general purpose tools or simulation specific software.

This paper describes the concepts for modular simulation environments that meet the above requirements and provide the specified benefits. The different types of users of a simulation environment are defined. The overall architecture of a modular simulation environment is presented. The organization of data in the environment database is described. The tailoring of software tools within the environment is discussed. One possible modular simulation environment is presented.

2 USER TYPES AND THEIR REQUIREMENTS

Multiple types of users, each with distinct skills and project objectives, will use modular simulation environments (Standridge and Centeno, 1991). Understanding the skill and functional requirements of each user type allows the modular simulation environment to be tailored for optimal use. The user types represent typical roles in a modeling and simulation project. For any particular project, the same individual may take on multiple roles or many individuals may participate in the same role. We have identified five types of users: decision makers, model users, model builders, system builders, and tool builders.

2.1 Decision Makers

A modeling and simulation project provides critical information, recommendations, and unique perspectives, unavailable from any other source, for decision making in a timely fashion. Often meeting this goal requires the development and assessment of multiple alternatives.

A decision maker chooses among the multiple alternatives. To support the decision maker, the modular simulation environment should support the transfer of data and other information generated in the modeling and analysis phases into forms suitable for management evaluation, and report and presentation generation. Thus, a modular simulation environment should include a presentation graphics package and word processor as well as more common simulation animation and graph generation capabilities.

2.2 Model Users

The model user supports the decision maker through the assessment of system alternatives using an existing model. This user describes alternative scenarios through model input parameter values and examines results to assess and compare alternatives. Thus, this user type does not need model construction skills. The model user has knowledge of the system under study.

The model user needs a usable interface for entering model parameter values and examining simulation results. Spreadsheets are one possibility since they are commonly used, provide functionality for statistical computations and generating graphs, and are tailorable to specific contexts. Since there are many commercial spreadsheets, the ability to support whichever spreadsheet is familiar to the model user is critical.

2.3 Model Builders

The model builder enables the tasks of the model user and information transfer to the decision maker. The model builder constructs models of the system of interest to the model user. In addition, the model builder defines the input parameters whose values the model user must specify as well as providing the interface for entering these values. For example, the model builder could write a spreadsheet macro that prompts the model user for the required input values. Furthermore, the model user defines the performance measures estimated by the simulation and provides methods by which the model user examines them. For example, the model builder could specify the collection of times series of performance measure values. The simulation environment could then transfer them to a general purpose statistical analysis package where previously defined sets of instructions (macros) could perform a variety of statistical analyses and create graphical displays of time series and summary statistics under the direction of the model user.

2.4 System Builders

The system builder constructs a specific simulation environment by specifying the tools that operate within it. The system builder uses modular simulation environment capabilities to attach the selected tools. It

is possible to produce simulation environments with different tools and capabilities specifically addressing model builder and model user needs. Furthermore, the system builder can tailor tools as appropriate for use by the model builder. Tailoring includes writing spreadsheet macros, building module libraries for model construction, and defining the class of problems the simulation environment can address.

2.5 Tool Builders

Tool builders gather off-the-shelve software tools, both generic tools such as spreadsheets, presentation graphics programs, word processors, and statistical analysis programs and simulation specific tools such as model builders and simulation animators. Tool builders provide fundamental modular simulation environment capabilities such as those that allow tools to be attached and detached as required.

3 MODULAR SIMULATION ENVIRONMENT DEFINITION

Our current working definition of a modular simulation environment is as follows:

A modular simulation environment is characterized by the ability to attach and detach software tools from the environment as needed using a well defined protocol as well as to tailor tools within the environment to each specific application context.

This definition implies that the flow of project related information, including input parameter values, time series of observations of simulation performance measures, summaries statistics, and models, must be controlled by the environment. Standard interface mechanisms between the environment and software tools that provide for the bi-directional flow of information must be defined and implemented. Tools should employ a user interface standard such as that provided by Microsoft Windows. Tools should support tailoring of their user interfaces to particular application contexts.

4 MODULAR SIMULATION ENVIRONMENT ARCHITECTURE

Figure 1 gives the architecture of a modular simulation environment. Note that the architecture is similar to that of existing simulation environments (Balci and Nance, 1987; Balci and Nance, 1992; Standridge, 1985; Standridge and Pritsker, 1987). A database manager organizes, controls, stores, and retrieves all information in the environment. Each member of a collection of software tools uses existing information in the database and adds the results of its own operations to the database. Thus, the database manager provides the mechanism by which information is shared among the software tools (Centeno and Standridge, 1992; Centeno and Standridge, 1993; Cornelio and Navathe, 1993). The software tools use a standard, common user interface.

The capabilities shown in italics are significant modifications of or new additions to typical existing simulation environments required to create a modular simulation environment. The simulation data controller provides a standard organization for simulation data as well as for storage and retrieval. Data organization, storage and retrieval capabilities are mapped into the more general capabilities of a database manager. The standard data interface uses the capabilities of the simulation data controller to provide standard data interchange capabilities between the database management system and the software tools of the simulation environment. These standard interface capabilities must accommodate the mechanism used by pre-existing, independent software package to input and output data. The packages cannot be expected to adapt to the database management requirements of the environment. The package controller provides information to the standard data interface to enable a new software tool, such as the trace processor described by Standridge and Tsai (1992), to receive information from the database and to store information in the database.

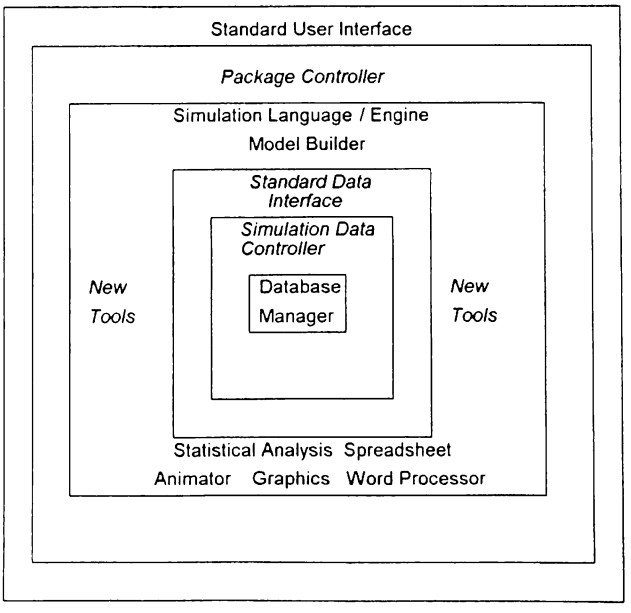

Figure 1: Modular Simulation Environment Architecture

5 MODULAR SIMULATION ENVIRONMENT DATABASE ORGANIZATION

The simulation environment database can be viewed as containing three types of information:

1. *General data* including model input parameter values, data from which the model input parameter values are estimated, or other data of general interest in a simulation project.
2. *Simulation results* including time series of observations of performance measures and their statistical summaries.
3. *Models and experimental controls.*

Each general data value is labeled with the following:

1. *Scenario:* The system alternative described by the data value.
2. *Table:* The collection of variable values to which the value belongs.
3. *Variable:* The unique identifying name of the value.
4. *Key value:* The variable whose value uniquely identifies the record to which the general data value belongs, if any.

Standridge, LaVal, and Reust (1992) present a case study that uses this labeling scheme.

A record is a collection of variable values and corresponds to row within the table. Consider the example of a table of operation times for two part types at three stations shown in Figure 2. The times reflect current system operations as indicated by the scenario name Current_Case. A second table with different values and a different scenario name would be used to described a proposed or new system scenario such as one evaluating faster machines. The variables are: Part_ID, Station_1, Station_2, and Station_3. The key values are those in the Part_ID column.

Table: Operation_Times
Scenario: Current_Case

Part_ID	Station_1	Station_2	Station_3
P1	3.5	4.5	4.2
P2	3.8	4.2	4.6

Figure 2: Example Simulation Input Data

Multiple tables, each containing the values of different variables, can be used to describe a single system scenario for simulation as shown in Figure 3. Alternative_1 is described by the Current_Case values for Operation_Times and Production_Demand and a New_Possibility for Routes.

System_Scenario: Alternative_1

Table	Scenario
Operation_Times	Current_Case
Routes	New_Possibility
Production_Demand	Current_Case

Figure 3: Example Definition of System Scenario

Observations of simulation performance measures are labeled in a similar way (Standridge, 1988).

1. *Scenario:* The system alternative to evaluate.
2. *Table:* The collection of variable values to which the value belongs.
3. *Variable:* The unique identifying name of the value.
4. *Time:* Simulation time at which the value was observed.
5. *Replicate:* ID number of the simulation replicate.

Consider the observations from the first simulation replicate of the length of two queues organized together into the table named Queues as shown in Figure 4. The scenario is as the system currently operates: Current_Case. A row of the table corresponds to a point in time when at least one of the queue lengths changed. Since queue lengths are time persistent variables, each length is for an interval beginning at the value in the Time column of the same row and ending at the value in the Time column of the following row.

Table: Queues
Scenario: Current_Case
Replicate: 1

Time	Q_1	Q_2
0.0	0	0
1.1	1	0
1.9	2	0
2.2	1	1
2.4	2	1
.	.	.
.	.	.
.	.	.

Figure 4: Example Table of Observations of Queue Lengths

Figure 5 gives the queue length values for a second system scenario.

Table: Queues
Scenario: Alternative_1
Replicate: 1

Time	Q_1	Q_2
0.0	0	0
0.9	1	0
1.7	0	1
2.6	1	1
.	.	.
.	.	.
.	.	.

Figure 5: Example Table of Observations of Queue Lengths - Second Scenario

Suppose it is desired to output a table containing the length of Q_1 from each of the two scenarios. This helps in comparing the length of Q_1 across the scenarios. Remembering that each row represents the queue length values over an interval of time, a join can be performed on the tables shown in Figures 4 and 5 to produce the table shown in Figure 6. Note that there is one row in the table in Figure 6 for each row in the tables in Figures 4 and 5.

Table: Queues
Replicate: 1

Time	Q_1: Current_Case	Q_1: Alternative_1
0.0	0	0
0.9	0	1
1.1	1	1
1.7	1	0
1.9	2	0
2.2	1	0
2.4	2	0
2.6	2	1
.	.	.
.	.	.
.	.	.

Figure 6: Example Table of Observations of One Queue Length from Two System Scenarios

Models can be viewed as sequence of records (Centeno and Standridge, 1991). Each record contains a variable number of fields. Each model can be labeled with its name. A model could be a module and a system model composed of many modules. Hierarchical relationships between modules could be included. A more detailed discussion of the use of database management systems for managing models is given by Hitz, Werthner, and Oren (1993), Rovira, Spooner, and Haddock (1993), and Lenard (1993).

6 TAILORING MODULAR SIMULATION ENVIRONMENT TOOLS

Many general purpose software tools that operate in windows environments may be tailored for use in a particular context such as a modeling and simulation study of a particular system. Tailoring involves developing a user interface particularly for the problem at hand or at least specific to simulation tasks. This requires specifying menu bars, prompt for values, or ways of transparently performing computations or generating graphics.

Consider using a spreadsheet to generate a graph of the average number in a queue across four replicates. The queue length for each replicate is stored in a different table. The simulation data controller would generate a table with the columns shown in Figure 7. The standard data interface would generate a file with values in this table in a format readable by the spreadsheet. A spreadsheet macro would load the file, compute the average, and generate the graph.

Table: Queues
Scenario: Current_Case

Time	Q_1: Rep 1	Q_1: Rep 2	Q_1: Rep 3	Q_1: Rep 4

Figure 7: Example Structure of Table of Observations of One Queue Length from Five Replicates

7 EXAMPLE SIMULATION ENVIRONMENT

Consider the architecture of a modular simulation environment consisting of a graphical model builder, experiment editor, a spreadsheet, an animator, a statistical analysis package, and a word processor shown in Figure 8. In this environment, the statistical analysis package, word processor, and spreadsheet are general purpose software and could be tailored by the tool builder for general simulation tasks, the system builder for the class of systems with which the environment deals, and the model builder for the specific system under study. Each of these general purpose tools can receive text files from the standard data interface and return text files to the standard data interface. For

example, a text file containing the information shown in Figure 7 could be transmitted to the statistical analysis package for computation of averages and confidence intervals that could be returned to the standard data interface for storage in the database. Later the averages and confidence intervals could be transmitted to the spreadsheet for graphical display. The graphs could be cut and paste into a word processor document using standard windows procedures.

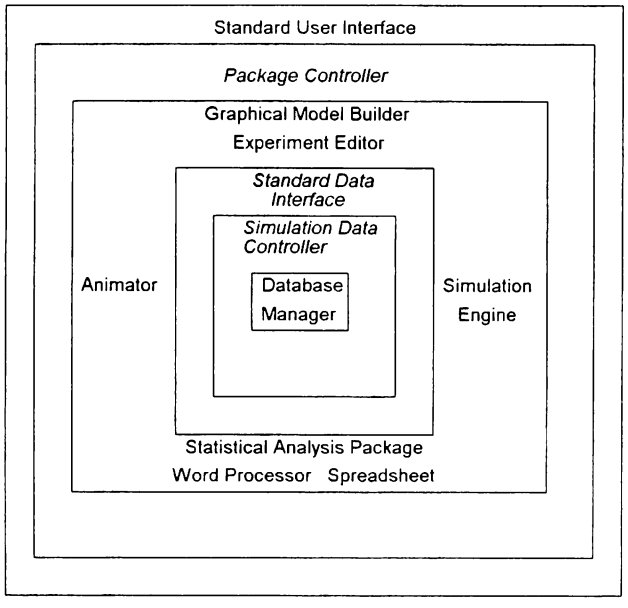

Figure 8: Example Modular Simulation Environment

The graphical model builder and experimental editor produce text files as well. Lines consists of a varying number of fields. The standard data interface "knows" how to merge the model and experiment for input to the simulation engine as well as how to extract observations of performance measures from the simulation engine. In the same way, the standard data interface provides simulation results to the animator in a format that the animator can understand. The animator would not necessarily come from the same software supplier as the graphical model builder, experiment editor and simulation engine.

The users of the environment would work with the tools of the environment in the normal way, except for whatever tailoring was performed. The standard user interface would provide a common way of accessing the tools and the standard data interface would control the flow of information in the environment.

8 SUMMARY

This paper defines and illustrates the concepts for modular simulation environments. A heterogeneous set of software tools, some simulation specific and some general purpose, provide the functionally of such an environment. Standard data interfaces are defined and implemented so that the tools may retrieve data from and store data in a common database through the existing data exchange mechanisms employed by each tool. Furthermore, individual tools can be tailored to problem contexts or at least general purpose tools tailored for simulation tasks. Modular simulation environment capabilities allow particular simulation environments to reflect the requirements of individual users and the contexts of individual problems.

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.

Centeno, M. A. and C. R. Standridge. 1991. Modeling manufacturing systems: an information-based approach. In *Proceedings of the 24th Annual Simulation Symposium*, ed. A. H. Rutan, 230-239. IEEE Computer Society Press, Los Alamitos, California.

Centeno, M. A. and C. R. Standridge. 1992. Databases and artificial intelligence: enabling technologies for simulation modeling. In *Proceedings of the 1992 Winter Simulation Conference*, ed. J. J. Swain, D. Goldsman, R. C. Crain, and J. R. Wilson, 181-189. IEEE, Piscataway, New Jersey.

Centeno, M. A. and C. R. Standridge. 1993. Databases: designing and developing integrated simulation modeling environments. In *Proceedings of the 1993 Winter Simulation Conference*, ed. G. W. Evans, M. Mollaghasemi, E.C. Russell, and W. E. Biles, 526-534. IEEE, Piscataway, New Jersey.

Cornelio, A. and S. B. Kavathe. 1993. Applying active database models for simulation. In *Proceedings of the 1993 Winter Simulation Conference*, ed. G. W. Evans, M. Mollaghasemi, E.C. Russell, and W. E. Biles, 535-5. IEEE, Piscataway, New Jersey.

Cota, B. A. and R. G. Sargent. 1992. A modification of the process interaction world view. *ACM*

Transactions on Modeling and Computer Simulation 2: 109-129.

Gordon, K. J., R. F. Gordon, J. F. Kurose, and E. A. MacNair. 1991. An extensible visual environment for construction and analysis of hierarchically-structured models of resource contention systems. *Managment Science* 37: 714-732.

Hitz, M., H. Werthner, and T. I. Oren. 1993. Employing databases for large scale reuse of simulation models. In *Proceedings of the 1993 Winter Simulation Conference*, ed. G. W. Evans, M. Mollaghasemi, E.C. Russell, and W. E. Biles, 544-551. IEEE, Piscataway, New Jersey.

Lenard, M. L. 1993. A Prototype Implementation of a Model Management System for Discrete-Event Simulation Models. In *Proceedings of the 1993 Winter Simulation Conference*, ed. G. W. Evans, M. Mollaghasemi, E.C. Russell, and W. E. Biles, 560-568. IEEE, Piscataway, New Jersey.

Rovira, M., D. L. Spooner, and J. Haddock. 1993. The Concepts of Views in Simulation. In *Proceedings of the 1993 Winter Simulation Conference*, ed. G. W. Evans, M. Mollaghasemi, E.C. Russell, and W. E. Biles, 552-559. IEEE, Piscataway, New Jersey.

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

Standridge, C. R. 1986. An approach to model composition from existing modules. in *Modeling and Simulation in the Artificial Intelligence Era*, ed. M. S. Elzas, T. I. Oren and B. P. Zeigler. North-Holland.

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

Standridge, C. R. 1988. Databases for simulation. In *Encyclopedia of Systems and Control*, ed. M. G. Singh, 893-894. Pergamon Press.

Standridge, C. R. and M. A. Centeno. 1991. Concepts for production modeling systems based on multiple user types. In *Proceedings of the 1991 Winter Simulation Conference*, ed. B.L. Nelson, W.D. Kelton, and G.M. Clark, 428-434. IEEE, Piscataway, New Jersey.

Standridge, C. R., D. K. LaVal and J. Reust. 1992. Model input management: a case study. *Simulation* 58: 199 - 208.

Standridge, C. R. and J. Tsai. 1992. A method for discrete event trace processing. *Simulation* 59: 384-391.

Zeigler, B. P. 1990. *Object-Oriented Simulation with Hierarchical, Modular Models: Intelligent Agents and Endomorphic Systems*, Academic Press, San Diego, California.

CHARLES R. STANDRIDGE is an associate professor in the Department of Industrial Engineering at the Florida A&M University / Florida State University Joint College of Engineering. He led the development of the Simulation Data Language (SDL) and of The Extended Simulation Support System (TESS) for Pritsker Corporation. His current research interests are in the development of modular simulation environments, the integration of simulation into manufacturing design processes, and the analysis of health care delivery systems.

MARTHA A. CENTENO is an assistant professor in the Department of Industrial and Systems Engineering at the Florida International University. She received a B.S. in Chemical Engineering from ITESO (Mexico) in 1981, a M.S. in Industrial Engineering from Louisiana State University in 1985, and a Ph.D. in Industrial Engineering from Texas A&M University in 1990. Her current research interests are in the areas of intelligent environments for systems analysis. Dr. Centeno is a member of ASA, Alpha Pi Mu, IIE, ORSA, TIMS, and SCS.