Proceedings of the 1993 Winter Simulation Conference
G.W. Evans, M. Mollaghasemi, E.C. Russell, W.E. Biles (eds.)

DATABASES: DESIGNING AND DEVELOPING INTEGRATED SIMULATION MODELING ENVIRONMENTS

Martha A. Centeno, Ph.D.
Charles R. Standridge, Ph.D.
Department of Industrial Engineering
FAMU/FSU College of Engineering
Florida A&M University / Florida State University
P.O. Box 2175
Tallahassee, Florida 32316

ABSTRACT

Database management systems (DBMS) provide robust information storage, retrieval, and indexing functions needed by a simulation modeling environment (SME). Such capabilities standardize information handling requirements stipulated by the principles of software engineering. Furthermore, database management technology helps to realize the concept of language neutrality in SMEs. Neutrality with respect to the simulation language has been long sought by simulation researchers because the ideal SME tool should allow inputting the description of *systems* from which *models* can be defined. Furthermore, the simulation modeling tool should possess an interface that unites the various analysis and definition tools needed for the simulation modeling process.

1 INTRODUCTION

Simulation modeling requires large amounts of information (about the system being modeled) and broad technical knowledge to transform it into meaningful parameters of the system. It also generates large amounts of information (about the performance of the system under study) once the outputs have been analyzed and interpreted. Consequently, it has always been desirable to provide adequate data and knowledge management support in a SME. A database management system (DBMS) provides for the storage and retrieval of information in an orderly and coherent fashion, regardless of volume and kind of data, with user instructions expressed in a query language such as SQL (Structured Query Language). Thus, a DBMS embodies concepts and capabilities which enable SME designers to design flexible and cohesive components including the following:

a) *An interface* which facilitates the interaction between users an the *information entry* component of the SME. It should be sensitive to the role of the user in the simulation modeling process (SMP), and it should enable integration of the various tools needed in the SMP.
b) *An information entry and editing component* that combines interactive graphical and textual approaches to populating the SME database with the description of systems, models scripts, and so forth. Combining icons and text seems to be appropriate for a SME based on the findings of Kacmar (1989), Egido and Patterson (1988), and Benbasat and Todd (1993).
c) *An information processing* component that converts information previously stored in the database into appropriate inputs for models, stores simulation results for future processing, and converts simulation results into meaningful information.
d) *A model extraction component* which enables the retrieval of systems' descriptions and converts it into specific models, or it retrieves models and experimental controls, and transforms them into the format needed by the *simulation engine*.
e) *A simulation engine* which executes the simulation model. The simulation engine is either specifically developed for the SME or is an existing simulation language incorporated into the SME.

For SME development, emphasis must be given 1) to the fact that components of a SME, except for the model and experimental control editors and model extraction, do not need any knowledge of the operating characteristics of the simulation engine, 2)

each of the components ((a) to (e)) of the SME may be composed of one or more sub-components to make the environment modular, flexible and extensible and 3) to the fact that the DBMS may have not been designed for simulation; thus, close attention should be paid to its interactive response time, its programmatic capabilities, and its capability to be submissive.

In this paper, we discuss the ways in which database technology has aided in designing and implementing commercial and research prototype SME, especially with regards to the components listed above. Section 2 describes the major characteristics of an integrated SME. Section 3 describes the roles databases may play in a SME.

2 INTEGRATED SME

Simulation modeling environments is a concept that has matured over the last decade. During this time, users and researchers of simulation offered definitions and guidelines about what a *comprehensive* simulation tool should be All these guidelines and definitions have converged to what we nowadays refer to as a SME. It is worth noting that this consensus has been greatly influence by advances in artificial intelligence and database technologies.

An integrated Simulation Modeling Environment (SME) is a software tool that provides computer support for the various activities in the simulation modeling process (SMP), which has been described by many as a process that includes problem formulation, model abstraction, data collection, model building, model verification and validation, analysis of outputs, documentation, and implementation (Banks and Carson (1990), Law and Kelton (1991), Pegden, Shannon and Sadowski (1990)). The SME, therefore, needs to *integrate* a set of tools designed to support a specific activity in the SMP. Among these tools are statistical tools (to prepare inputs, to design experiments, and to analyze outputs), model building tools (expert systems and simulation languages), documentation and report writing tools (word processors), and knowledge based systems (to assist in the less structured activities such as problem definition and model abstraction). Integration of these tools must be both vertical as well as horizontal. Vertical integration makes these tools accessible to the user through a common interface while horizontal integration enables communication between the tools, so as to be consistent with the iterative nature of the SMP.

Real world simulation projects involve two or more persons working as a team. Each member of the team may have different skills to contribute to the entire project; therefore, a SME must adequately support each member's technical needs as well as his/her tool interaction needs. The technical needs are supported by the various tools integrated in the SME, whereas the tool interaction needs are supported by effective user interfaces that shelter team members from detailed operational characteristics of the tools. An effective user interface encourages its users to focus their energies on the substance of their work rather that on extraneous requirements of the tool being used. Furthermore, to the extent a user interface successfully models the system's and user's domain, the process of interacting with the computer begins to assume the character of manipulating domain objects directly. This leads the user to think about the problem being solved rather than the way it is being solved.

From the conceptual point of view, a SME must detach the *simulation modeling* activity from the actual simulation language or simulation executor. *Language independence* or *language neutrality* requires that the simulation language utilized in the SME be *just one* of many components needed to conduct sound and valid simulation projects. Realizing *language neutrality* would enhance the modeling aspect of the SMP; thus, making simulation more accessible to various types of users. However, as pointed out by Balci (1986), it was not until very recently that simulation tool developers were forced to build them around a specific simulation language.

Simulation projects are initiated to simulate a system for the purpose of understanding its behavior (if the system already exists) or of establishing the impacts of it (if the system is at the design stage). A *system*, for the purposes of designing a SME, should therefore be viewed as a group of objects that are linked together by a set of relationships, i.e. a system must be treated as an object composed of other "smaller" objects. This view of *systems* minimizes data entry redundancy by allowing reusability of previously described system components, and it enables consistent classification and labeling of each node in the hierarchy representing the attributes of the system.

In summary, implementing an integrated SME would realize at least the following benefits:

1. Reusability of components previously described since they can be used for more than one model without the need of re-entering their basic information.
2. Realization of language independence as systems (and models about them) can be described in terms

of *objects* which are not necessarily linked directly to the underlying execution mechanism.

3. Simulation of a *system* at various levels of aggregation. Since a system is defined as a set of hierarchical connected objects, it can be modeled as a black box (at the higher level of abstraction), or it can be opened up to simulate it at an expanded level of detail.
4. Support for various types of users as it is needed in simulation studies.
5. Support for the various phases in the SMP.
6. Focus on the modeling aspects of the SMP as the model builder and model user are relieved from having to know low level details about the simulation language and other tools in the SME.

3 THE ROLE OF DATABASES IN A SME

Over the years, database technology has evolved to a point in which commercial DBMS (relational models in particular) can be either the *master* (active) component of a software environment, or a *subordinate* (passive) component of it. For a SME, a DBMS must be mostly *passive*, i.e. it should submit itself to the control of other software tools (such as high level programming languages), so as to allow simulation specific packages to retrieve and store data from/into databases. However, the DBMS may be *active* for typical data management activities, so as to exploit all of its potential.

As with most evolving technologies, the role of a DBMS when used for simulation purposes has changed over the years. Three roles have been given to a DBMS in this context: 1) as a *programming language* to develop interfaces as well as simulation functions (*master* role), 2) as a *partner* that communicates with other tools in batch via the operating system, and 3) as a *submissive partner* that communicates directly with other tools via an embedded query language

During the early 1980's, researchers were forced to treat the DBMS as the *master* because these DBMS were closed, isolated packages that allowed no foreign tools to penetrate their databases. These packages often came equipped with a limited programming language that could do *some number crunching* on user inputs or on the database contents; however, these languages were in general interpreted languages which made them very slow to process the massive quantities of data in a simulation run.

Enhancements to DBMS and to operating systems made communication among heterogeneous tools, *via text files*, a common activity; in this way, the DBMS became a *partner* working independently, yet informing (and receiving information from) the simulation tool. Although simulation related performance was not being impacted by the performance of the DBMS, the overall simulation project was still being penalized with the overhead of data import and export activities. Attempts to overcome this hurdle lead many researchers to develop their own DBMS for their SME. Ketcham (1986, 1989), for instance, developed a hierarchical DBMS that enables IBIS (his SME) to store *model* components as well as simulation functions. He then configured the simulation engine in such a way that it would look into a *procedure field*, retrieve the name of the simulation procedure to execute, and then match it against IBIS executable library.

TESS [Standridge (1981b), Standridge, Vaughan, and Sale (1985)] is another example of a customized database. TESS was one of the first commercial simulation environments. It integrates simulation, data management, and graphics capabilities, providing a common interface for SLAM II (Pritsker, 1986), SDL (Standridge, 1981a), MAP/1 (Norman, 1991), and GPSS/H (Schriber, 1990). TESS organizes simulation outputs in a customized relational database similar to that used in SIMDABS (Standridge and Pritsker, 1978). Some of the schemes in the database are pre-defined, whereas others are defined within the experimental control information.

To overcome some of these problems, Balci and Nance (1987) and Centeno (1990) proposed SMEs based on commercial relational DBMS. SMDE (Balci and Nance) utilizes INGRES as the relational DBMS, whereas ISME (Centeno) uses ORACLE as its RDBMS. Both INGRES and ORACLE offer full implementation of the relational data model, portability across multiple platforms, and SQL as the query language. SQL became the industry standard during the 1980's. It attained a level of maturity in which semantic information may be incorporated through special constructs such as *associations*, *properties*, *entities* [Codd (1979), Gardarin and Valduriez (1989), and Stonebraker, Anton and Hanson (1987)], *classes* and *hierarchies* [Goldberg and Robson (1983), Hammer and McLeod (1980), Katz (1985), Smith and Smith (1977)].

The potential of the relational data model for the SME underlying database stems from the work done by Smith and Smith (1977) which clearly depicts a relational model that can be enhanced to hold various types of entities, including hierarchy-like entities. Through the concept of *aggregation* (a concept in which a relationship between objects is in itself *another* object at a higher level in the taxonomy) and

of *generalization* (a concept in which a set of similar objects are regarded as belonging to the same generic object class), it is possible to represent the kind of hierarchical systems that are found in simulation studies.

These efforts have led to a better understanding of the inner workings of available database technology, and how it can be intertwined with simulation. We discuss next how databases can be used for some of the SME components.

3.1 Interfaces for a SME

An effective user interface encourages its users to focus their energies on the substance of their work rather that on extraneous requirements of the computer software being used. Furthermore, to the extent that a user interface successfully models the system's and user's domain, the process of interacting with the computer begins to assume the character of manipulating domain objects directly. This allows the user think about the problem being solved rather than the way it is being solved.

A database may be used to store the characteristics of the interface, such as display mode, headings, menus, menu options, and so forth; thus, supporting the SME designer to tailor the SME to a particular application.

Relational DBMS, such as ORACLE or INGRES, offer the capability of embedding SQL statements in a high level programming language such as C or FORTRAN. This programmatic capability has enabled SME designers to design and implement *user-sensitive* and *extensible* interfaces for simulation modeling.

ISME (Centeno, 1990), for instance, uses an object based approach to the interface. Screen definitions and available options are stored in a set of tables whose schema holds information. This information reveals the coordinates of the upper left corner of the window screen to be displayed, as well as the height and width of it, so as to properly positioned it. Other information pertains to messages that are to be displayed as part of the screen. Options in any menu are retrieved from the SME database, at execution time, along with the name of the corresponding function that executes the actions of the option.

3.2 Information Entry And Editing

A SME database is populated by using graphical editors, textual templates, or sub program invocations from within a model (at execution time - *horizontal integration*) or transparently from a SME tool through the SME interface (*vertical integration*). Current relational and object-oriented DBMS posses the capabilities to create user interfaces that take advantage of text-based icons, "*point and click*" devices, English-like query languages, or natural language processors.

Graphical editors are important to SME as far as the animation component of the simulation model is concerned. Although current RDBMSs, such as ORACLE and INGRES, do not provide full graphical support for all platforms, their versions for windows-based platforms do provide some support. We believe that current trends of RDBMS, in this particular area, will converge to meet the graphical editor provided with simulation tools such as SLAMSYTEM and SIMAN/ARENA. Specific models of a system can be described through a collection of icons, each one uniquely identified by its spatial location and a name. The data structures needed hold these icons, and the operations to manipulate these structures are supplied by the DBMS. Under a strict relational model, hierarchically linked tables are needed to fully describe the attributes of irregular shapes. These attributes include the number of vertices and the coordinates of each point, the type of arc connecting any two vertices, the name of the icon, the initial position to display it, color attributes, and shading.

Textual editors provide for entering and changing model input data, simulation experimental control information, animation scripts, presentation formats and the like, directly from the user. Information of this type can be viewed as a set of records with each record composed of a set of fields (*tuples* in the relation data model). Different record classes have different sets of fields (different tables). Operations supported by textual editors include copying, deleting, and moving as well as changing the field values in existing records. As with a model, the textual information may be sufficiently voluminous to preclude concurrently keeping it all in main memory. A user works with only a small subset of the information at a time. Thus, buffering of textual information is required. Current DBMS provide application building capabilities to develop *user application* with data entry forms that act as the textual editor. These data entry forms are manipulated directly by the DBMS; therefore, data integrity, data buffering, and similar operations are carried transparently to the user.

For example, TESS has two types of textual editors. One type is for information organized into the rows of one database table. Such information includes experimental controls, animation scripts, and input data values. The other type of editor processes

presentation formats organized into one database table row. Keyed retrieval functions are used to access a format object by name and replace the function to update that object when editing is completed. ISME uses a dynamic template approach to gather information, e.g. the actual node at level i+1 in the data collection tree is determined by the actual value entered for the node at level i. The data management functions used to perform with the collected data are executed using SQL. By using embedded SQL, ISME relies on the DBMS to properly handle all integrity constraints defined for the particular schema.

3.3 Information Processing

This component collects and stores simulation results, including traces, time series of values, and summary statistics, for future analysis and presentation. Each class of simulation results is a set of records. Each object in a particular simulation result class may have a unique set of fields comprising its records. Multiple sets of time series and statistics may be saved. Statistical summaries are stored at the end of a simulation run as well as optionally throughout the run.

To support the storage of the results of a simulation run, the SME database must accept information into many distinct tables at one time. This is done using a standard database storage function (e.g. INSERT INTO). Buffering is not an issue since no reference is made during a run to previously collected results. The results, therefore, can be place on secondary storage as necessary. The organization of results on secondary storage, however, is important due to volume being a major issue. One ad hoc method for doing this has been reported by Norman (1991). Within an SME, minimizing secondary storage requirements and using organizations that provide for fast retrieval of information processing operations are necessary. Experience has shown that one good organization for results, in keeping with the overall structure of an SME database, is as follows. Each table of results occupies its own set of physical records; thus, the number of records retrieved during information processing per table is minimized. Each record also has three pieces of overhead: the ID of the previous physical record, the ID of the subsequent physical record, and the offset from the beginning of the record where the next row can be placed.

Statistical analysis is typically performed on one table of time series values, or a selected subset of its rows and variables. The computed statistical quantities are either displayed immediately or stored for later presentation or use within a model. Thus, statistical analyses of results need not be specified or performed within simulations but can be defined and computed on a post-simulation basis. Rows of data values are retrieved, one at a time, using database sequential retrieval functions (e.g. SELECT - FROM). The necessary statistical computations are made, and rows of statistical summaries are rewritten to the SME database. Again, using TESS as an example, two statistical analysis operations are available: fitting a distribution function to data and computing summary statistics. Fitting a distribution function is an interactive, graphically based process where a set of values for one variable are inputted from one database table using a sequential retrieval operation. All potential distribution functions and their parameters that could fit the data are stored in another database table using a sequential storing operation. On the other hand, computing statistical summaries is not an interactive operation. Summary statistics are computed from all values, or a selected subset of all variables, in one database table. Multiple sets, one for each specified subdivision of simulation time, of summary statistics may be computed for each variable. The result of the computations are stored in one database table per variable. Rows of a table correspond to batches, and each batch has a unique ID that serves as its key.

ISME uses a prototype KBS to analyze the raw data to establish the inputs for a particular model. Most of the standard statistical analysis is carried out without the intervention of the human user. In selecting among several feasible hypotheses, however, the prototype KBS presents to the user the various alternatives and requests the final decision from the user. Database technology is being used to allow the user to explore as many alternatives as he decides without having to conduct the entire analysis every time. The statistical analysis is thus viewed as a decision tree about which partial results are kept at various node levels. At any point, the user may backtrack and choose another analysis path.

3.4 Model Extraction Component

Models can be defined about systems previously described and stored in the database. This component has two major functions to fulfill in a SME: 1) convert, with the help of the end user, information about a system into a model of it, and 2) convert the model into a specific format to meet the requirements of the simulation engine. Through the SELECT operation of the DBMS (executing JOINs and/or UNIONs on tables), components of a system can be presented to the modeler for his/her modeling

decision. Once models have been fully defined, they need to be "translated" into the format of the simulation engine. This translation can be made in several ways using database capabilities. The database can be used to "merely" store the final product of the translation process, or it can be used to dynamically define the classes of systems that can be modeled, and the simulation engine to utilize. The latter approach may, in the near future, enable language independence in a SME.

In TESS, the translation of the graphical and intermediate network forms to a model executable by SLAM II and is accomplished externally to the database. Classes of symbols providing definition information, resource blocks for example, are selected and processed first. Connectivity between nodes and branches is checked based on relative graphical location. Accomplishing this includes processing the network in order of the horizontal coordinate location of each symbol. The executable form of the network is stored in the TESS database for later simulation.

In ISME, on the other hand, a builder (editor) is used to define the classes of systems that can be simulated and models of systems in this class. Once a system class and models have been defined, ISMEs model synthesizer produces an equivalent SIMAN (Pegden, Shannon, Sadowski, 1990) model and experiment source file. During the synthesis, the engine makes an extensive use of SQL statements to ensure that the integrity of object relationships is preserved. For instance since ISME captures the definition of a system independently of the models about it, it may be possible that there are no models defined for a requested system at the moment the simulation engine is triggered. Thus, before attempting the synthesis of "nothing", the engine checks that there is at least one model for the given system, using the following SQL statement from within a C program:

```
SELECT   MODLCODE, MODLNAME
INTO     :modcodes, :modnames
FROM     MODELS
WHERE    PROJCODE = :projcod;
```

In general, ISMEs simulation engine begins by retrieving all the "customers" to be processed through the simulated system and their relevant attributes. It then proceeds to retrieve the "resources" in the system. To illustrate, consider a class of manufacturing systems where *customers are parts* and *resources are stations or equipment*. Model retrieval is achieved through a series of SQL statements such as the following:

parts to process:

```
SELECT   PARTCODE, BATCHSIZE, INTERTPARS,
         TERTDIST
INTO     :pcodes,:bsize,:iat_pars,:iat_dist
FROM     PART_MODEL
WHERE    MODLCODE = :modlcod;
```

attributes to observe for each part:

```
SELECT   ATTRIBDESCR, ATTRIBCLASS
INTO     :attr_desc, :attr_class
FROM     PART_ATTRIBUTES
WHERE    PARTCODE = :partcod
   AND   MODLCODE = :modlcod;
```

routes to be followed by parts:

```
SELECT   SEQUENCE, PARTFROM, PARTGOTO,
         TRVLTIME, PROCTIME, PROB
INTO     :seq,:sfrom, :sgoto, :ttime, :ptime,
         :probability
FROM     PART_ROUTES
WHERE    PARTCODE = :partcod
ORDER BY SEQUENCE ASC;
```

equipment:

```
SELECT   EQUIPMENT.EQUINAME,
         STATCOMP.COMPQNTY
INTO     :equiname, :compqnt
FROM     STATION_MODEL,
         STATCOMP_MODEL, STATCOMP,
         EQUIPMENT
WHERE
   STATION_MODEL.MODLCODE=:modlcod AND
   STATION_MODEL.STATCODE = :partgot AND
   STATCOMP.COMPTYPE = 'dual'        AND
   EQUIPMENT.EQUICODE=STATCOMP.COMPCOD;
```

Notice that SQL statements can be conditioned to retrieve information in certain order, or they can be multi-conditioned (or nested) to execute a join operation across several tables in the database. Consequently, a system may be modeled at various levels of expansion.

3.5 The simulation engine

In a SME, neutrality is easily realized and further enhanced by database technology since these generic structures, and instances of them, can be physically stored in a database, and later retrieved and updated. For instance, structures for input/output data objects (and the operations performed on them) have been found not to depend on the simulation language that needs them or outputs them. The neutrality of simulation results is a direct result of the philosophical approach used by many simulation languages: *the next event approach*, which yields results in the form of *traces*, *time series*, *sets of tallied values* as they occur, and *summary statistics*.

In the particular case of TESS (Standridge, 1981), individual events belong to a particular *Event Class*, and each event is identified by its class and its ID within the class. Possible event class schema may vary from one simulation language to another, but this becomes an implementation problem only if the underlying DBMS is custom made. Now consider the logical organization for time series of a model's state variables as proposed in SIMDABS. A column in the schema must hold the simulation clock time, whereas the other columns may correspond to user selected state variables. Each row in the table contains one value for each state variable, as well as the simulation clock, and thus represents simulated changes of state. These tables may well be part of a larger simulation database, with their schemes defined in the experimental control unit of a model run.

Neutrality using descriptive modeling has been proven feasible by Centeno (1990). As an example, consider the system shown in Figure 1 (Pegden, et al.). A description of the various objects in the system is mapped into the relational tables shown in Figures 2 and 3. The modeler uses a model builder/editor to describe the objects in the system which in turn populate these tables. Thus, the modeler needs no knowledge of the model analysis component, e.g. the simulation language. The Model Extraction component transforms the information in these tables into the inputs required by the simulation engine.

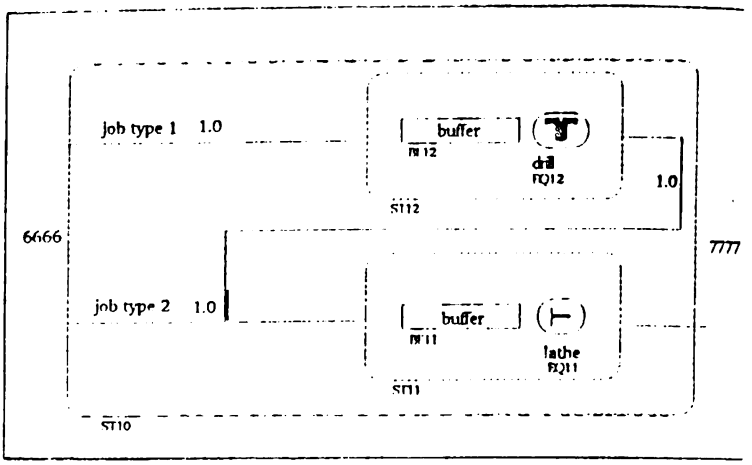

Figure 1: GT Cell Example

jobs

job type	inter arrival time	batch size
1	14.0	5
2	ex(3.0)	8

routes

job type	seq.	from	to	travel time	process time
1	1	6666	st12	0.0	co(3)
1	2	st12	st11	0.0	un(2,3)
1	3	st11	7777	0.0	<null>
2	1	6666	st11	0.0	un(1,2)
2	2	st11	7777	0.0	<null>

Figure 2: Job objects for GT cell example

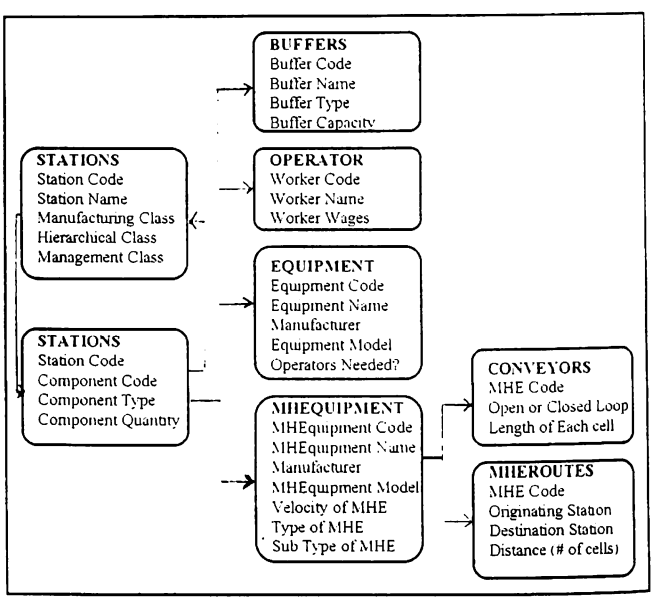

Figure 3: Station objects for GT cell example

4. SUMMARY

Database management concepts and capabilities are an important enabling technology for the design and implementation of simulation environments. This has been demonstrated by the use of these concepts and capabilities in previously developed commercial and prototype SMEs. Database concepts contribute to the development of effective, user-based interfaces that

that help unite the various components of a SME.

Language independence may be realized by using DBMS concepts such as data hiding and standard query languages. Although current commercial RDBMS do not offer all of their potential to the simulation domain, efforts to make them accessible and compatible with other technologies, such as artificial intelligence, are under way.

REFERENCES

Balci, O. (1986), "Requirements for Model Development Environments," *Computers and Operations Research*, 13(1), 53-67.

Balci, O. and R.E. Nance (1987), "Simulation Model Development Environments: A Research Prototype," *Journal of the Operational Research Society*, 38(8), 753-763.

Banks, J. and J.S. Carson (1990), *Discrete-Event System Simulation*, 2nd. Edition, Prentice-Hall, Englewood Cliffs, N.J.

Benbasat, I. and P. Todd (1993), "An Experimental Investigation of Interface Design Alternatives: Icon vs. Text and Direct Manipulation vs. Menus," *International Journal of Man-Machine Studies*, 38, 369-402.

Centeno, M.A. (1990), "Design of an Integrated Simulation Modeling Environment Using a Relational Database Framework," *Doctoral Dissertation*, Industrial Engineering Department, Texas A&M University, College Station, Tx 77843.

Centeno, M.A. and C.R. Standridge (1991), "Modeling Manufacturing Systems: An Information Based Approach," *Proceedings of the 24th Annual Symposium 1991, SCS Eastern Multiconference*, April 1-5, New Orleans, La.

Codd, E.F. (1979), "Extending the Relational Model to Capture More Meaning," *ACM TODS*, 397-434.

Egido, C. and J. Patterson (1988), "Pictures and Category Labels as Navigational Aids for Catalog Browsing," *CHI 1988 Human Factors in Computing Systems Conference*, 127-132.

Frankel, V.L. and O. Balci (1989), "An On-Line Assistance System for the Simulation Model Development Environment," *International Journal of Man-Machine Studies*, 31, 699-716.

Gardarin, G. and P. Valduriez (1989), *Relational Databases and Knowledge Bases*, Addison-Wesley, Reading, Ma.

Goldberg, A. and D. Robson (1983), *Smalltalk: The Language and its Implementations*, Addison-Wesley, Reading, Ma.

Kacmar, C.J. (1989), "An Experimental Comparison of Text and Icon Menu Item Formats," *Working Paper*, Florida State University, Department of Computer Science, Tallahassee, Florida 32316.

Katz, R. (1985), *Information Management for Engineering Design*, Springer-Verlag.

Ketcham, M.G. (1986), "Computer Simulation as a Decision Support Tool," *Doctoral Dissertation*, Industrial Engineering Department, Texas A&M University, College Station, Texas 77843.

Ketcham, M.G., R.E. Shannon, and G.L. Hogg (1989), "Information Structures for Simulation Modeling of Manufacturing Systems," *Simulation*, 52(2), 59-67.

Law, A.M. and W.D. Kelton (1991), *Simulation Modeling and Analysis*, 2nd Edition, McGraw-Hill Book Co., N.Y., N.Y.

Norman, T.A. (1991), "Storing Data from Simulation Runs for Compactness and Quick Retrieval," *Simulation*, 57(2), 107-110.

Pegden, C.D., R.E. Shannon, R.P. Sadowski (1990), *Introduction to Simulation Using SIMAN*, McGraw-Hill Book Co., N.Y., N.Y.

Pritsker, A.A.B. (1986), *Introduction to Simulation and SLAM II*, 3rd. Ed., Halsted Press, N.Y., N.Y.

Schriber, T.J. (1990), Introduction to Simulation Using GPSS/H, John Wiley and Sons, New York.

Shannon, R.E. and M.A. Centeno (1990), "Expert Simulation System Based on a Relational Database," *Proceedings of the 1990 Winter Simulation Conference*, O. Balci, R.P. Sadowski, R.E. Nance (eds), 412-414.

Smith, J.M. and D. Smith (1977), "Database Abstractions: Aggregation and Generalization," *ACM TODS*, 2(2) 105-133.

Standridge, C.R. and A.A.B. Pritsker (1978), "SIMDABS: A Data Base System Tailored for Use in Simulation Studies," *Proceedings of the 1978 Winter Simulation Conference*, H.J. Highland, M.G. Spiegel, and R.E. Shannon (eds), 309-312.

Standridge, C.R. (1981a), "Using the Simulation Data Language (SDL)," *SIMULATION*, 45(2), 73-81.

Standridge, C.R. (1981b), "Performing Simulation Projects with the Extended Simulation System (TESS)," *SIMULATION*, 45(3), 238-242.

Standridge, C.R. and E.A. Matalon (1983), "A Relational Schema and Views for Traces of a Simulation Run," *Modeling and Simulation*, 14(3), 657-662.

Standridge, C.R., D.K. Vaughan and M. Sale (1985), "A Tutorial on TESS: The Extended Simulation System," *Proceedings of the 1985 Winter Simulation Conference*, D. Gantz, G. Blais, S. Solomon (eds), 73-79

Stonebraker, M., J. Anton and E. Hanson (1987), "Extending a Database System with Procedures," *ACM TODS*, 12(3), 350-376.

OTHER RELATED REFERENCES

Balci, O. and et al. (1990), "Model Generation Issues in a Simulation Support Environment," *Proceedings of the 1990 Winter Simulation Conference*, O. Balci, R.P. Sadowski, R.E. Nance (eds), 257-263

Codd, E.F. (1990), *The Relational Model for Database Management - Version 2*, Addison-Wesley, Reading, Ma.

Cox, C.J. (1987) *Object Oriented Programming: An Evolutionary Approach*, Addison-Wesley Publishing Company, Reading, Ma.

Date, C.J. (1985), *An Introduction to Database Systems*, Addison-Wesley, Reading, Ma.

Lamsweerde, A.V., M. Buyse, B. Delcourt, E. Delor, M. Ervier, M.C. Scheyes (1986), "The Kernel of a Generic Software Development Environment," *Communications of the ACM*, 208-217.

Lilegdon, W.R. and J.N. Erlich (1990), "Introduction to SLAMSYSTEM," *Proceedings of the 1990 Winter Simulation Conference*, O. Balci, R.P. Sadowski, and R.E. Nance (eds), 77-79.

Musselman, K.J. (1992), "Conducting a Successful Simulation Project," *Proceedings of the 1992 Winter Simulation Conference*, J.J. Swain, D. Goldsman, R.C. Crain and J.R. Wilson (eds), 115-121.

Nance, R.E. (1977), *The Feasibility of and Methodology for Developing Federal Documentation Standards for Simulation Models*, Final Report to the National Bureau of Standards, Department of Computer Science, VPI&SU, Blacksburg, Va., June.

Reppy, J.H. and E.R. Gansner (1986), "A Foundation for Programming Environments," *Communications of the ACM*, 218-227.

Reese, R. and S. Sheppard (1983), "A Software Development Environment for Simulation Programming," *Proceedings of the 1983 Winter Simulation Conference*, S. Roberts, J. Banks, B. Schmeiser (eds), 419-426.

Wiener, R.S. and L.J. Pinson (1988), *An Introduction to Object-Oriented Programming and C++*, Addison-Wesley Publishing Co., Reading, Ma.

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 system analysis. Dr. Centeno is a member of ASA, Alpha Pi Mu, IIE, ORSA, TIMS, and SCS.

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 interest are in the development of modeling and analysis environments for manufacturing systems, in the use of animation as a modeling and analysis tool, and in the analysis of health care delivery systems.