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

THE OBJECT FLOW MODEL FOR DATA-BASED SIMULATION

Lois M.L. Delcambre
Department of Computer Science and Engineering
Oregon Graduate Institute of Science & Technology
Portland, OR 97291, U.S.A
lmd@cse.ogi.edu (503) 690-1689

Lissa F. Pollacia
Department of Mathematical and Physical Sciences
Northwestern State University
Natchitoches, LA 71457, U.S.A
pollacia@alpha.nsula.edu (318) 357-5038

ABSTRACT

This paper introduces the notion of *data-based simulation* to describe simulations where the basic entities and timing for the simulation are provided by explicitly captured data, e.g., in a database. This is in contrast to traditional simulation where entities are usually generated when needed, according to the appropriate distributions.

This paper also introduces the Object Flow Model, where a single model can serve as the basis for the application software, for data-based simulation, and for traditional simulation. The Object Flow Model uses an object-oriented database to describe entities and methods for manipulating entities and provides a visual formalism called the Object Flow Diagram (based on network-based process-oriented discrete event simulation languages), to describe the dynamic processing of an application. A data-based simulator for the Object Flow Model has been implemented for an apparel manufacturing shop floor based on data captured from a real-time payroll system to provide detailed, near-term advice for the shop floor manager.

1 INTRODUCTION

Discrete event simulation models and languages provide a model of some real world application by describing the entities, events, and processing steps of the application [Bratley, et al, 1983; MacDoughall, 1987; Banks, et al, 1984; Nance, 1981]. Traditionally, the entities are described by their type(s) and their arrival or inter-arrival times. Similarly, events are described by their type and their time of occurrence. The processing steps, the way in which they are invoked by events, and the flow of entities through the processing steps are usually described in some sort of visual manner through a directed graph representation, e.g., in network-based, discrete event simulation languages.

Frequently, some aspects of the system to be simulated are already computerized. For the purpose of this paper we describe such a repository of structured data as a database although we know that such data may also exist in a file system, perhaps with less functionality than that provided by a database management system (DBMS).

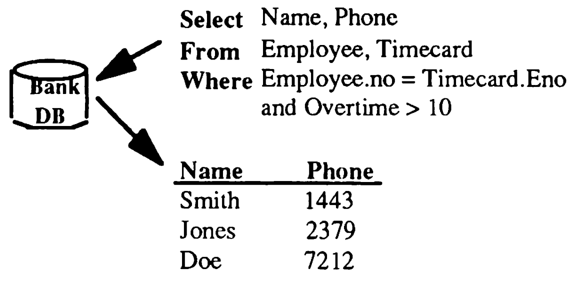
In this paper we explore the use of such data in discrete event simulation. We believe that this type of input can be easily accommodated by a discrete event simulator and that it can result in more specific, detailed output. We believe further that this type of simulation, which we call data-based simulation, has a place in the world of simulation. A data-based simulator for a discrete manufacturing shop floor has been implemented and provides near-term guidance for the shop floor manager.

Based on our experience designing and building this simulator and on the strength of discrete event simulation languages to describe dynamic processing, we have developed the Object Flow Model. The Object Flow Model provides a conceptual model both for application database software and for simulation modeling. The Object Flow Model assumes an object-oriented database and adds a formally defined, visual representation for processing called the Object Flow Diagram. Applications expressed using the Object Flow Model easily support data-based simulation as well as more traditional, statistically-based simulation. The Object Flow Model demonstrates that a single formally-defined model can serve as a model of the application software as well as a simulation model. This, coupled with the idea of data-based simulation, allows simulation to be used as a form of rapid prototyping (including the possibility of animation) for application software.

The organization of this paper is as follows. Section 2 describes the idea of data-based simulation and Section 3 presents the Object Flow Model. Section 4 compares these ideas to related work and Section 5 offers a discussion of work in progress.

2 DATA-BASED SIMULATION

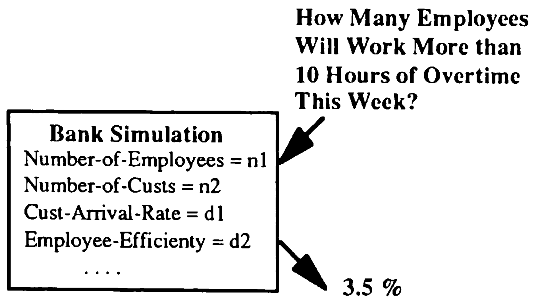
We begin by considering a typical database and a typical simulation environment. Consider the example query issued to a database shown in Figure 1. This is a fairly typical query requiring a join between the Employee relation and the Timecard relation. The query requests a listing of names and phone numbers for employees that have worked more than ten hours of overtime. The query answer lists three specific employees with their telephone extension number. Figure 2, on the other hand, indicates the parameters that might be used for a simple simulation. One possible output from the simulation of the coming week (or of a typical week) would be the number of employees that would work over 10 hours of overtime. The output for this example indicates that 3.5% of all employees would work more than 10 hours of overtime in the simulated week.



```
Select Name, Phone
From   Employee, Timecard
Where  Employee.no = Timecard.Eno
       and Overtime > 10
```

Name	Phone
Smith	1443
Jones	2379
Doe	7212

Database Query Example
Figure 1



Simulation Example
Figure 2

These two examples point out a number of the similarities and differences between database applications and simulations. The similarities include:

(1) They both represent a model of the application.

(2) They both are fundamentally discrete.

(3) They both respond to the question concerning the employees who work more than ten hours of overtime.

One of the most striking differences is the type of questions that can be asked and the level of detail of the answers. Depending on the needs of the user, either of these answers and thus either of these approaches might be appropriate or totally inappropriate. As an example, if a management consultant is trying to discover the general trend for excessive overtime in the bank, then either approach would work but the simulation is probably more direct. On the other hand, if the president of the company would like to write a letter of personal thanks to the individual employee, then only the database approach would work. Some of the differences between these two approaches include:

(1) The database response gives more detail and the information is more accurate. We know for sure that Smith, Jones and Doe worked more than ten hours of overtime in the most recent week and that no other employee did (based on the closed world assumption [Reiter, 1984] implicit in databases). Also, it is very unlikely that a simulation would report the phone numbers for any employees.

(2) The database reports only on the past; the simulation reports predictions for the future (based perhaps, in part, on the past).

(3) The database must hold a potentially large quantity of data (e.g., thousands of tuples for the employee relation and hundreds of thousands of tuples for the timecard relation) whereas the simulation may require only a small number of parameters.

(4) The database is able to match (i.e., join) information about multiple entities; a simulator can do so only probabilistically.

Traditional simulators generally capture the dynamic behavior of a system only in terms of statistical distributions and probabilities. Such models predict "steady state" performance characteristics of the system. For example, such a model of a bank might predict the average throughout and average delay for customers. A queuing theory model is an example of a model (which is often simulated) that deals with steady state performance based only on a statistical characterization of the application.

Data-based simulators, on the other hand, use entities, attributes, relationships, events, and perhaps even service times that are *explicitly* represented. A simulation model becomes more and more data-based in as much as it relies on explicitly captured data. The explicit description of the entities and relationships involved in a simulation has been recognized as an important need, e.g., with the System Entity Structure formalism used with DEVS, the Discrete Event System Specification [Zeigler, 1989]. Traditional databases all provide the facilities to describe, store, and query entities and relationships as defined in the schema. Further, numerous techniques exist to develop appropriate schemas through the construction of E-R diagrams [Chen, 1976], for example. We believe that all simulations can benefit from a logical, relatively high-level description of the data structures involved in a simulation. A data-based simulator can also access the database as needed during a simulation. The database query language provides a means to express the data required by the simulator, e.g., to match customers with their accounts, during the simulation as the database entities progress through the process steps.

Further, if the application system keeps a log of all transactions and they are time-stamped, then this log can serve as a file of transaction-begin and transaction-end events. Such a log can be used to initialize the simulator up to some predefined point (in the past). Perhaps there was one particularly busy day when the teller configuration did not work well. The log file for that day can be used to bring the simulation up until, say, 1:30 PM on that day. The simulation can then explore various options that might have responded to the situation more gracefully.

We use this approach for the apparel manufacturing plant. Based on the data captured by a real-time payroll system, the simulator begins with the current situation and allow the shop floor manager to simulate in the near term, say the next few hours or the next shift. This type of data-based simulation provides the type of detailed information to actually be useful. The manager may want to know "What will the work backload be like by the end of the shift if I move Jones from Station 12 to Station 5 at 2:00 PM?" It is not particularly useful for this manager to be told that on the average, the work backload at the end of the shift is estimated to be 5% of the production. Detailed questions require the detailed use of employee records, machine records, job records, efficiencies, etc.

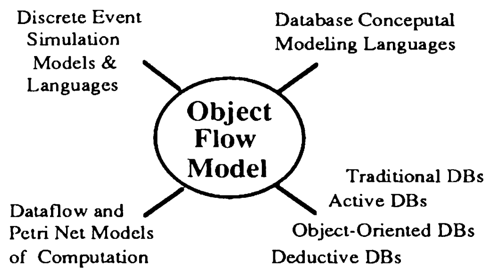
Each simulation may use a database as a source of data to varying degrees. We mean by the term data-based, that the simulator accesses the database directly during the simulation as a source of entity, event, or other detail.

Another use of a database for simulation would be to summarize or statistically characterize the various entity or event populations in a database and then use the distributions uncovered as parameters of the simulation. This uses the database for a form of (statistical) data mining but we distinguish it from the direct database access in data-based simulation. In the other direction, another use of a database would be to explicitly generate the events and/or entities to be used according to the distributions provided and then store them in a database. Although this would be trivially data-based simulation, it serves no purpose other than to use secondary storage. The benefit of data-based simulation derives from the access to actual data that represents details of the real world.

3 THE OBJECT FLOW MODEL

The Object Flow Model builds on the capabilities of an object-oriented database to provide an active, data- and event-driven component to a database [Delcambre, et al, 1990, Delcambre, et al, 1993, Pollacia, 1991]. The Object Flow Model was developed as a conceptual modeling language for what we call *object-driven* applications. The description of the database structure (the entities and relationships) are described in the schema for the object-oriented database. The major contribution of the Object Flow Model is the Object Flow Diagram (OFD) a graphical representation of the object-driven invocation of processing steps.

The motivation for the model and for the OFD came from a number of different research topics, as summarized in Figure 3. Traditional databases contribute the ideas of an explicitly stored extension that conforms to the schema and a query language to access the data. Active databases use rules or triggers to invoke processing steps, often motivated by the occurrence of real-world events. An object-oriented database is assumed for this model to supply rich structural description and methods for manipulation of objects. Deductive databases provide the formal semantics for the OFD. Conceptual modeling languages have described the structure and behavior of database applications but not necessarily with data-driven semantics where data-objects are consumed upon "firing" or invocation of the processing step. The strongest influence for the Object Flow Diagram comes from discrete event simulation languages, for the intuitively appealing graphical representation, and the data flow model of computation, for the invocation of processing steps based on the availability of data. The formal semantics of the Object Flow Diagram arises from predicate transition networks (an extended form of Petri nets) or equivalently from deductive database rule languages [Abiteboul, et al, 1990] The active invocation of processing steps in a database environment was inspired by the dataflow model of computation [Dennis, et al, 1975], and predicate transition nets.


Discrete Event Simulation Models & Languages
Database Conceptual Modeling Languages
Object Flow Model
Traditional DBs
Active DBs
Object-Oriented DBs
Deductive DBs
Dataflow and Petri Net Models of Computation

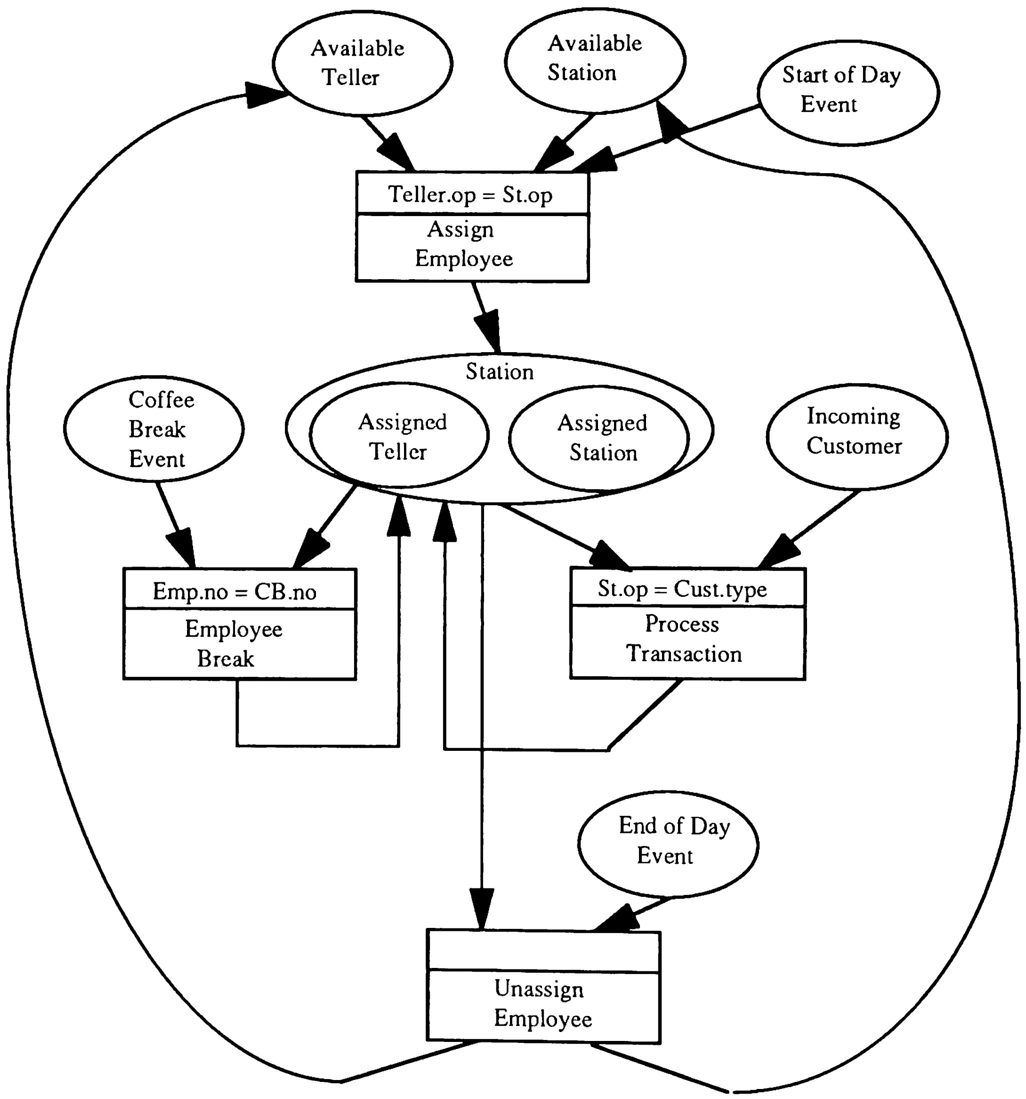
Foundation for the Object Flow Model
Figure 3

The OFD describes the active invocation of processing steps based on the occurrence of external events and on the availability of data. An example OFD is shown in

Figure 4.

The basic OFD is a bipartite, directed graph where ellipses represent explicitly stored collections of objects and rectangles represent processing steps. The basic firing mechanism emulates simulation languages and the dataflow model because the availability of data and events triggers the process. The dataflow model has been extended by placing named database collections of objects on every arc (rather than just holding individual tokens) and by adding a guard or condition that must be true for the process to be triggered. The guard can express selection and/or matching criteria and is shown inside the process box above the line. The usual case is for the database objects (that satisfy the guard and thus trigger the processing step) to be *consumed.* As an example, for the process box labeled Employee Break, the Coffee Break Event is removed from the event collection upon firing. In some cases, the object simply changes state, e.g., from the Available Teller to Assigned Teller collection.

The graphical representation of the OFD is not complete because the details of how the output objects are constructed is not shown. The OFD is completed by a textual language called the Process Flow Language which is a single-assignment language that includes method invocation.



Sample Object Flow Diagram
Figure 4

The OFD shown in Figure 4 indicates the initial processing in the morning where tellers are assigned to stations as long as they are trained for the operation to be performed. This results in an aggregate object being constructed, called Station, that is ready to serve customers. The main processing step is Process Transaction which performs the customer's transaction and produces (i.e., releases) a station. A station may also enter into an Employee Break when the Coffee Break Event occurs for the teller. Finally, at the end of the day, all stations are disassembled, ready for the next day. Note that all input arcs shown in Figure 4 indicate that the inputs are to be consumed (i.e., deleted) from the database collection shown on the input arcs. This ensures that events will each be processed once, and that tellers will only be assigned to one available station. The OFD is non-deterministic. A station for which there is a Coffee Break Event as well as an appropriate Incoming Customer, can go either way (non-deterministically) but the consumption semantics assures that it will only go one way at a time. The formal semantics of the diagram comes from the equivalent rule program or predicate transition network, both of which are known to be non-deterministic.

4 ANALYSIS

Data-based simulation with direct access to the database provides a new concept for simulation where the simulation proceeds at a completely detailed level. For the apparel manufacturing example, we simulate thousands of actual bundles progressing through hundred of stations according to the efficiencies of the actual operators.

Comparison with the Simulation Field

The scope of the Object Flow Model is as a conceptual modeling language for data-based and traditional discrete event simulation. Data-based simulation is specifically targeted for discrete simulation, primarily because databases are fundamentally discrete.

As a simulation model, the Object Flow Model has some features of all three world views commonly used in discrete simulation: event scheduling, activity scanning, and process-oriented [Hooper, 1986]. A world view provides the framework for the system under study and influences both the structure of the simulation model as well as the implementation of the simulator. Figure 5 presents a number of discrete event simulation languages, classified according to their world view. For the references and a more detailed discussion see [Pollacia, 1991].

Event Scheduling	Activity Scanning	Process Oriented
GASII, III	CSL	GPSS
SIMSCRIPT	ECSL	Q-GERT
SLAM,SLAMII		SLAM, SLAMII
		SIMAN
		SIMULA

Common Simulation Languages
Figure 5

The Object Flow Model includes the definition of Event Classes in much the same manner as entity classes. But the OFM is richer than pure event scheduling because the guard is strictly more expressive than simple event-based invocation. The triggering and end-process events can be defined for all process nodes in an Object Flow Diagram, providing a lower-level implementation. However, event scheduling does not handle matching or joining of multiple inputs well.

The activity scanning world view describes applications as activities or operations in which entities engage [Pid84]. Each activity contains test conditions and actions where the actions will be executed whenever the conditions are satisfied. The Object Flow Model uses many elements of the activity scanning approach: conditions are analogous to guards and activities are analogous to process node bodies. But the Object Flow Model includes events and the guard includes the ability to access and match data.

The process interaction world view usually adopts a directed graph or flow diagram to describe all of the operations that an entity engages in during its lifetime. Entities can be conditionally and unconditionally delayed. A common implementation uses a future event list and a current event list (to indicate all steps to be processed in the current time) [Zeigler, 1976]. Each OFD is directly analogous to a process.

Thus, the Object Flow Model world view is essentially a process interaction world view but includes elements of the other two in the form of event classes and individual process nodes with guards (corresponding to elements of the event scheduling and activity scanning world views, respectively). The underlying implementation for the Object Flow Model adopts the traditional event list approach but supplements it with techniques to provide matching for the non-trivial guards, as described in a recent paper [Delcambre, et al, 1993]. The matching technology most suited for the Object Flow Model is the database technology for satisfying rule conditions [Hanson, et al, 1993] inspired by the RETE network [Forgy, 1982] used in expert systems .

Object-oriented simulation models, e.g., [Raczynski, 1988; Petty, et al, 1988; Reddy, et al, 1986; Malloy, et al, 1986] are complementary to the Object Flow Model since we assume an object-oriented model. However, the contributions of the Object Flow Model to object-oriented databases (listed below) apply to object-oriented simulation, as well.

From a simulation point of view, the Object Flow Model contributes a rich database structure for objects and relationships, a conceptual modeling language for the description of simulation models, and, particularly, a formal semantics for the Object Flow Diagrams

Comparison with the Database Field

From a database point of view, the Object Flow Model contributes a formally-defined yet intuitive conceptual modeling language for object-driven applications. Although not discussed here, the Object Flow Model offers rich structuring capability using both generalization and aggregation to relate various collections of objects. We see an interplay between detailed schema design and OFD construction where the schema supplies a rich set of collections to be used in OFDs or, conversely, the OFDs induce various collections in the schema. We believe that the Object Flow Model is complementary to object-oriented databases but contributes a missing component for the active invocation of processing steps. From an active database point of view, the Object Flow Model directly integrates triggers with an object-oriented database, provides a description of an entire set of triggers or a trigger program, and provides formal semantics for the trigger programs. Other contributions include: the ability to invoke processing (directly) based on two or more objects rather than just by a single message, the ability to construct and destruct progressively more complex objects (like the station) and show this construction visually, and the consumption semantics to model (even non-deterministic) object-driven invocation.

Data-based simulation is complementary to traditional simulation, easily exploits existing database and discrete event simulation technology, provides detailed results from a simulation and also provides a form of prototyping for application software. This work is motivated by the work on System Entity Specification [Zeigler, 1984], but brings the idea of structural specification directly into the purview of database systems. This work applies mainly to the area of predictive, state transition models for discrete systems [Fishwick, et al, 1991].

5 FUTURE WORK

The original research leading to the apparel manufacturing near-term simulator was funded by the Defense Logistics Agency of the Department of Defense in conjunction with Clemson University. A second research grant in progress funded by the Louisiana Educational Quality Support Fund is transferring the Object Flow Simulator and testing it in an actual manufacturing plant, Jennings Manufacturing. The installation of the real-time payroll is in progress and the installation of the data-based simulator is planned for later this year.

The formal semantics for the Object Flow Model is currently being finalized through a detailed algorithm that translates the OFD to a deductive database rule program. The challenge is to map the rich structure of an object-oriented database to a rule language intended for relational databases. A graphical editor for Object Flow Model schemas has been implemented and a similar tool for OFDs is currently under development. Finally, the use of the Object Flow Model as the basis for domain modeling, as part of constructing a domain-specific software architecture, is being considered.

ACKNOWLEDGMENTS

The original idea for the apparel manufacturing simulator based on detailed payroll data was suggested by Dr. John C. Peck at Clemson University. He has been a Principal Investigator in both funded projects and has provided invaluable insight into data-based simulation.

REFERENCES

Abiteboul, S. and Simon, E., "Fundamental properties of deterministic and nondeterministic extensions of Datalog", *Journal of Theoretical Computer Science,* 1990.

Banks, J., Carson, J.S., *Discrete Event Simulation*, Englewood Cliffs, 1984, Prentice Hall.

Bratley, P., Fox, B.L., Schrage, L.E., *A Guide to Simulation*, New York 1993, Springer-Verlag.

Chen, P.P.S., "The Entity-Relationship Model: Towards a Unified View of Data", *ACM Transactions on Database Systems*, Vol. 1, No. 1, pp. 9-36, March 1976.

Dennis, J.B. and Misunas, D.P., "A preliminary architecture for a basic data-flow processor", *Proceedings Second Annual Symposium on Computer Architecture*, January 1975, pp. 126-132.

Delcambre, L., Landry, S., Pollacia, L., Waramahaputi, J., "Specifying Object Flow in an Object-Oriented Database for Simulation", *Proceedings. of the SCS Multi Conference on Object-Oriented Simulation*, San Diego, CA, January 1990.

Delcambre, L., Narayanswamy, J., Pollacia, L., "Simulation of the Object Flow Model: A Conceptual Modeling Language for Object-Driven Applications"

Proceedings. 26th Annual Simulation Symposium, Arlington, VA, April 1993, IEEE Computer Society Press.

Forgy, C.L., "A fast algorithm for the many pattern/many object pattern match problem", *Artificial Intelligence*, Vol. 19, pp. 17-37, 1982.

Fishwick, P.A. and Zeigler, B.P., "Quantitative Physics: toward the automation of systems problem solving", *Journal of Experimental and Theoretical Artificial Intelligence*, Volume 3, pp. 219-246, 1991.

Hooper, J.W., "Strategy-Related Characteristics of Discrete-Event Languages and Models", *Simulation*, Vol. 46, No. 4, pp. 153-159, April 1986.

Hanson, E.N. and Widom, J., "An Overview of Production Rules in Database Systems", *Knowledge Engineering Review*, June 1993.

MacDoughall, M.H. *Simulating Computer Systems*, Cambridge, MA, 1987, MIT Press.

Malloy, B., Soffa, M.L., "SIMCAL: The merger of SIMULA and Pascal", *Proceedings. 1986 Winter Simulation Conference*, pp. 397-402.

Nance, R.E., "The Time and State Relationships in Simulation Modeling", *Communications of the ACM*,, Vol. 24, No. 4, pp. 173-179, 1981.

Petty, M.D., Moshel, J.M., Hughes, C.E., "Tactical Simulation in an Object-Oriented Graphics Environment", *Simulator*, Vol. 19, No. 2, pp. 31-46, June 1948.

Pidd, M. *Computer Simulation in Management Science*, John Wiley & Sons, New York, 1984

Pollacia, L., "The Object Flow Model: A Conceptual Modeling Language for Object-Driven Software", Ph.D. Dissertation, University of Southwestern Louisiana, Lafayette, LA, May 1991

Raczynski, S., "Process Hierarchy and Inheritance in PASION", *Simulation*, Vol. 50, No. 6, pp. 249-251, June 1988.

Reddy, Y.V.R., Fox, M.S., Hussain, V., McRoberts, M., "The Knowledge-Based Simulation System", *IEEE Software*, Vol. 3, No. 2, pp. 26-37, March 1986.

Reiter, R., "Towards a Logical Reconstruction of Relational Database Theory", *On Conceptual Modeling*, Ch. 8, pp. 191-233.

Zeigler, B.P., *Theory of Modeling and Simulation*, New York, 1976, John Wiley and Sons.

Zeigler, B.P., *Multi-faceted Modeling and Discrete Event Simulation*, New York 1984, Academic Press.

Zeigler, B.P., "The DEVS Formalism: event-based control for intelligent systems", Proceedings *of IEEE*, Vol. 74, No. 1, pp. 27-80.

AUTHOR BIOGRAPHIES

LOIS M.L. DELCAMBRE is currently an Associate Professor of Computer Science at the Oregon Graduate Institute of Science and Technology. She is also a Pacific Northwest Laboratories Affiliate Staff Scientifst and the Director of the Data-Intensive Systems Center, a research center comprised of researchers from the Oregon Graduate Institute and Portland State University. From 1982-1992 she served on the Computer Science Faculty at the University of Southwestern Louisiana and as the Associate Director of the Apparel CIM Center. Her research interests are in the area of database data models, design databases, conceptual modeling, and expert database systems. She received her B.S. in Mathematics from the University of Southwestern Louisiana, her M.S. in Mathematical Sciences from Clemson University, and her Ph.D. in Computer Science from the University of Southwestern Louisiana. She is a member of ACM and the IEEE Computer Society.

LISSA F. POLLACIA received a B.S. in Mathematics Education from Northwestern State University, an M.S. in Mathematics from Northwestern State University in August, 1980, an M.S. in Computer Science from the University of Southwestern Louisiana in December, 1984, and a Ph.D. in Computer Science from University of Southwestern Louisiana in May, 1991. She was awarded both Master's and Ph.D. Fellowships from USL. She is a member of Phi Kappa Phi. Her professional work experience has included four years as a high school Advanced Mathematics teacher in Leesville, LA, and four years as a Computer Information Systems Instructor for Northwestern. She is currently an Assistant Professor of Computer Science for Northwestern. She is a member of ACM, Mathematics Association of America, Louisiana Academy of Sciences, and Louisiana Association of Computer Using Educators.