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

INTERVAL TIME CLOCK IMPLEMENTATION FOR QUALITATIVE EVENT GRAPHS

Ricki G. Ingalls
SEMATECH
2706 Montopolis Dr.
Austin, TX 78741

Douglas J. Morrice
Andrew B. Whinston
Department of Management Science and Information Systems
Graduate School of Business
The University of Texas at Austin
Austin, TX 78712

ABSTRACT

In this paper, we will discuss implementation issues of a simulation modeling methodology that combines discrete-event simulation with qualitative simulation. Our main reason for doing so is to extend the application of discrete-event simulation to systems found in business for which precise quantitative information is lacking. The approach discussed in this paper is the implementation of interval time specifications in the discrete-event model and the construction of an interval time clock for the qualitative simulation model.

1 INTRODUCTION

Computer simulation is a flexible modeling technique used to solve problems in business, engineering, the physical sciences, and the social sciences. A computer simulation model is a computer program designed to characterize the behavior of an actual system. A number of different approaches to computer simulation modeling exist. In the physical sciences and engineering, a system is often modeled and simulated by a set of differential or difference equations. This approach provides very precise information about the behavior of the system as it evolves through time. It is often referred to as continuous time simulation. The continuous time paradigm has been abstracted or modified in a number of different ways to produce other approaches to simulation modeling.

One popular abstraction used extensively in engineering and business applications is called discrete-event simulation. Discrete-event simulation provides an efficient approach to modeling systems in which the state of the system changes at discrete points in time called events. To model such systems, the clock of the simulation model is incremented asynchronously through time by proceeding from one event to another. The behavior of the system is not monitored continuously through time because it is assumed that nothing of interest happens between events.

Another abstraction of the continuous time approach is called qualitative simulation or qualitative physics. Qualitative simulation was originally developed in the physical sciences (Forbus (1988)), but more recently has found application in economics and business (Hinkkanen et al. (1993)). The qualitative approach is useful when the level of knowledge about the system being modeled is imprecise. In fact, qualitative simulation is designed to represent whatever level of knowledge is available. For example, variables describing the state of a system might be represented in a qualitative simulation model as simply increasing, decreasing or constant with respect to time if no other information is available. Inferences derived from the results of a qualitative simulation model, although less precise, are often considered more general and robust since these inferences do not rely on precise and perhaps faulty assumptions.

Discrete-event simulation suffers from many of the same problems that motivated the pioneers of qualitative simulation. The amount of detail needed in a simulation often is overwhelming. Statistical distributions used in the simulations often are based on incomplete information or the intuition of the model builder. Because of the very nature of discrete-event simulation, appropriate levels of abstraction often are difficult to determine and justify. These problems make discrete-event simulation models difficult to build and to verify. The simulation expert can usually discredit a simulation by criticizing the input distributions, the appropriate level of detail, etc. Decision makers often consider simulation impractical because of lengthy analysis times and the difficulty of evaluating alternatives.

In this paper, we will discuss implementation issues of a simulation modeling methodology that combines discrete-event simulation with qualitative simulation. Our main reason for doing so is to extend the application of discrete-event simulation to systems found in business for which precise quantitative information is lacking. Discrete-event simulation can be qualitatively defined by permitting imprecise specification of elements that are typically quantitatively specified. These include state variables, the simulation time clock, and events scheduled to occur in the future. In this paper, we consider on possible implementation of a qualitatively specified simulation time clock. Another approach is found in Zeigler and Chi (1991).

Our methodology will be developed within the modeling framework of Event Graphs (EG's) (Schruben (1983), Som and Sargent (1989), Yucesan (1990)). EG's are used because they provide a simple yet general representation of a discrete-event simulation (Yucesan (1990)). In addition, various aspects of the simulation model such as the state variables can be represented qualitatively using EG's. We call the combination of event graphs with qualitative simulation *Qualitative Events Graphs (QEGs).* QEGs were introduced in Ingalls, et.al. (1994).

2 BACKGROUND

2.1 Qualitative Simulation

Qualitative Simulation was developed to describe complex physical phenomena in the absence of good quantitative information. Forbus (1988) describes a model that is ideally suited for qualitative simulation when he describes modeling a robot that, among other things, makes coffee. What equation could accurately describe the cup? Several aspects of the cup participate in the physical system and influences what goes on in the system. Such complexity lends itself to a more abstract formulation of the cup and how it interacts with the system. Forbus goes on to explain that even if the equations existed and they could be calculated (using our pocket supercomputer), the output would be insufficient for the robot to use as good information. The robot may need to know alternatives available to him. It is this need for meaningful values and enumerating alternatives that qualitative simulation is meant to do.

The first aspect of the qualitative simulation framework is that an underlying deterministic, continuous model exists. The qualitative model is an abstraction of this underlying quantitative model. This underlying continuous model assumption means that most qualitative models are specified much in the same way that continuous models would be specified. The qualitative model specifies relationships between variables as first-order relationships. These relationships may be very simple such as: As a increases, b increases.

The qualitative simulation uses qualitative state variables. There are several implementations of qualitative state variables. Ordinal-valued variables are completely enumerated and ranked. Interval-valued variables can be implemented as continuous intervals on $\Re$ or as subsets of the ordinal values for a variable. The ordinal set may simply mark time landmarks (Kuipers (1987)) as distinguished time points.

One of the distinguishing characteristics of qualitative simulation is coverage. "A central goal of qualitative physics is to achieve a degree of systematic coverage and uniformity" (Forbus (1988)). In practice, a qualitative simulation simulates all possible threads or envisionments. When a qualitative simulation determines the next possible state, it can easily determine that there are several next possible states because of the imprecise nature of the data. A qualitative simulation will execute each of these possible next states. The resulting set of envisionments will include all of the possible event sequences. One of the envisionments contains the true, underlying, deterministic, continuous model.

2.2 Discrete-Event Simulation And Event Graphs

Event graphs were introduced by Schruben (1983) in order to have a discrete-event simulation methodology that was based on system events. Other types of discrete-event models such as block diagrams, process networks, activity wheel or activity life cycle, are structured from the activity standpoint. The event orientation allows discrete-event simulation without the traditional entity definition that was previously required.

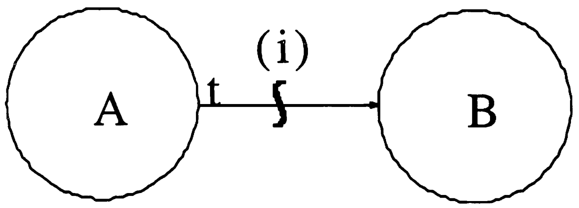

Figure 1: The Most Basic Construct for an Event Graph

The basic construct in the EG framework is shown in Figure 1. The nodes labeled A and B represent events. The edge specifies that there is a relationship between the two events. More specifically, the construct can be interpreted as follows "whenever event A occurs, if condition (i) is true then event B will be scheduled to occur t time units later." The quantity t may assume the value zero, in which case B happens at the same instant as A. Note that it is possible (and often necessary) to specify an edge with no condition.

3 INTERVAL VALUES IN MODELING

3.1 Overview

Interval values on $\Re$ have been used in the qualitative analysis of continuous systems by Kiang, Hinkkanen, and Whinston (1993) and Balakrishnan and Whinston (1991). The purpose for describing state variables with interval values was to allow the user to describe the inherit uncertainty of the decision maker or modeler when it comes to the true value of the variable. Interval specification allows the decision maker to refine the estimate of the true variable value as he gathers more information. For example, if an input to the model was demand, and the (unknown) actual demand value was 42, the decision maker may be able to run experiments or by intuition determine that the true demand lies in the range [35,45]. For the purposes of this paper, we have only allowed the interval value specification to be used to describe activity times (or delay times) in the discrete-event model.

3.2 Interval Math

In order to be able to handle interval values, we adopted the conventions of Allen (1983).. Figure 2 gives the algebra that is used for intervals in this implementation. In addition, we will use the following notation:

and:	&
or:	$\mid$
negation:	!

This algebra was sufficient to implement interval time for discrete-event simulation.

4 Interval Time Calendar In Qualitative Event Graphs

A *future events calendar* or simply a *calendar* is a method for determining the order of events in discrete-event simulation. Discrete-event simulation calendars are a strongly ordered list of future events. Typically, this list is always sorted by time where the earliest events are at the front of the list. If there are events that scheduled to occur at the same time, the order is determined by some predetermined rule or a user priority. Predetermined rules vary from first-on, first-off to random and vary according to implementor's discretion. Regardless, the future events list maintains its strongly ordered characteristics.

In the simplest qualitative discrete-event model, one with constant delay times, it is likely that there would be ties on the future events calendar. But unlike quantitative discrete-event simulation, the qualitative implementation would not assume a tie breaking strategy, but rather the qualitative model would create a thread, or envisionment, for every possible ordering of the ties. Thus the qualitative future events calendar looses the strongly ordered characteristic of its quantitative counterpart. In fact, the qualitative future events calendar becomes the union of many *non-deterministicly ordered sets (NOSs).* In essence, a NOS is a set of events on the calendar whose execution order is uncertain. We call the members of a NOS *non-deterministicly ordered events (NOEs).* We will elaborate on the concept and use of NOS later in the

Relation	Symbol	Symbol for Inverse	Definition	Example
t before s	<	>	$t^+ < s^-$	TTT SSS
t equals s	=		$(t^- = s^-)$ and $(t^+ = s^+)$	TTT SSS
t overlaps s	o	oi	$(t^- < s^-)$ and $(t^+ > s^-)$ and $(t^+ < s^+)$	TTT SSS
t meets s	m	mi	$t^+ = s^-$	TTTSSS
t during s	d	di	$((t^- > s^-)$ and $(t^+ <= s^+))$or $((t^- >= s^-)$ and $(t^+ < s^+))$	
t starts s	s	si	$t^- = s^-$	TTT SSSSSS
t finishes s	f	fi	$t^+ = s^+$	TTT SSSSSS
In addition, we defined the following:				
t intersects s	i		(t overlaps s) $\mid$ (s overlaps t)	TTT SSS
t + s	+		$[t^- + s^-, t^+ + s^+]$	
t - s	-		$[t^- - s^-, t^+ - s^+]$	
max(t,s)	max		$[\max(t^-,s^-),\max(t^+,s^+)]$	
min(t,s)	min		$[\min(t^-,s^-),\min(t^+,s^+)]$	
intersection(t,s)	$\cap$		$[\max(t^-,s^-),\min(t^+,s^+)]$,if $(\max(t^-,s^-) \leq \min(t^+,s^+))$	
			$\varnothing$,otherwise	

Figure 2: Interval Algebra for a given interval $t = [a,b]$, $t^- = a$, $t^+ = b$.

paper. Because we do not have a strongly ordered calendar, and because one of the foundations of qualitative simulation is coverage (Forbus, 1988), the qualitative discrete-event simulation creates different threads, or envisionments, to execute the possible orderings of a NOS.

4.1 Interval Time Calendar Constructs

A *calendar event* is an instance that includes time, the node that put the event on the calendar (fromNode), the node that will be executed (toNode), priority of the event, and any attributes that are passed to the node that will be executed. For the purposes of this paper, we will refer to event A's interval time as A.time. The node that event A will execute as A.toNode. The event priority is denoted as A.priority.

A *calendar* is an ordered collection of calendar events that are ordered where each event's time is ascending. Specifically, if A and B are two calendar events, A precedes B in the calendar if:

1. A.time < B.time.
2. A.time intersects B.time & A.priority < B.priority
3. $A.time^- < B.time^-$
4. $A.time^- = B.time^-$ & $A.time^+ < B.time^+$

A calendar also has an attribute that tracks the current time of the simulation. For the purposes of this paper, we will refer to calendar C's current time as C.currentTime.

4.2 The Interval Time Calendar Algorithm

The *Interval Time Calendar(ITC)* has been implemented in Smalltalk. The Interval Time Calendar currently implemented executes a depth-first search on all possible threads. This calendar is implemented as follows:

```
0. Create a calendar with the designated "first event" on it. Designate calendar as the current calendar and the first event as the current event. Set the current time to [0,0].
1. Begin a new thread with the current calendar starting with the current event as the first event being executed the time of its execution.
2. If this is the first event of the new thread, go to Step 7.
3. Determine the NOS that must be executed next.
4. If the number of NOE ∈ NOS = 1, go to Step 7.
5. Save the current state of the system.
6. Loop over the NOS (i = 1 to number of NOEs)
    6.1. Determine the execution time of the event i, and make event i the first event.
    6.2. Call Step 1.
    6.3. Restore the system state stored in Step 5. If last NOE, Return.
7. Execute the event.
8. If the thread has reached a simulation stopping condition, Return.
9. If there are no more threads on the stack, stop the simulation.
10. Go to Step 2.
```

4.3 Non-Deterministicly Ordered Sets And Events

When the order of events is uncertain (step 6 above), the simulation will try all the combinations of the events. For any two events A and B, event orderings are considered uncertain if A.time intersects B.time and A.priority = B.priority. When these conditions exist, then the set of events are NOS. In the qualitative model, the future events calendar is made up of a union of NOSs. For the calendar mechanism to work correctly, we must be able to determine the NOS that will be executed next given the current state of the calendar. More formally, the NOS of step 6 in the algorithm is determined as follows:

P is defined as the NOS in step 6.

Given the current calendar C, c_i is the ith event on the calendar.

$P = \{ c_1.time \text{ intersects } c_i.time \; \& \; c_1.priority = c_i.priority \} \; \forall c_i \in C.$

The only assumption in this statement is that c_1 could be the next event to be executed. In the calendar implementation, we have guaranteed that the condition holds.

Upon determining the non-deterministicly ordered events that make up the NOS, this qualitative simulation creates a thread for each event in the set. In each thread, the ith event in the set becomes the first event to be executed. The remaining events in the set remain in the future events calendar for that thread.

Each thread must determine what the new current calendar time will be for that thread. After the set P is determined, there is an issue of calculating the current time of the new simulation thread.

```
Let P be the NOS of step 6
    $p_i$ is the ith event in the set.
    C is the current active calendar.
    $D_i$ is the calendar for the thread whose first event is $p_i$.
For each $p_i \in P$
1. Set $D_i = C$.
2. Set $D_i.currentTime = [\max(p_i.time^-, C.currentTime^-), \min(p_j.time^+ \; \forall p_j \in P)]$
3. Execute Step 6.2 with calendar $D_i$ and first event $p_i$ starting at time $D_i.currentTime$
```

5 AN EXAMPLE

As an example, consider the EG of a drive-in teller with a single service window and large parking area for waiting cars. The node labeled RUN is the first node executed in the system. It is used to initialize the state variables. The node labeled ENTER represents the event that a car arrives to the system. The nodes START and LEAVE represent the following events: begin service at the window and end service at the window, respectively. The state variables are Q for the number of cars waiting for service, S for the status of the teller and E for the number of customers who have exited. If S=1, then the teller is idle; if S=0, then the teller is busy. Since changes in the state variables happen only when an event occurs, the changes are stated below each event. For example, when the START service event occurs, the queue of waiting cars decrements by one (Q=Q-1) and the teller status changes to busy (S=0).

The conditional expressions that appear on some of the edges are based on the state of the system. For example, the condition between the ENTER event and the START event is a condition to test if the teller is idle (S>0). In other words, if a car arrives and the drive through window is not busy, then the car enters service immediately. It should also be noted at EG edges can have priorities. This model has high priorities (1) on the edge between ENTER and START and the edge between LEAVE and START.

5.1 Interval Time Calendar Example

Using the example model in Figure 3, we want to illustrate how ITC works by simulating 5 customer service completions (LEAVE events). The example below tracks the first thread in the system to completion. Thread 520 is created when the model cannot determine the ordering for the two events on the calendar: [3,8],ENTER,9 and [4,6],LEAVE,9. These two events have event times that intersect ([3,8] and [4,6]) and have the same priority (9). Thread 1 continues as if [3,8],ENTER,9 was the first event. Thread 520 continues as if [4,6],LEAVE,9 was the first event. The remaining non-deterministic orderings found in threads 1 and 520 help create the 897 threads that were created during the simulation run.

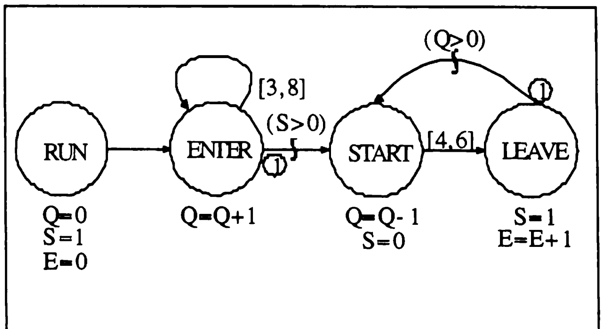

Figure 3: The Qualitative Example

5.2 Interval Time Outputs

When interval time is used in discrete-event simulation, the results are can be analyzed by examining the timing of events or conditions. For example, Figure 4 illustrates that the fifth exit occurs in interval [20,30] for thread 1 and [20,32] for thread 520. Using all possible threads, one can determine that the fifth exit of the system must occur in the interval [20,38]. Interpreting the inputs to the model, we can say that regardless of the distribution placed on the arrival time interval of [3,8] and the service time interval of [4,6], the fifth exit of the system will occur in the interval [20,38]. For example, we ran 100 replications of a quantitative model where the input distributions were uniform over [3,8] and [4,6] and the fifth exit occurred between 21.7726 and 34.5777 in those experiments.

5.3 Coverage

As was stated earlier in the paper, a primary goal of qualitative modeling is coverage. This qualitative model is, in one sense, characterizing all possible sample paths of a structurally equivalent quantitative EG. Of the 897 threads created in our example, one would replicate the event sequence and bound the event time for any sample path generated by a structurally equivalent EG whose input distributions were bounded by [3,8] and [4,6], respectively. Second, it bounds the time that any specific event (or state) can occur. As with our example, we were able to bound the fifth exit in the interval [20,38]. Bounding the time of a specific event in this way can be very useful in determining if some given design limit has in fact been met. Third, the QEG characterizes all the possible states that could occur. This is very important because one is guaranteed that everything outside the generated threads is impossible.

6 Research Issues

During the course of this work, several key research issues came to light. First, there is the issue of the amount of computation required to generate the threads. Here, we have presented a depth-first search. We are currently working on breadth-first and other thread managing techniques to reduce the computation time. Second, there is research to be done in the area of characterizing the threads that are generated by the simulation. Are some threads "subsets" of others? When can we know that a thread contains no new

Steps for Thread 1	Thread	Calendar Time	Event Executed	S	Q	E	Future Events (Time,Node,Pri)	Thread	Calendar Time	Event Executed	S	Q	E	Future Events (Time,Node,Pri)
0,1,2,7,10	1	[0,0]	RUN	1	0	0	[0,0],ENTER,9							
2,3,4,7,10	1	[0,0]	ENTER	1	1	0	[0,0],START,1 [3,8],ENTER,9							
2,3,4,7,10	1	[0,0]	START	0	0	0	[3,8],ENTER,9 [4,6],LEAVE,9							
2,3,4,5,6,6.1, 6.2,1,2,7,10	1	[3,6]	ENTER	0	1	0	[4,6],LEAVE,9 [6,14],ENTER,9	520	[4,6]	LEAVE	1	0	1	[3,8],ENTER,9
2,3,4,5,6,6.1, 6.2,1,2,7,10	1	[4,6]	LEAVE	1	1	1	[4,6],START,1 [6,14],ENTER,9	520	[4,8]	ENTER	1	1	1	[4,8],START,1 [7,16],ENTER,9
2,3,4,7,10	1	[4,6]	START	0	0	1	[6,14],ENTER,9 [8,12],LEAVE,9	520	[4,8]	START	0	0	1	[7,16],ENTER,9 [8,14],LEAVE,9
2,3,4,5,6,6.1, 6.2,1,2,7,10	1	[6,12]	ENTER	0	1	1	[8,12],LEAVE,9 [9,20],ENTER,9	520	[7,14]	ENTER	0	1	1	[8,14],LEAVE,9 [10,22],ENTER,9
2,3,4,5,6,6.1, 6.2,1,2,7,10	1	[8,12]	LEAVE	1	1	2	[8,12],START,1 [9,20],ENTER,9	520	[8,14]	LEAVE	1	1	2	[8,14],START,1 [10,22],ENTER,9
2,3,4,7,10	1	[8,12]	START	0	0	2	[9,20],ENTER,9 [12,18],LEAVE,9	520	[8,14]	START	0	0	2	[10,22],ENTER,9 [12,20],LEAVE,9
2,3,4,5,6,6.1, 6.2,1,2,7,10	1	[9,18]	ENTER	0	1	2	[12,18],LEAVE,9 [12,26],ENTER,9	520	[10,20]	ENTER	0	1	2	[12,20],LEAVE,9 [13,28],ENTER,9
2,3,4,5,6,6.1, 6.2,1,2,7,10	1	[12,18]	LEAVE	1	1	3	[12,18],START,1 [12,26],ENTER,9	520	[12,20]	LEAVE	1	1	3	[12,20],START,1 [13,28],ENTER,9
2,3,4,7,10	1	[12,18]	START	0	0	3	[12,26],ENTER,9 [16,24],LEAVE,9	520	[12,20]	START	0	0	3	[13,28],ENTER,9 [16,26],LEAVE,9
2,3,4,5,6,6.1, 6.2,1,2,7,10	1	[12,24]	ENTER	0	1	3	[15,32],ENTER,9 [16,24],LEAVE,9	520	[13,26]	ENTER	0	1	3	[16,26],LEAVE,9 [16,34],ENTER,9
2,3,4,5,6,6.1, 6.2,1,2,7,10	1	[15,24]	ENTER	0	2	3	[16,24],LEAVE,9 [18,32],ENTER,9	520	[16,26]	LEAVE	1	1	4	[16,26],START,1 [16,34],ENTER,9
2,3,4,5,6,6.1, 6.2,1,2,7,10	1	[16,24]	LEAVE	1	2	4	[16,24],START,1 [18,32],ENTER,9	520	[16,26]	START	0	0	4	[16,34],ENTER,9 [20,32],LEAVE,9
2,3,4,7,10	1	[16,24]	START	0	1	4	[18,32],ENTER,9 [20,30],LEAVE,9	520	[16,32]	ENTER	0	1	4	[19,40],ENTER,9 [20,32],LEAVE,9
2,3,4,5,6,6.1, 6.2,1,2,7,10	1	[18,30]	ENTER	0	2	4	[20,30],LEAVE,9 [21,38],ENTER,9	520	[19,32]	ENTER	0	2	4	[20,32],LEAVE,9 [22,40],ENTER,9
2,3,4,5,6,6.1, 6.2,1,2,7,8	1	[20,30]	LEAVE	1	2	5	[20,30],START,1 [21,38],ENTER,9	520	[20,32]	LEAVE	1	2	5	[20,32],START,1 [22,40],ENTER,9

Figure 4: ITC Threads Example

information? Under what conditions can threads "merge"? These are a few of the questions that need to be addressed. Third, we plan to expand the use of intervals to state variables.

As we ran the qualitative models, a fourth area of research became evident: how to assign priorities to arcs so that the model executes in a fashion that you desire. In our example, when we did not have two arcs with priority 1, we found ourselves in states where Q < 0 or the calendar would have more than one LEAVE event scheduled. Using QEG to determine priorities on arcs in order to avoid these system states would be desirable from an EG modeling standpoint (Ingalls, et.al., 1994).

ACKNOWLEDGMENTS

We gratefully acknowledge the financial support of the CBA/GSB Faculty Research Committee of the College of Business Administration of the University of Texas at Austin.

REFERENCES

Allen, James F., 1983. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11), 832-843.

Balankrishnan, Anantaram and Andrew B. Whinston, 1991. Information Issues in Model Specification. *Information Systems Research.* 2 (4), 263-286.

Forbus, K.D., 1988. Qualitative Physics: Past, Present, and Future. *Exploring Artificial Intelligence*, Howard Shrobe, ed. San Mateo: Morgan Kaufmann Publishers, Inc.

Hinkkanen, A., Lang, K.R., and Whinston, A.B., 1993. A Theoretical Foundation of Qualitative Reasoning Based on Set Theory. Working Paper.

Ingalls, Ricki G., Douglas J. Morrice, and Andrew B. Whinston, 1994. Qualitative Discrete Event Simulation Using Event Graphs, to appear in

Proceedings of the 1994 European Simulation Multi-Conference, Barcelona, Spain, June, 1994.

Kiang, Melody Y., Aimo Hinkkanen, and Andrew Whinston, 1993. Reasoning in Qualitatively Defined Systems Using Interval-Based Difference Equations. Working Paper.

Kuipers, B., 1987. Qualitative Simulation as Causal Explanation. *IEEE Transactions on Systems, Man, and Cybernetics* 17(3), 432-444.

Schruben, L.W., 1983. Simulation Modeling with Event Graphs. *Communications of the ACM*, 26(11), 957-963.

Som, Tapas K. and Robert G. Sargent, 1989. A Formal Development of Event Graphs as an Aid to Structured and Efficient Simulation Programs. *ORSA Journal on Computing*. 1(2), 107-125.

Yucesan, E., 1990. Simulation Graphs: A Mathematical Framework for The Design and Analysis of Discrete Event Simulations. Ph.D. Dissertation, School of Operations Research and Industrial Engineering, Ithaca, New York.

Zeigler, B.P. and Chi, S. 1991. Symbolic Discrete Event System Specification. *Proceedings IEEE Conference on AI, Simulation and Planning in High Autonomy Systems*, Cocoa Beach, Florida, 130-141.

AUTHOR BIOGRAPHIES

RICKI G. INGALLS is on the technical staff of the Operational Modeling Group at SEMATECH in Austin, Texas. He has been involved in the application and development of operational modeling tools and techniques in the electronics industry for over 10 years. He has a B.S. in Mathematics from East Texas Baptist College, a M.S. in Industrial Engineering from Texas A&M University and is currently in the Management Science Ph.D. program at the University of Texas at Austin. Prior to joining SEMATECH, he was Manager of Operations Analysis at Compaq Computer Corporation, a consultant with the Electronics Automation Application Center of General Electric Co. and an Industrial Engineer with Motorola, Inc. He is a member of the Society for Computer Simulation.

DOUGLAS J. MORRICE is an assistant professor in the Department of Management Science and Information Systems at The University of Texas at Austin. He received his undergraduate degree in Operations Research at Carleton University in Ottawa, Canada. He holds a M.S. and Ph.D. in Operations Research and Industrial Engineering from Cornell University. His research interests include discrete event and qualitative simulation modeling, the statistical design and analysis of large scale simulation experiments, and the statistical aspects of quality control. He is a member of The Institute of Management Science and the Operations Research Society of America.

ANDREW B. WHINSTON is a professor of both business and computer science at the University of Texas at Austin, where he is also a fellow of the IC2 Institute and director of the Center for Information Services Management, and holds the Hugh Roy Cullen Centennial Chair in Business Administration. His research deals with decision support systems theory, distributed AI, organization modeling and qualitative modeling. He received his Ph.D. in management from Carnegie Mellon University, Pittsburgh, PA, and is a member of the Institute of Management Science.