Proceedings of the 2005 Winter Simulation Conference
M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.

QUALITATIVE DISCRETE EVENT SIMULATION

Yen-Ping Leow-Sehwail
Ricki G. Ingalls

School of Industrial Engineering & Management
322 Engineering North
Oklahoma State University
Stillwater, OK 74078, U.S.A.

ABSTRACT

A real world system is full of uncertainties and more than often the parameters, processes or events under study are poorly understood. In order to study a real world system, we often have to make a set of assumptions about how it works using statistical, mathematical or logical relationships. Qualitative discrete event simulation involves the development of simulation models which require less assumptions, less data requirements and yet more robust. This paper presents the concepts involved in the development and implementation of qualitative discrete event simulation models and algorithms.

1 INTRODUCTION

Simulation continues to be a widely used technique for solving problems in engineering, business, physical science, artificial intelligence and economics. It is without hesitation that simulation is a powerful and important tool as it provides a method for evaluating existing or alternative decisions, plans and policies without having to conduct real experiments. In some situations, simulation is set as a required analysis tool before any major capital investments.

The creation of a simulation model could be highly dependent on the availability of system data. If data is available, modelers normally would begin with the development of a frequency distribution or histogram of the data and then try to fit it to a family of probability distributions. Sometimes, it is impractical or impossible to get these data and even if the data is available, there are still other issues that need to be addressed such as data overload and data abstraction. A great deal of effort is required to distill quality data and encode the information that is available into building a simulation model.

A regular discrete event simulation requires the modelers to specify exactly which family of probability distributions together with any associated parameters that serves as input to the simulation model. The choice of input model could significantly impact the prediction or decision to be made based on the simulation results. Depending on the type of distributions that is used in the model, the modelers are required to make the necessary assumptions. Every simulation analysis is based on a variety of assumptions that are made about the nature of the data. It is likely to make erroneous conclusions if one does not carefully evaluate the validity of the assumptions behind the analysis. It has become more of a standard procedure to make these assumptions in a regular discrete event simulations.

Qualitative simulation has created a new construct to allow the modelers to specify input parameters and state variables qualitatively. Different qualitative specifications have been developed over the years (which will be discussed later in Section 2). Qualitative simulation inherits all the benefits of simulating and transforming the study of the complex systems and yet offers more than "what-if" analysis, which allows the modelers to answer wider range of questions that are of interests to users. Questions regarding the behavior of the system especially those that concern the viability of the system, i.e. under what conditions the system will fail, could be answered using qualitative simulations. It is difficult to answer these types of questions that explore the transient behavior of the system using a regular discrete event simulation. The reason is that the transient behaviors that a regular discrete event simulation is able to detect are behaviors that the simulation randomly generates. Also, the qualitative approach is very useful when the level of knowledge about the system is imprecise. In fact, qualitative simulation is designed to represent whatever level of knowledge is available. This could be a solution to solving complex problems in many industries that have been struggling with data overload and data abstraction.

Although there has been an increasing attention in the field of qualitative simulation, a literature search for the research developed is found mainly in the area of continuous time models. The number of research work in the area

of qualitative discrete event simulation is barely a handful. The future for qualitative discrete event simulation looks bright and promising as it combines all the abilities of a regular discrete event simulation and yet provides more flexibility in specifying input parameters with less data requirements, less assumptions and a more "complete" solution.

The purposes of this paper are twofold. First, this paper reviews the literature on the development and implementation of qualitative discrete event simulation. Second, this paper offers research propositions to motivate future research, based on the material presented. This paper is not intended to provide detailed implementation of the qualitative discrete event simulation. The authors focus on providing a general idea and understanding of the concepts in this area. This paper is organized in the following manner. The next section presents a brief introduction of qualitative simulation. The general concepts of qualitative discrete event simulation is presented in Section 3. Section 4 presents the algorithms that have been developed and implemented. Section 5 addresses the use of thread scoring techniques to separate the more-likely-to-happen sequences of events from the less-likely-to-happen sequences. Future research proposition is presented in the final section.

2 QUALITATIVE SIMULATION

Qualitative simulation is a reasoning technique that derives useful inferences from the modeling of a system when there is a lack of good quantitative information about the system under consideration. It is motivated by the desire to reason about the objects, processes or events in order to uncover all possible behaviors that may exist in the system. One major distinction between qualitative simulation and quantitative simulation is that a quantitative model is a result from a *particular* experiment while the output from a qualitative model is in response to *all possible* experiments (Cellier 1991). 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 then execute each of these possible next states which results in a series of envisionments of all possible event sequences.

Another major distinction is in the representation of state variables. A model is quantitative if the state variables are real-valued, otherwise the model is qualitative (Cellier 1991). This is an important distinction that exhibits in different types of simulation models. Simulation models are largely classified into two major types, namely the continuous simulation and discrete event simulation.

For the case of continuous simulation that often deals with the modeling of a physical system over time by a representation in which the state variables change continuously with respect to time, a regular continuous simulation model often uses differential equations to describe the rate of change or the interaction of state variables with time. The abstraction is based on ordinary differential equations, which are numerical in character. Kuipers (2001) described a continuous time qualitative simulation using qualitative differential equation model as an abstraction of an ordinary differential equation, consisting of a set of real-valued variables with functional, algebraic and differential constraints among them. Qualitative differential equation consists of variables which are described in terms of their ordinal relations with a finite set of symbolic landmark values. The relationships could be a first-order relationship as simple as: As a increases, b increases. The values a and b could be described as increasing or decreasing over a particular ranges.

As for discrete event simulation, the concept of qualitative simulation was further explored by Ingalls (1999) who then developed a simulation methodology that combines discrete event simulation with qualitative simulation using *temporal interval* as simulation time specification. Temporal interval allows the user to define time as an interval in the simulation. This means that the current state of the simulation can occur at any time during the interval defined by $t = [t\text{-},t\text{+}]$. Temporal intervals are represented by the modeling of their endpoints, assuming for any interval t, the lesser endpoint is denoted by t- and the greater by t+. The authors would like to acknowledge that when we mention Qualitative Discrete Event Simulation (QDES) in this paper, QDES refers to the qualitative approach taken by Ingalls (1999).

Another qualitative approach to discrete event simulation was taken by Ziegler and Chi (1992), known as the Symbolic Discrete Event System which uses the linear polynomial representation. This approach allows the modelers to express times symbolically for both situations when timing of events is unknown and when timing is known, but can vary. The time advance values could be expressed as linear polynomials such as *1,2, s, t, 2s, t+s, 2t-s+9,* etc, which must evaluate to nonnegative real numbers (Ziegler and Chi 1992).

3 QUALITATIVE DISCRETE-EVENT SIMULATION (QDES)

Qualitative discrete event simulation (QDES) is an event-scheduling approach to simulation modeling developed by Ingalls (1999). It extends the concept of qualitative simulation to be applied particularly in discrete event systems. QDES applies the next event time advance approach as in a regular discrete-event simulation (RDES). At the start of a simulation model, the simulation clock is initialized to zero and the time of occurrence of the most imminent future event is determined and inserted into a future event calendar. The simulation clock is advanced from the occurrence of one event to another at which the state of the system and the times of the occurrence of future events are updated.

This process advances the simulation clock until all the events in the future event calendar have been executed or canceled, or when a certain stopping criterion is met.

One distinctive characteristic of QDES is that it allows the modelers to specify elements qualitatively. These elements include the times of occurrence of future events, the simulation time clock and state variables.

The times of occurrence of future events are represented as time points in RDES. Unlike in RDES, QDES assume imprecise specification of event occurrences. The uncertainty of the event time is represented in a closed time interval in $\mathcal{R}$, which is also known as temporal interval. There are two types of temporal intervals, namely the constant interval and the uncertain interval. A constant interval is an interval whose values remain the same throughout the entire simulation, while an uncertain interval is an interval that changes values every time the interval is evaluated. In using constant intervals, it is assumed that the actual value of the variables is a constant that lies somewhere within an interval. An uncertain interval is equivalent to the modeling of sampling from an unknown probability distribution that is bounded by an interval.

As a result of the time interval representation in QDES, the future event calendar is also represented as time intervals. Future event calendar in QDES is also sorted according to event times but it is not a strongly ordered list. Events are sorted according to interval mathematics outlined in Allen (1983). It is likely that there would be ties on the future event calendar because of the uncertain order of events. If there is a tie, QDES would not assume a tie breaking strategy as in the case for RDES. Instead, QDES would create threads that make up all of the possible ordering of ties, which differentiates between QDES and RDES. The differences are that RDES's future event calendar is a strongly ordered list according to the event times and RDES uses several tie-breaking mechanisms to determine the order of simultaneous events, sometimes ad hoc and specific to the simulation protocols being used. The future event calendar in QDES collects all the event notices whose execution order is uncertain and group them in a set, called the nondeterministically ordered set (NOS).

The capability of generating all possible scenarios is achieved with the thread generation algorithm. There are currently two algorithms developed so far, namely the Depth-First algorithm and the Breadth-First algorithm. These two algorithms are discussed in the next section in more detail. The execution of QDES algorithms resembles the RDES to some extents. The algorithms contain the basic steps of RDES such as initializing the simulation clock, advancing simulation clock from its current value to another, inserting events into the future event calendar, determining the next event to be executed and so on. When QDES is executed, the next possible state is determined. Due to the lack of precise data about the real-world, there could be several next possible states or a tie. QDES algorithm will have some major additional steps to ensure that each of these states will be executed in turns and result in a set of threads that will include all of the possible ordering of event sequences. An example of a new thread being generated is when the execution times (expressed in temporal intervals) for two or more events overlap.

The distinctive characteristic of QDES of generating all possible ordering of event sequences is known as coverage (Ingalls, Morrice et al. 2000). The coverage property ensures that all outcomes of QDES are characterized and no outcome will be missed out. In RDES, the simulation outcome is based on a sampling approach. The coverage property in QDES is very useful for planning and scheduling problems. Schedules would not have to be rerun every time if something did not happen according to plan. The output of QDES would be able to characterize the changes of events and give information on the next scheduling position, as long as the input interval is respected. Another advantage of coverage property is in debugging simulation models. QDES would characterize all possible scenarios, including anything that is characterized in a RDES model and sequences that have low probability of execution. This would give the modeler absolute confidence in the validity of the simulation model (Ingalls, Morrice et al. 2000).

Allen (1983) mentioned that time-point representation does not allow any uncertainty of information and often the exact relationship between two time-points is not known. Thus, the notion of time point is not decomposable and is not useful in a reasoning system. Temporal interval representation is sometimes more useful in certain situations. An example to illustrate the use of temporal interval is shown in the modeling of the start time of a crisis. Let's say that there is an alarm system that triggers to inform the appropriate authority when there is a crisis. In this case, the start time of the crisis is not known even if the start time of the triggering alarm could be determined accurately. An upper bound could be placed on the start time of the crisis if we assume that the crisis happen at a time earlier than the alarm time. In this case, it is only possible to specify the start time of the associated crisis as a time interval.

Temporal interval specification extends the regular time base from real numbers to interval representation. Another form of time specification is extended to linear polynomials over the real numbers, also known as the Symbolic Discrete Event System specification (Ziegler and Chi 1992). This specification also serves the purpose of representing uncertainty in event execution times. The linear polynomial representation is used to allow manipulation of expressions for time with symbols representing unspecified event times. For example, let p = waiting time at register A, and q = process time at register A, then "p+q" is a valid expression according to symbolic discrete event system specification. This expression could be used to represent the time a customer spends at register A where p and q are symbols that are used to represent unspecified event

times. When p and q are assigned numerical values, the expression evaluates to a real number.

4 QDES ALGORITHM

QDES is designed and developed using simulation graph model. The simulation graph framework was first introduced by (Yucesan and Schruben 1992); (Schruben and Yucesan 1993) and then extended by (Ingalls 1999). Let simulation graph, $G\ (V(G),E_S(G),E_C(G),\Psi_G)$ be a directed graph where $V(G)$ is the set of vertices of G, $E_S(G)$ is the set of scheduling edges, $E_C(G)$ is the set of canceling edges and Ψ_G is an incidence function that associates with each edge of G. The Simulation Graph Model (SGM) is then defined as $S = (F,C,X,T,\Gamma,G)$ where

$F = \{f_v: v \in V(G)\}$, the set of state transitions functions associated with vertex v
$C = \{C_e: e \in E(G)\}$, the set of scheduling edge conditions
$X = \{X_e: e \in E(G)\}$, the set of execution edge conditions
$T = \{t_e : e \in E_S(G)\}$, the set of edge delay times as time intervals, and
$\Gamma = \{\gamma_e\ e \in E_S(G)\}$, the set of event execution priorities

The simulation graph G specifies the relationships that exist between the elements of the set of entities in a simulation model. The basic construct of the event graph with edge execution condition is given in Figure 1. The nodes labeled A and B represent events and the edge specifies that there is a relationship between the two events. The construct is interpreted as follows: If condition (i) is true at the instant event A occurs, then event B will be scheduled to occur t time units later. Event B will be executed t time units later with the state variables in array n set equal to the values in array k if condition (j) is true t time units later.

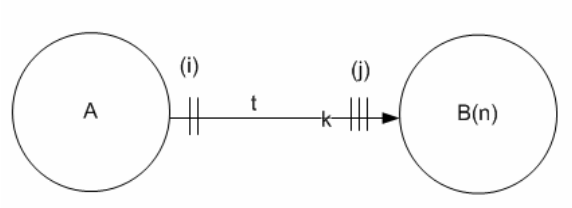

Figure 1: Event graph with execution condition

4.1 Depth-First Algorithm

A depth-first algorithm was created and implemented by Ingalls (1999) that would completely finish generating one whole thread before moving on to another thread. Any additional threads will be stored on a stack waiting to be executed at a later time. When there is a tie, two threads or more will be generated. The state of the simulation after each thread explosion is saved in a stack called the save-state stack. Each save-state consists of a global event calendar and state variable information so that all possible states in the simulation could be executed recursively. The algorithm would assume first thread, put the rest of the threads on stack and continue. When the generation of this thread is complete, the algorithm restores the system from the save-state stack and continues the simulation for the second thread. A save-state counter is set up to count the number of saved states and to iterate through the save-state stack. Recall that all event notices whose execution order is uncertain are grouped in a set, called the nondeterministically ordered set (NOS).

In depth-first algorithm, a NOS counter is used to iterate through the NOS. The shortened version of this algorithm that shows the basic steps is illustrated in Figure 2.

Step 1: Initialize the save-state set and save-state counter. Initialize the simulation clock. Insert event notices into the event calendar.

Step 2: Determine the NOS. If there is only 1 next possible event notice to be executed next (no tie), go to Step 4.

Step 3: Initialize the variable to loop through the NOS. Set NOS counter to 1. Save the state of the simulation in the save-state stack and increment the save-state counter.

Step 4: Remove event notice from event calendar. Evaluate the execution edge condition. Determine possible new simulation clock time. Update simulation clock. Assign attributes to appropriate state variables. Evaluate state change and schedule further events.

Step 5: Stop simulation if any of these is true: stopping time is exceeded, stopping condition is met, or event calendar is empty. If the save-state stack is empty, then terminate the simulation.

Step 6: Increment the NOS counter. If there are still events from NOS that has not been executed, go to Step 4.

Step 7: Restore the last save-state values and decrement the save-state counter. Go to step 4.

Figure 2: Depth-First Algorithm

The depth-first algorithm is very useful if a complete analysis of all possible scenarios is needed in decision-making. Since all possible schedules are characterized and as long as input interval is not violated, users could make

fast strategic decisions based on the previously run output, and thus saving time and effort.

4.2 Breadth-First Algorithm

A breadth-first algorithm was developed and implemented by Agrawal (2003). In a breadth-first algorithm, all the active threads are evaluated simultaneously. The explosion of threads in this algorithm could be viewed as a tree diagram. The QDES simulation starts either with one parent node or a set of parent nodes. The simulation execution continues and spawns new threads from each of these parents, one by one, until all possible child nodes from each of these parents are explored. Before advancing to the next level down the tree structure, the breadth-first algorithm ensures that all sibling nodes are executed. Agrawal (2003) proposed a queue structure in his implementation to store the *breadth-first nodes*, a set consists of the event notice that are to be executed next, together with information such as the state variable and global events calendar. This queue is denoted as breadth-first node queue. The shortened version of the algorithm is shown in Figure 3.

Breadth-first algorithm proceeds and gives system snapshots of all event sequences through time. It keeps track of possible system state that is available and how it leads to that system state. It would also be useful to eliminate threads that are considered unimportant or unlikely to give good information. For example, elimination of thread could be added to the breadth-first algorithm if the number of thread explosion exceeds a certain limit .

However, depth-first algorithm has a speed advantage over breadth-first algorithm. This is due to the way breadth-first algorithm is structured.

Step 1: Initialize the simulation clock. Insert event notices into the event calendar. Create a breadth-first node for the first event notice to be executed and put it in the breadth-first node queue.

Step 2: Restore the system with values retrieved from the first breadth-first node. Remove this node from the queue.

Step 3: Evaluate the execution edge condition. Determine possible new simulation clock time. Update simulation clock. Assign attributes to appropriate state variables. Evaluate state change and schedule further events.

Step 4: Determine the NOS. Set NOS counter to 1. If the NOS counter ≤ |NOS|, evaluate when the next event is scheduled to occur. Go to step 5.

Step 5: Go to Step 5 if any of these is true: stopping time is exceeded, stopping condition is met, or event calendar is empty. Otherwise, copy the global event calendar to a temporary calendar. Remove the event notices from the temporary calendar and create a breadth-first node and insert it to the breadth-first node queue. Increment the NOS counter.

Step 6: If breadth-first node queue is empty, terminate the simulation. Otherwise, go to Step 2.

Figure 3: Breadth-First Algorithm

5 SCORING TECHNIQUES

Some of the threads that are generated by either of the above QDES algorithms may have a less likelihood to happen than other threads. As the complexity of the system increases, the number of threads generated using QDES will also increase. The thread generation could increase exponentially and causes the problem of extracting meaningful information from the output of the model. Thus, some scoring techniques were introduced to approximately rank the threads according to their likelihood of event execution sequences. Thread scores could be used in breadth-first algorithms to eliminate threads that have lower scores in relative to other threads and thereby reducing run time of the algorithm. Ingalls (1999) described three scoring techniques.

The midpoint ranking calculates the midpoint of each interval and ranks them accordingly. The second method is the multiple midpoint method, which also uses the midpoint of each interval. However, the resulting midpoint has taken into account the relative magnitude of all the midpoints and thus the event that is likely to execute first has a higher rank.

Let E_n, $n = 1, 2, \ldots, N$ ($N \geq 2$) denotes events with execution intervals $[L_n, U_n]$ that overlap. Let M_n be the midpoint of interval $[L_n, U_n]$ for $n = 1, 2, \ldots, N$. Let Rank(M_n) denotes the rank of M_n. The calculation of using multiple midpoint ranking is given as in the following equation:

$$Rank(M_n) = \frac{M_n - \min\limits_{1 \leq n \leq N}(L_n)}{\min\limits_{1 \leq n \leq N}(M_n) - \min\limits_{1 \leq n \leq N}(L_n)} \quad (1)$$

The uniform method assumes that each overlap interval sections have equal chances of being executed next and each interval follows a uniform distribution. The score is given to each interval, determined according to the probability that a given event would be executed first. The uniform distribution is chosen because it is easily programmed into the simulation and it has a closed form density function.

6 FUTURE RESEARCH

QDES creates a brand new arena in the field of discrete event simulation. In the area of the QDES algorithm, breadth-first and depth-first algorithms had been developed and are outlined in this paper. Both of these algorithms are tested with single-server queuing model. Depth-first algorithm was implemented in a PERT scheduling environment. It would be an interesting research to look into the scaling and expansion of this simulation methodology to solve real life industrial size problems.

With the increasing complexity in real-life problems, research could be embark on running the QDES algorithms on parallel processors, which includes research in the areas of problem representation, database storage of simulation output and scheduling criteria to yield reduction in run time.

Another interesting research is to look into the possibility of reasoning with probability distribution to produce an exact output statistics from QDES. The advance in this area would allow modelers to exactly represent each thread with its probability occurring, instead of sampling approach in the RDES models. If imposing statistical distribution to describe the threads is possible, modelers could also identify low probability threads that are not significant, which could contribute to the area of thread scoring.

There are three scoring techniques developed so far. All of these techniques have not been tested rigorously. It is possible to have more research work done in expanding thread scoring techniques.

The research on qualitative discrete event simulation is developing. In contrast to its quantitative counterpart, QDES is still in infancy.

REFERENCES

Agrawal, N. S. 2003. Breadth-First Algorithm for Qualitative Discrete Event Simulation. Stillwater, Oklahoma State University. Master of Science.

Allen, J. F. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 21(11): 832-843.

Cellier, F. E. 1991. Qualitative modeling and simulation: promise or illusion. In *Proceedings of the 1991 Winter Simulation Conference,* Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Ingalls, R. G. 1999. Qualitative simulation graph methodology and implementation. Austin, University of Texas. Doctor of Philosophy.

Ingalls, R. G., D. J. Morrice, et al. 2000. The implementation of temporal intervals in qualitative simulation graphs. ACM *Transactions on Modeling and Computer Simulation,* 10(3): 215-240.

Kuipers, B. 2001. Qualitative simulation. *Encyclopedia of Physical Science and Technology*. R. A. Meyers, Academic Press: 287-300.

Law, A. M. and W. D. Kelton. 2000. *Simulation Modeling and Analysis,* McGraw-Hill.

Schruben, L. and E. Yucesan. 1993. Modeling paradigms for discrete event simulation. O*perations Research Letter*, 13: 265-275.

Yucesan, E. and L. Schruben. 1992. Structureal and bahvioural equivalence of simulation models. *ACM Transactions on Modeling and Computer Simulation*, 21: 82-103.

Ziegler, B. P. and S. Chi. 1992. Symbolic discrete event system specification. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(6): 1428-1443.

AUTHOR BIOGRAPHIES

YEN-PING LEOW-SEHWAIL is currently a PhD student in the School of Industrial Engineering and Management at Oklahoma State University. She has a B.S. in Mathematical and Computer Science from the University of Adelaide, South Australia, Australia and a M.S. in Industrial Engineering and Management from Oklahoma State University, Stillwater, Oklahoma. She is a member of IIE, Alpha Pi Mu Industrial Engineering Honor Society, American Society for Quality and INFORMS. Her research interests include qualitative discrete event simulation, simulation methodology and analysis, probability and statistics. Her e-mail address is yenl@okstate.edu.

RICKI G. INGALLS is an Associate Professor and Director of the Center for Engineering Logistics and Distribution (CELDi) in the School of Industrial Engineering and Management at Oklahoma State University. He was also the Co-Editor of the *Proceedings of the 2004 Winter Simulation Conference*. He joined OSU in 2000 after 16 years in industry with Compaq, SEMATECH, General Electric, and Motorola. He has a B.S. in Mathematics from East Texas Baptist College (1982), a M.S. in Industrial Engineering from Texas A&M University (1984) and a Ph.D. in Management Science from the University of Texas at Austin (1999). His research interests include the supply chain design issues and the development and application of qualitative discrete event simulation. He is a member of IIE. His e-mail address is ricki.ingalls@okstate.edu.