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

DATABASE ORIENTED MODELING WITH SIMULATION MICROFUNCTIONS

Thomas Wiedemann

Technical University of Berlin
FR 5-5 Franklinstr. 28/29
10587 Berlin, GERMANY

ABSTRACT

This paper presents an approach towards building a flexible modeling and simulation environment with database technologies. The main problem of defining complex systems by component based simulation systems is solved by a set of predefined micro-functions, similar to modern micro-processor architectures. The execution order and additional parameters are also stored in a database.

1 INTRODUCTION

In the last ten years a lot of simulators have been implemented and introduced (Wiedewitsch and Heusmann 1995, Roberts and Dessouky 1998). The large number of different simulation systems is caused by the following problems:

- The modeling of real processes is a very complex task and depends from a big range of details. Frequently the abstract model is so specific, that standard simulation tools can not be applied. In other cases the resulting simulation model would be to slow or to tricky. Thus often a new simulation system is created quite often.
- The inner structure and kernel functions of traditional simulation systems are hidden. Therefore it is almost impossible to integrate simulation systems into other, large information systems (PPS, Work flow systems, etc.). Although the model could be realized, due to the missing kernel interface, the system can not be controlled from outside and a new system is implemented.
- Almost all modern business systems are working on databases. Some simulation systems have good import and export interfaces, but it is not easy to implement a automatic reload of actual database entries during the simulation.
- Optimizations on simulation models require a high performance and remote access to the model parameters. A lot of simulation systems allow a change of input or working parameters during the simulation cycle. Nevertheless there is no system, which allows a change of the model structure and the behavior of the model objects, which would be very important for real optimizations in conjunction with expert- and AI-Systems. In regular, specialized systems are implemented as prototypes and disappear after a short time.

As a result of the discussed problems, the large number of different simulation systems diffuse the knowledge and power of the simulation community and produces a lot of additional work. The effect is well known - simulation in general is a very good tool for decision making, but the number of successful applications is not so big as expected some years ago. In order to change this, we need a new approach to a very flexible, open and high-performance simulation environment.

Additional requirements are a high level of re-usage and different kinds of user interactions - e.g. for web-based, client-server-based or embedded simulation systems. As a conclusion of the discussed problems, we have to solve the following major tasks, which lead to a really modern simulation environment:

- Unified methods and structures for the storage of all model and simulation data,
- Flexible architectures for data-interchange and bi-directional control with other, complex information systems,
- Easy generation and flexible management of the data, which define the behavior of the model objects,
- Flexible and powerful interfaces to related software packages concerning statistical result analysis, presentation, optimization, Data-mining and knowledge reasoning.

A concept for the solution of this tasks and a first software implementation, called SimCoDB, will be presented on the following pages of this paper.

2 MODEL DATA HANDLING AND STORAGE

The proprietary data structures of current simulators cause tremendous problems for all users. The solution for this problem should be the same as it has been for business systems years before - a database. So all simulation data will be stored in a database, which includes :

- model data with all parameters and definitions of the model behavior,
- experiment parameters and optimization methods,
- simulation results and statistical values.

The main problem of this simulation database consists in the high complexity of real processes. The database structure has to allow a high flexibility of relationships between all objects of a simulation model. As it concerns other information systems, there is a serious discussion about the usage of relational or object-oriented databases.

Regarding that relational databases have a strong performance and well defined interfaces, for the current implementation we use a relational database. This should not be considered as a general decision against object-oriented databases.

The Entity-Relationship-Model (ERM) of the simulation database is shown in Figure 1.

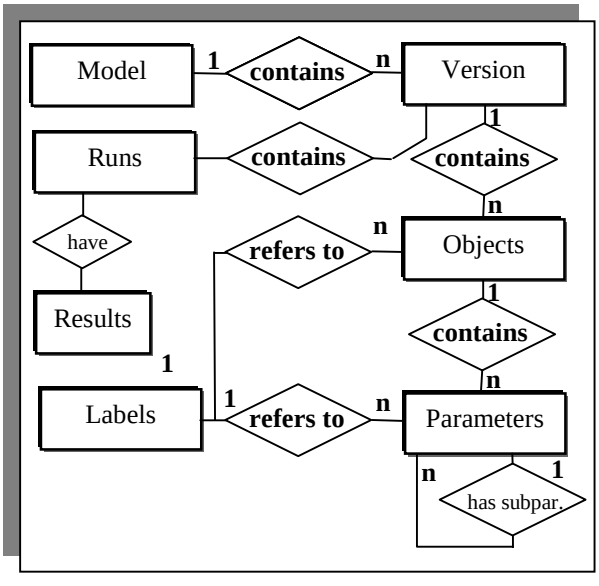

Figure 1 : The ERM-Model of the SimCoDB-Database

The attributes of the database-tables are divided into two groups:

- Attributes for administration and for building the relationships between the different levels of the model objects (objectid, parentobjectid, datatype, subtype, unique database ID).
- A set of attributes for the storage of the object data themselves. Currently we have 3 integer fields (i,j,k), 3 double fields (d,e,f) and two string attributes (s,c).

This data structure is currently used for all model entities. A similar data record structure is used inside the simulator. There is a 1:1 relationship between the database rows and the memory data entities during run-time.

The most important relationship is created by the objectid-parentid relation. Each object can have N child-objects or parameters. Sets of parameters can be stored as a list of sub-parameters. There is no restriction in the depth of the parameter hierarchy. We are limited only by increasing access time, because each additional level requires one more memory-access.

The internal data names like i,j,k are covered by dynamic changing labels in the user interface. Each data object has a application specific reference number for a set of variable names and meanings. A new type of object or parameter is supported by a new set of labels, which provides the user interface with a high flexibility and allows an application specific user-interface without changing the simulation kernel.

3 MODEL MANIPULATION WITH SIMSQL

In result the database technology, all operations during the modeling process can be realized by the help of traditional interfaces like SQL, ODBC or native database-drivers. For the current system we use a MS-Access-Database, but any other relational database would be as good.

Since SQL is well known and well defined in all areas of business systems, the idea of using a SQL-like language for manipulating the model during design and run-time seemed to suggest itself. A related language, called SimSQL was introduced and presented in (Wiedemann 1998) and on the web. By using SimSQL, which in general is very similar to SQL, a client system is able to ask for information by using SELECT-statements or is capable to make changes by applying UPDATE-commands for all model and experiment parameters in the simulation database.

SimSQL may be used also for a set-oriented manipulation of parameters, which does not work in almost all known simulators. So with one single Update statement:

UPDATE SET Capacity=8 WHERE ObjectID >11;

the capacity of the Buffer in all objects, with a number larger than 11, can be changed.

3 MAIN SYSTEM ARCHITECTURE

The requirements for control over the model and the simulation experiment are strongly different in dependence of the whole application structure:

- traditional, stand-alone systems combine user-interface- and run-time very closely,
- Client-server-oriented simulation systems, e.g. Web-based systems, require a strong separation of the user interface and the simulation kernel.

In order to satisfy both requirements, the whole system is divided into a non-visual simulation kernel and a support module for modeling and debugging – the SimExplorer. (see Figure 2 and Figure 3) A additional module can be used for building specific user interfaces. For running a simulation only the kernel is required.

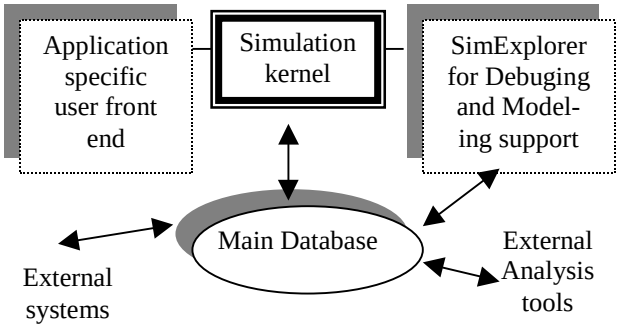

Figure 2 : The System Architecture

Nowadays the development of applications on databases is easier than on proprietary data file structures. In result of ready to use components, e.g. for visualization of database grids and hierarchical tree-views, the existing support module can be replaced or extended by user defined functions. In the current system all modules were first tested with MS Access-forms and then they were realized with Delphi 4.0.

3.1 Batch-Mode and Server-Mode-Architecture

In this architecture only the simulation kernel is used in batch mode. Parameters for simulation, like duration and types of system reports are stored in the database. Afterwards the kernel is started. The kernel mirrors the model from the database into the memory and executes the simulation. All simulation results are stored in a result table of the database. Result analysis is done by the client process, in most cases by standard database reports or graphical representations on the result table.

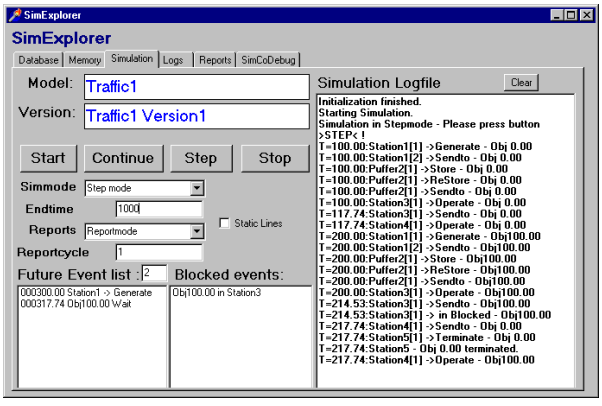

Figure 3 : The SimExplorer

3.2 Application Specific Simulation Frontends

This architecture, which uses all modules of Figure 2, can be used for simulation experiments with direct user interaction and immediate result visualization during the simulation. The application specific fronted is created by standard components of the Delphi toolbox and a range of simulation specific components (see Figure 4). The connection to the simulation kernel is made by a special component, called TSimTimer, which offers a set of methods like "StartSimulation" and "StopSimulation" in order to control the simulation kernel (see Wiedemann 1997 for details). This methods are called by short program statements, when the buttons on the form are pressed. By using interactive control elements like scrollbars, the model parameters can be changed during simulation, which gives a very good impression about the dynamic model behavior. Results are displayed at run-time in simple text-boxes and in Delphi´s very powerful Chart-components.

All important simulation events of the simulation kernel, like generation, entries and exits of new dynamic objects are linked with events of the front end components. If the user inserts own program statements in the event body, the behavior of the model can be changed in a very flexible way.

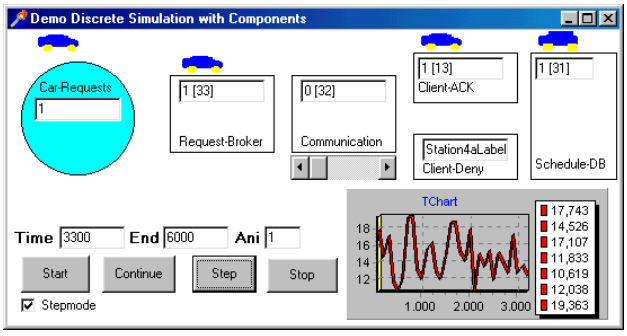

Figure 4 : A Demo User Interface

4 MANAGEMENT OF SYSTEM BEHAVIOR

Defining the whole range of object functions, application specific strategies of product handling, exotic queuing types and individual decision algorithms, is the most difficult task in the implementation of the new system.

In the past there was a strong, antagonistic opposition between flexibility, performance and user friendliness. Thus for example, high performance simulators on universal programming languages were very hard to use by non-experts. Flexible systems with a high-level simulation-specific script language had often a low performance. In result of the analysis of modern simulators like Taylor or Arena and some knowledge about current microprocessor-architectures we favour the following solution: As the micro-code-commands in modern processors do, which define assembler commands by a lot of different combinations, we divide the necessary simulation functions in simulation-micro-functions.

Currently we define the micro-functions:

- *Generate, EnterRequest, Enter*
- *Store, StoreWarehouse, Queue,*
- *Operation, OperationMulti,*
- *Sendto, SendtoMulti, Terminate*
- *AnimatePath, AnimateObject.*

Especially the .Multi-micro-functions are defined for complex algorithms using SimSQL-statements. The current simulation-micro-functions are implemented with Delphi's Object Pascal. New micro-functions are defined by a inherited class-definition and new Object-Pascal-statements.

The behavior of each simulation object is defined by a combination of micro-functions. Some examples for defining complex simulation objects are the following combinations of micro-functions:

Machine objects could be presented by :

- EnterRequest , Enter, Operate, Sendto.

Machine with 2 included buffers:

- EnterRequest , Enter, Queue, Operate,Queue, Sendto

Queue (with infinite capacity):

- Enter, Queue, Sendto

Distributor:

- EnterRequest , Enter, StoreWarehouse, Operate SendtoMulti

The names of the micro-functions and the names of the complex objects are completely at the users disposal. Thus experienced users of GPSS or other traditional languages could align the names to former names - like Enter/Leave; Queue/Depart or Advance/ Terminate. Perhaps, there is a chance to convert older GPSS programs to this new database oriented system.

The data structure of the micro-function is similar to all other model objects. In the current system the micro-functions and prototype definition are managed by the same tree view (see Figure 5) , which is used for all model objects.

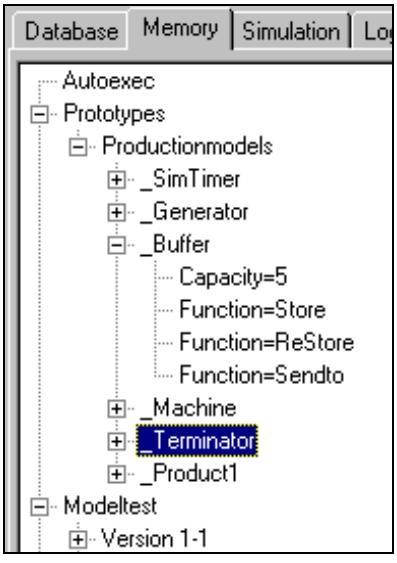

Figure 5: The Treeview of the Prototype Definitions with Microcode-Functions References

The sequences of micro-code-functions, which lead to the behavior of the prototype objects and at least the basic behavior of the model elements, are also stored in the database. The order of execution is defined by the database sequence and not by the program code.

A typical sequence of micro-functions is shown in Figure 6.

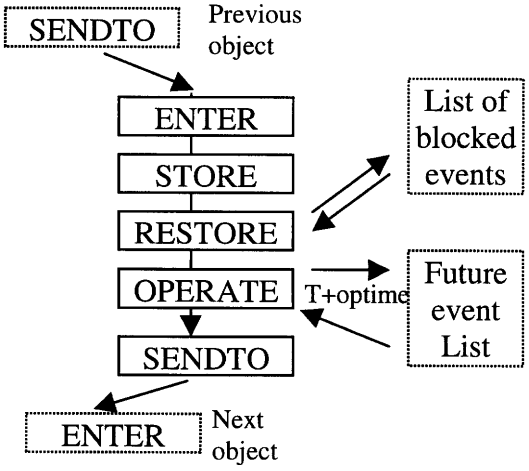

Figure 6: A Typical Micro-code Sequence

With the help of additional parameters of the model object different options of the micro-functions can be selected. In general each micro-function attempts to execute all its own tasks and transfers the dynamic object to the following micro-function. If this does not succeed, the object is blocked. The Sendto-function determines the following model objects and attempts to send the object to one or more of this stations by using their EnterRequest- and Enter-micro-functions.

In the current testing period of the system we have tried to define a set of micro-functions, which is applicable for nearly all typical simulation tasks.

SUMMARY

This paper has offered an approach for building a modern simulation environment. The most important features of the system are:

- all model and simulation data is handled with databases,
- a SQL-like, set-oriented language for the management of the model and simulation experiments,
- the definition of the typical behavior of discrete models with micro-functions, which are realized with a universal programming language.

Even the first prototype of the new simulation environment allows very interesting applications in the area of client-server-simulation and hyper-computing. Thus the presented concept could be an interesting perspective for the future development of modeling and simulation.

REFERENCES

Roberts, C. and Dessouky, Y. C. 1998. An Overview of Object-Oriented Simulation, *SIMULATION* Vol. 70, No. 6, 359-368

Wiedewitsch, J. and Heusmann J. 1995. Future Directions of Modeling and Simulation in the Department of Defense, *Proceedings of the SCSC'95*, Ottawa, Ontario, Canada, July 34-26, 1995

Wiedemann, T. 1997. Perspectives of Component-based Modeling and Simulation, In *Proceedings of the Wold Congress on Systems Simulation* (Singapore, Sep. 1-3 1997), WWW-copy also in [SimCo99]

Wiedemann, T. 1998. Sim-Mining and Simsql - A Database Oriented Approach For Component-Based and Distributed Simulation, *Summer Simulation Conference*, Reno Nevada,

WWW-Link: http://www.aedv.cs.tu-berlin.de/simco/

AUTHOR BIOGRAPHY

THOMAS WIEDEMANN is a Scientific Assistant in the Department of Computer Science at the Technical University of Berlin. His research interests include simulation methodology, tools and environments in distributed simulation and manufacturing processes.