MODELING FOR ANALYSIS

Lee Schruben

Industrial Engineering and Operations Research
University of California
Berkeley, CA 94720, USA

ABSTRACT

In simulation analysis research papers there is the implication that the data are generated by an actual simulation program. Other than tacitly presuming simulations can produce arbitrary, unlimited amounts of relatively cheap data, little further consideration is given to the details of the simulation model's design or to the computer program that would be supplying the data. The work required to produce the output used in a simulation analysis procedure is typically measured only coarsely by sample size: the number of replications, number of observations in a run, number of regenerative cycles, etc.

This paper explores some possibilities for exploiting the different analytical properties of different types of simulation models. Suggestions are made for how simulation analysis considered in the explicit context of discrete event simulation models, can create new opportunities for meaningful research. Modeling decisions can play a significant role in the performance of analytical procedures. How a simulation model is designed can enable, inhibit, or even invalidate analytical procedures and methodology research results. Modeling matters.

1 INTRODUCTION

This paper was influenced primarily by three discussions about simulation methodology research I have participated in during the past two-plus decades. Some of the conclusions from those discussions and most of the concerns raised then are still valid today.

1.1 1983: Listening to Titans

The first of these three conversations was almost 25 years ago between Alan Pritsker and Jack Kleijnen (my participation was pretty much limited to listening). Alan, Jack, and I were giving a simulation methodology workshop at Purdue where the relevance of steady-state output analysis to simulation practice was questioned. Jack noted that most non-academic simulation models are of systems that are inherently transient. Alan presented the opposing long-term economic arguments that are typically used to rationalize the study of queueing theory.

What I recall most from that discussion was the observation that much of the research in simulation analysis methodology was motivated by problems from the queueing literature, such as estimating steady-state means. Queueing theorists had posed most of the questions that were regarded as being academically interesting[1]. As a result, many academics regarded simulation as a middling "method of last resort" for solving these problems[2]. Simulation analysis research was widely regarded in academia as intellectually inferior to rigorously proving mathematical theorems about hypothetical queueing systems.

1.2 1995: A WSC panel with diverse opinions

The second of the three discussions that motivated this paper occurred over a decade later[3]. At the 1995 Winter Simulation Conference, I chaired a session on the relevance of simulation analysis research to simulation practice. That panel included two academic researchers and two simulation language developers: Peter Glynn, Jim Henriksen, Dennis Pegden, and Bruce Schmeiser. We addressed the following questions:

- What simulation analysis research results should be more widely used in practice?
- What are the incentives and barriers to implementation?
- What are some high-impact open simulation questions?
- What role should analysis methodology play in college-level simulation courses?
- What role should modeling play in college courses?
- What are some promising new directions for research?
- What sources of funding are available for simulation analysis research?
- What are some research successes stories? Why?

It was interesting to re-read our thoughts and predictions for the last decade.

What struck me most from that panel discussion was the strong disconnection between simulation analysis and simulation modeling. These seemed like two different fields! At the time, few methodology researchers had done much serious simulation modeling and few practitioners

had even tried to implement the newer results from simulation methodology research.

In light of the different viewpoints of the panelists, the discussion of the last question revealed an interesting social phenomenon. The success of academic analysis methodology researchers, to a large degree, was dependent on the judgment of simulation software developers. Research results that were automated and included as part of a commercial simulation language were considered successful. This was, and still is, the only way that the results of simulation methodology research can have broad impact.

In turn, simulation modeling and application papers submitted to the top academic journals were mostly refereed by analysis methodology researchers. Acceptance for publication was mostly determined by academics that valued mathematical rigor over modeling innovation or study impact. What was done, or why, was not nearly as important to publication as how it was done.

By the transitive property of this intellectual value chain, dominance and broad leadership in both simulation modeling and analysis came primarily from the commercially successful simulation language developers.

1.3 2005: A WSC panel with consensus

The most recent of these three discussions on simulation modeling and analysis was a panel at the 2005 Winter Simulation Conference Chaired by Enver Yücesan titled "Analysis Methodology: Are we done?"[4]. This time, the other panelist were all simulation analysis methodology researchers: Sigrún Andradóttir, David Goldsman, and Bruce Schmeiser. The proposition put before the panel was that all the important problems in simulation analysis methodology have effectively been solved. Since I have worked on these problems for nearly 30 years, but am not ready to retire, I hoped the answer would be "We are *nearly* done. Good work!"

2 SIMULATION MODELING AND ANALYSIS

The historical divisions between simulation analysis and simulation modeling I first saw over two decades ago have faded, but, I believe, still limit simulation's development. Most university courses and textbooks focus mainly on either simulation modeling or on simulation analysis. Undergraduate simulation courses are primarily language-based, where the modeling approach and analysis are determined by the software used. On the other hand, many advanced PhD simulation courses are essentially simulation model free. ("Graduate students should be able learn a simulation language on their own.")

At major conferences, simulation modeling and analysis tend to have distinct, conflicting tracks with little overlap in the participants. There are even distinct conferences focusing on one aspect of simulation or the other[5]. Like this one.

In the remainder of this paper, I will try to persuade simulation analysis methodology researchers that simulation modeling matters.

3 MODELING MATTERS

I will use the language of Event Relationship Graphs (ERGs) to specify alternative simulation models of discrete event system dynamics.[6] ERGs are concise abstractions of causality in general discrete event system models. They are independent of any particular simulation language or modeling "world view". Each node in an ERG simulation model represents the state changes that might happen when an event occurs. Directional arcs in an ERG explicitly specify at what times and under what conditions an event might cause another event.

Alternative ways to express discrete event dynamics include generalized semi-Markov processes (GSMPs),[7] Stochastic Timed Petri Nets (STPNs)[8] and more mathematically rigorous Simulation Graphs.[9] However, the notational burdens of these other paradigms are significant and not necessary here, and they can be unnecessarily limiting.

A simulation model for the number of jobs in a G/G/s system, *Q(t)*, is shown in Figure 1 I will refer to this model as ERG_1.

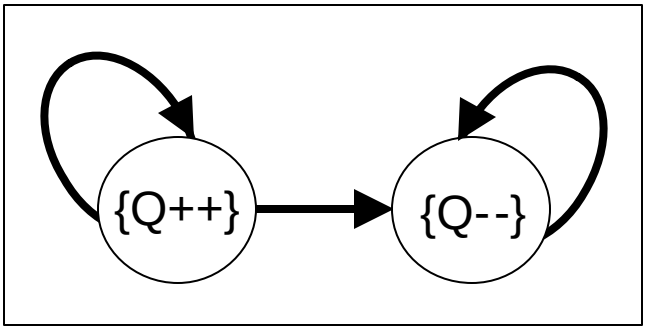

Figure 1: ERG_1 for the Q(t) process of a G/G/s system

Figure 1 uses a simplified ERG notation where bold arcs represent time delays between event executions and thin arcs represent instantaneous, conditional, event sequences.[10] In Figure 1 the event node on the left increments *Q(t)* when a job enters the system and the event node on the right decrements *Q(t)* when a job leaves. In this simple model, the two events nodes are labeled with their state changes.

The most important features of simulation model ERG_1 are that it is small and fast. Its state memory footprint is a single static integer, Q. More importantly, its execution speed is insensitive to queue congestion; the model runs at the same speed if the system has 10 jobs or

10 million jobs – only the *value* Q is different. However, the execution speed of ERG_1 is sensitive to the number of busy servers.

The simulation model, ERG_1, is appropriate for simulating the *Q(t)* process with ultra-heavy (or nonstationary) queue traffic. This includes many interesting real service and communications systems.

3.1 An alternative (equivalent?) model

Many simulation texts present ERG_2 in Figure 2 as an alternative simulation model for a G/G/s queue[11]. Figure 2 does not use ERG shorthand, but fully specifies the relationships among three sets of difference equations in this discrete event simulation model.

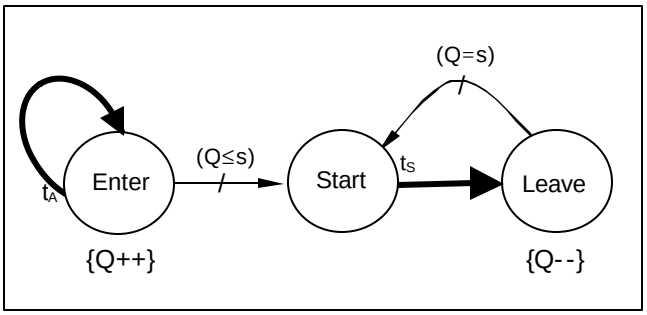

Figure 2: ERG_2 for the Q(t) process of a G/G/s system

It is generally accepted that the Start service event in ERG_2 is worse than superfluous. It does not change *Q(t)*, but requires the scheduling and execution of a another event for each job in a run. ERG_1 and ERG_2 are *behaviorally* equivalent for *Q(t)*, but ERG_2 runs slower. However, ERG_1 and ERG_2 are not the same simulation models. They are *structurally* different and this is important[12].

ERG_2 has some distinct *analytical* advantages over ERG_1. An obvious analytical advantage for ERG_2 is that it has a STPN dual (where ERG arcs become Petri net nodes). ERG_1 does not.[13] This is because ERG_2 has no cycles with two adjacent arcs of the same type. The considerable mathematical machinery developed for Petri net analysis is applicable to ERG_2, but not to ERG_1. There is a more compelling analytical advantage to ERG_2 for studying job delays.

3.2 Indirect estimation of job delays

An important weakness in simulation model ERG_1 is that it does not simulate the job delay process, $\{D_i\}$. Simulating job delays requires much more work than simulating *Q(t)*. This is because information on job delays must be recorded and stored for an interval of time, while queue sizes can be observed instantaneously at event times. There is an extensive discussion of job delay estimation in the previously cited book by Peter Haas, where a sophisticated scheme for recording job delays is developed for k-bounded Petri nets.

ERG_1 is easily modified to simulate job delays using a conventional priority queue abstract data type with operations INSERT (here called PUT) and DELETEMIN (here called GET) [14]. These operations can record the arrival and waiting times of jobs. The PUT and GET are implemented using functions that always return a value of 1 so the *Q(t)* process is identical to that for ERG_1. The resulting model, ERG_3, simulates both *Q(t)* and D_i and is shown in Figure 3. To observe job delays, the arrival event function is replaced by Q+PUT and the departure event by Q-GET, with the integer Q still representing the number of jobs in the system.

The execution speed of simulation model ERG_3 is, unfortunately, now sensitive to the number of jobs in the system as well as the number of busy servers. Implementations of the models ERG_1 and ERG_3 in both Arena and Sigma show that ERG_3 can take several orders of magnitude longer to simulate the same numbers of jobs as ERG_1. The comparisons were for a single server with moderately congested queues. (At most 100 jobs were allowed concurrently in the system in the student version of Arena.) These observed differences in run times are equivalent in a 90% variance reduction in the estimator for E[*Q*] using simulation model ERG_1 as compared to the estimator of E[*D*] using ERG_3. The difference in analytical efficiency between these two simulation models is probably more pronounced in practice where a large number of jobs are typically run and discarded to warm up each simulation run.

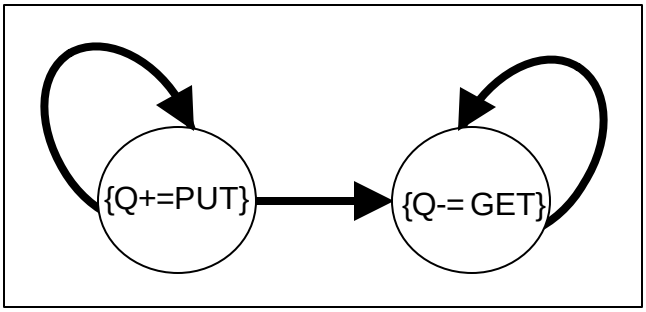

Figure 3: ERG_3 for Q(t) and $\{D_i\}$ of a G/G/s system

If one is interested only in estimating the mean job delay, E[*D*], then it would appear better to run the much faster simulation, ERG_1, to estimate E[*Q*] directly and then apply Little's Law to estimate E[*D*]. However, the literature on the variance reduction technique of indirect estimation (IE) recommends just the opposite[15,16.] The recommendation in the literature is to run the slower simulation, ERG_3, to estimate E[*D*] directly and indirectly estimate E[*Q*] via Little's Law. This recommendation is based on the variance of the average delay being

less than the variance of the average queue size (exploiting the fact that the average service time is known) and the incorrect assumption that an equivalent amount of computation is required to simulate {*Di*} and *Q(t)*.

The reported variance reductions of IE are around 20%. Furthermore, the variance advantage of the proposed IE estimator disappears for high congestion queues. On the other hand, the efficiency of simulation model ERG_1 compared to ERG_3 increases as congestion increases.

It seems likely that the advice for using IE given in the simulation research literature is incorrect, particularly when studying congested queues. This is a simple example of where a consensus recommendation from rigorous simulation output analysis research, taken out of the simulation modeling context, is probably wrong most of the time.

3.3 Direct estimation of job delay probabilities

Assume a simulation study is interested in more than the mean job delay. It is probably more realistic, particularly when simulating service systems, to be interested in estimating the probability that a job will be delayed longer than some service performance standard. For a particular performance threshold, *d*, the problem is to estimate the value of the distribution function of *D* at *d*,

$$q(d) = F_D(d) = \text{Prob}\{D_i \leq d\}.$$

Using the slower model, ERG_3, to simulate job delays is the customary approach. However, we note that the typical estimator of *q(d)* uses the indicator, $I_{[Di \leq d]}$ (equal to one when $D_i \leq d$ and zero otherwise)

$$\hat{q}(d) = \frac{1}{n}\sum_{i=1}^{n} I_{[D_i \leq d]}.$$

Here n is the number of simulated jobs, perhaps after a warm up period. Since the process {D_i} is first converted to a sequence of indicator function values to compute $\hat{q}(d)$, we might save work by designing our simulation model to generate $I_{[Di \leq d]}$ directly.

If we assume that jobs are served in the order they arrive, then simulating the delay indicator for *d* is easily done by adding another "superfluous" event to simulation model ERG_2. This new event simply counts the number of jobs that arrived more than *d* time units earlier than the current *Start* event. This simulation model, ERG_4, is in Figure 4.

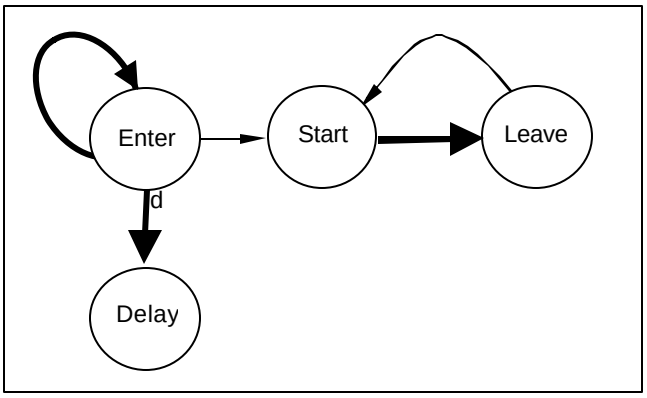

Figure 4: ERG_4, implicit estimation of job delay.

At the time of each job *Start* event, if the count of *Delay* events is greater than the count of *Start* events, then we know that the job starting service has spent less than *d* time waiting in line. At each *Start* event, we also know each job's service time so, the value of the indicator, $I_{[Di \leq d]}$, is defined at the instant this event occurs – job arrival times do not need to be stored. This method for simulating points on the job delay probability distribution can be easily adapted to equivalent Petri net simulation models.

At first glance, it looks like simulation ERG_4 is merely using the master list of scheduled events to effectively store the arrival times of each job. However, the Delay event in ERG_4 is *implicit*: it does not schedule any other events and does not change *Q*(*t*) or {D_i}; therefore, it is not essential that this event be either scheduled or executed.

This leads to the second analytical advantage of ERG_2 over ERG_1. If this implicit job delay counting event were added to ERG_1, there would be an error due to job overtaking in multiple server queues. Overtaking occurs when jobs depart in a different order than they arrived. One does not have to read past the first page in some queueing texts to see the complexity overtaking adds to queueing analysis.[17] For FIFO queues, there is no overtaking between the *Delay* and *Start* events while a job waits in line.

Several extensions of ERG_4 are possible. If more points on the probability distribution function for *D* are desired, then a parameterized *set* of implicit job delay events, D, can be defined for a larger number of different arguments {d_i}.

If the queueing discipline is not first-come-first-served, then a "kanban" control limiting the number of concurrent implicit job delay events recorded to 1 for each element in D will avoid the error due to overtaking while a job is in line. This will work provided the job order in the queue does not change except at event times. This includes most common queue disciplines, but does not cover external queue reordering, say due to the exogenous changing of some of the job completion due dates.

This kanban control is related to transaction tagging studied for queueing systems.[18] There it was seen that little information about the mean delay is lost by not recording all job delays because of the high positive autocorrelations in the time series $\{D_i\}$, particularly when the queue is large. Whether this extends to different properties of D was not studied.

A potentially useful extension of ERG_4, with kanban controls, is to allow the delay times, $\{d_i\}$ for the set D of implicit delay events to be random variables. If these indices are exponentially distributed with different means, $\{\lambda_i\}$ then it may be possible to have the simulation model generate useful estimates of the Laplace transform of the delay probability distributions implicitly. It is also intriguing to consider state-dependent importance sampling or tagging for larger values of λ.

4 MODEL COMPLEXITY

The concept of computational complexity is intended to measure how the execution time for a computer algorithm slows as the problem being solved gets larger. This is a well-developed idea in many areas of computer science and operations research. A practical notion of complexity for simulation models remains to be developed.[19] What follows is a proposal in that direction, specific for queueing type simulations.

A simulation model's complexity should reflect how the program execution time slows as the system being modeled grows. For discrete event simulation codes, a major portion of the computing effort is spent in managing future events list(s) and updating entity data structures.

4.1 Resident and transient entity world views

For queueing simulations there may be events that must be scheduled and executed as well as entity records updated for every busy server and, depending on the model, perhaps for every concurrent job. We might usefully distinguish the work needed to simulate an output process as dependent on the number of concurrent jobs (generically, *transient* entities) or the number of busy servers (*resident* entities).

The distinction between resident and transient entities in simulation modeling is not new.[20] Simulation software design often explicitly distinguishes these types of system entities. The designation of a system entity as resident or transient is a modeling decision – considered more art than science – but it can greatly affect the efficiency of an analytical procedure. Table 1 gives some of the terminology.

Technology	Resident	Transient
Queueing Theory	Servers	Customers
Petri Nets	Token counts	(implicit)
Siman/Arena	Resource	Entity
Simscript	Permanent	Temporary
GPSS	Facility/Storage	Transactions
Memory model	Static	Dynamic

Table 1: some popular designations of physical entities

If the model's run time needed to generate and record output series {Y} is directly proportional to the average number of concurrent transient entities, its complexity might be denoted as $O(T:Y)$. If a model's run time is proportional to the number of busy resident entities, its computation complexity would be denoted $O(R:Y)$. There is empirical evidence and structural analysis that indicate the run time for ERG_3 is $O(\ln(R+T))$, where most of the work is spent inserting events on the future events list.

This suggests a meaningful model taxonomy for discrete event simulations of queueing systems that reflects the amount of work needed to simulate (or observe) different processes.[21] Simulation models with run times that are predominantly a function of the number of resident entities are called "resident entity" models (or resource-driven (RD) simulations). Simulation models with speeds that are very sensitive to job congestion are called "transient entity" models (or job-driven (JD) simulations).

4.2 Implementation

The observations in this paper apply to considerably more realistic simulations than the simple models presented here. The simulation model, ERG_1, can be enriched to simulate queueing networks of *any size*, processing *any number* of different classes of jobs (each having different routes, timing, and priorities) being served by *any number* of parallel server types having multiple modes of failure with batched arrivals and batched services possible at each station. This can be accomplished by parametrically enriching the event functions and adding only a single attributed arc to the graph, the rest of the graph structure stays the same. Actual large-scale RD and JD simulations of a semiconductor factory, which were extensions of ERG_1 and ERG_3, showed orders of magnitude faster run times than current simulation models.[22]

The models providing the examples in this paper are independent of a model "world view" or particular simulation language. ERGs can be directly mapped to STPNs or cycle diagrams that are typically executed using an "Activity Scanning" (AS) simulation engine. Transaction flow models often simulated using Process Interaction (PI) simulation engines are also easily modeled as ERGs using pending arcs. It is a common misconception that ERGs represent only an Event Scheduling (ES) world view. From

a PI viewpoint, an ERG graph is composed of the set of all directed paths emanating from exogenous events (ERG nodes with only self-scheduling arcs). From an AS viewpoint, an ERG graph is composed of the set of directed cycles.

The ease of implementing a simulation model ERG in a particular simulation language depends greatly on the openness of the software. In a "black box" transaction flow language like GPSS or Arena, where the source code is inaccessible, something like the following trick might be used. To implement, say, ERG_2 in Siman (Arena), CREATE jobs at arrival times, most of which are simply immediately COUNTed and DISPOSEd. However, the first job that initiates a busy period for a server *will become that server*. This server entity loops through a DELAY block for each new job's service time and is DISPOSEd when the busy period ends for the server becomes idle. The number of transactions in the simulation never exceeds twice the number of busy servers. This simulation's computational complexity is a function only of R. Its execution speed is insensitive to increases in congestion (T), so it will be efficient for simulating highly congested systems.

5 SOME RESEARCH OPPORTUNITIES

There are many situations where simulation analysis methodology research can benefit by considering different simulation model contexts. Modeling research can also be advanced by considering the analytical procedures for which a model is to supply data. A few examples are outlined in this section.

5.1 Determining an appropriate level of model detail.

It is recognized that simulation modeling problems as fundamental as determining an appropriate level of model detail cannot be meaningfully addressed irrespective of the problems the model will be used to solve. Answers to the vague question "How much detail is appropriate?" range from "as little as necessary" to "the more detail the better".

Unless the simulation is purely for animation, and many are, the amount of model detail should depend on the data being simulated as well as the analysis procedures used to study that data. As we see from the examples above, the analysis methodologies used can have an impact on the appropriate level of detail in a simulation model. Furthermore, the level of detail that is desired will probably change as a simulation analysis procedure progresses.

5.2 Sequential models for sequential procedures

Simulation analysis procedures typically do not consider the possibility of using models with different levels of detail during different phases of an experiment. Adding this dimension to the study of variance reduction techniques, ranking and selection procedures, and response optimization algorithms greatly enriches the degrees of freedom in the design of these methodologies and provides some unexplored opportunities for research on the synergy between simulation modeling and analysis.

The detail in simulation model, ERG_4, can be meaningfully enriched *while it is running* by increasing the cardinality of Δ or, in some cases, by increasing the number of kanbans tagging jobs. For FIFO queues, more tagged jobs result in a slower but more informative output.

For more general queueing disciplines, it is possible to move more or less continuously from the fast resident entity model ERG_1 to the slowest transient entity model ERG_3 by marking more jobs with PUT/GET objects whenever more detail on job delays is desired. This can be done *while the simulation model is running*. The PUT and GET methods return the value of 1 when detailed job delays are not desired, greatly speeding the simulation execution during the initial, or less interesting, phases of experimentation.

While running a sequential or multiple phase optimization or ranking and selection experiment, it makes sense to consider faster, but less detailed, models during the early phases of the procedure. The possibility of run time tradeoffs between model information and execution speed adds a new, and potentially high impact, dimension to the design of such procedures. For instance, during the first phase of a two-phase selection procedure an O(f(R)) model might be more effective, while an O(g(T)) might be better suited during the second phase. A two-phase procedure becomes a 4-phase procedure when only two different levels of model detail are considered. In different phases of the procedure, different levels of model detail are probably appropriate for different candidate systems, depending on how likely they are to be among the eventual winners. The options grow as powers of the numbers of possible models.

While considering the simulation model context generally enriches ranking and selection procedures, it can also lead to significant simplification. Parametric ERGs can model k different networks *simultaneously*. This model design essentially eliminates the cost of model switching when comparing models of different systems.[23] This idea extends significantly in the design of general time dilation simulation models and selection experiments where the number of systems considered can also be decreased or increased during a single simulation run.[24]

In using simulations to estimate rare events, such as large job delay times, it makes sense to run a faster pure resident-entity model when $Q(t)$ is low and then increase the detail by marking more jobs as the system becomes busier and observing $\{D_i\}$ is more likely to pay off. This way a simulation like ERG_1 is used to speed through uninteresting periods – enriching the model toward simulation ERG_3 as the system state becomes more interesting.

5.3 Causality, mathematical programming, and IPA

To demonstrate how modeling matters when estimating response gradients, consider a queue having state-dependent service times. Here the speed of the server might change with every event. A straightforward simulation model of this system is ERG_5 in Figure 5, which simply reschedules the departure of the job currently being processed when a new job arrives by using a (dashed) event-cancelling arc.

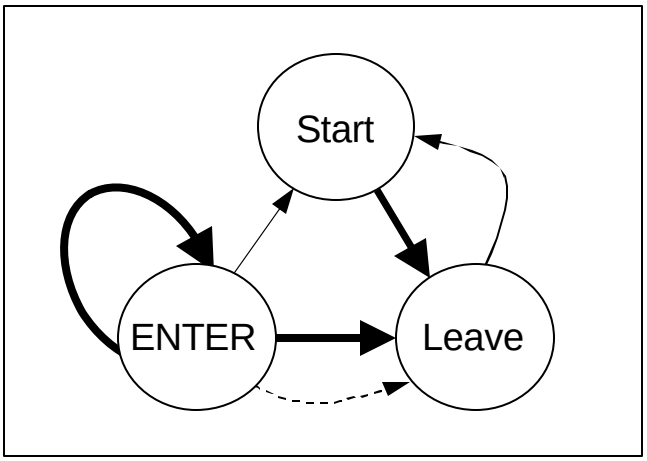

Figure 5: ERG_5, state-dependent queue with rescheduling

State dependent service can also be modeled without using an event-cancelling arc.[25] The simulation model has the same graph structure as ERG_5 without the cancelling arc, but with different arc conditions and richer event functions. In fact, event-cancelling arcs are theoretically never strictly necessary in a discrete event simulation model, but they are without doubt convenient. There is a major *analytical* difference between a simulation model with event-cancelling and one without it. With an event-cancelling arc, event *causality* is no longer explicit in the simulation model ERG. This prevents us from using current methodology to estimate response gradients using the values of dual variables from the ERG sample path representation as a linear program.[26]

6 CONCLUSIONS

Important problems where simulation modeling and simulation analysis researchers might work together have been suggested. Research on the analysis and design of simulation experiments that explicitly exploit the structure of realistic simulation models can have greater impact on the practice of simulation. As more influential leaders among simulation methodology researchers emerge, perhaps simulation language software companies can be persuaded to allow more open access to their codes to implement new analysis methodologies. Practitioners will then be more likely to appreciate the relevance of simulation analysis methodology research.

Simulation methodology researchers should also consider illustrating new procedures with models that are more sophisticated and realistic than ones inherited from queueing theory. Simulations can model the huge, complicated, non-stationary, non-homogeneous processes found in the real world, where simulation is the most practical approach, not the method of last resort. (Hypocritically, all the examples in this paper are from queueing theory.) The national conferences of most engineering and scientific disciplines predominantly feature simulations, in contrast with the few simulation tracks at operations research conferences.

Simulation analysis methodologies that explicitly exploit the realism, different levels of abstraction, control of uncertainty and time, and explicit causality in simulation models can produce much more powerful results. Simulation modeling *for* analysis opens new opportunities for meaningful research.

Analysis methodology: Are we done? No, we are only beginning!

ACKNOWLEDGMENTS

Most of the ideas presented here come from discussions with my students and my wife. However, the author accepts full responsibility for errors and offensive comments. Support from the National Science Foundation through grant DMI-0323765, Foundations for Discrete Event Stochastic Systems, to UC Berkeley is gratefully acknowledged.

REFERENCES

1. Fox, Bennett L..(1978) "Estimation and Simulation", Management. Science, vol. 24, pp 860-861.

2. Wagner, Harvey M.(1975), Principles of Operations Research (2nd Ed.), Sec 2.1., When all else fails...", Prentice Hall..

Proc. of the 1995 Winter Simulation Conference, "The Interface Between Simulation Output Analysis Research and Practice," Alexandria, VA, December 3-6, p. 346.

4. *Proc. of the 2005 Winter Simulation Conference*, "Analysis Methodology: are we done?" Orlando FL, pp. 790-796,

5. See, for example, programs for the Simulation Solutions Conferences, www.simsol.org.

6. Schruben, L.W., (1983) "Simulation Modeling with Event Graphs," Comm. of the ACM, 26.11, pp. 957-63.

7. Glynn, Peter W. (1989) "A GSMP formalism for discrete event systems", Proceedings of the IEEE, 77.1 pp. 14-23.

8. Haas, Peter J., Stochastic Petri Nets: Modeling Stability, Simulation, Chapter 8, Springer, 2002.

9. Törn, Aimo. A. (1981) "Simulation Graphs: a General Tool for Modeling Simulation Designs", Simulation, 37.6, pp. 187-194. and (completely different with same name) Schruben, L.W. and E. Yücesan, (1988) "Simulation Graphs," Proc. 1988 Winter Simulation Conference, San Diego, CA., December 12-14, pp. 504-50.

10. Law, A. M. (2006), Simulation Modeling and Analysis 4th ed., McGraw-Hill, pp 58, 59.

11. Seila, Andrew, V. Ceric, and P. Tadikamalla, (2003), Applied Simulation Modeling, Duxbury Press.,

12. Yucesan, E. and L.W. Schruben, (1992) "Structural and Behavioral Equivalence of Simulation Models," ACM Trans. on Modeling and Computer Sim, 2.1, pp. 82-103

13. Schruben, L.W. and E. Yucesan, (1994) "Transforming Petri Nets into Event Graph Models," Proc. 1994 Winter Simulation Conference, pp. 560-565.

14. Aho, A. V., J. E. Hopcroft, and Ullman, J. D. (1985) Data Structures and Algorithms, Addison-Wesley, p. 135-138

15. Carson, John S. and Averill M. Law, (1980) Conservation Equations and Variance Reduction in Queueing Simulations" Operations Research, 28, pp. 535-526.

16. Glynn, Peter W., and Ward Whitt (1989) "Indirect Estimation via $L=\lambda W$", Operations Research, 37.1, pp. 82-102.

17. Baccelli, Francois, Bremaud, Pierre, (2003) Elements of Queueing Theory: Palm Martingale Calculus and Stochastic Recurrences, Spring, pg. 1.

18. Schruben, L.W. and E. Yücesan, (1988) "Transaction Tagging in Highly Congested Queueing Simulations," Queueing Systems: Theory and Applications, 3, pp. 257-264.

19. Jacobson, Sheldon H. and E. Yücesan (1999), "On the Complexity of Verifying Structural Properties of Discrete Event Simulation Models", *Operations Research*, Vol. 47.3 pp. 476-481.

20. Markowitz, Harry M. (1979), "Simscript: past, present, and some thoughts about the future", Chpt. 3 in Current Issues in Computer Simulation, Nabil Adam and Ali Dogramaci, eds. Academic Press. pp. 27-60.

21. Roeder, T. (2003) "An Information Taxonomy for Discrete Event Simulations", PhD. Thesis, Dept. of IEOR, UC Berkeley.

22. Schruben, L. and T. Roeder. (2003). "Fast Simulations of Large-Scale Highly-Congested Systems." Simulation: Transactions, 79.3, pp. 1-11.

23. Hong, L. J. and B. L. Nelson. (2005) "The tradeoff between sampling and switching: New sequential procedures for indifference-zone selection". *IIE Transactions*, 37:623-634.

24. Hyden, Paul, (2002) "Time dilation: Decreasing time to decision with discrete-event simulation", PhD thesis, School of Operations Research and Industrial Engineering, Cornell.

25. Ignalls, Ricki G., Morrice, D. J., and Whinston, S. B. (1996) "Eliminating canceling edges from the simulation graph modeling methodology", Proc. Winter Simulation Conference, pp. 825-832.

26. Chan, Wai Kin (Victor), (2005) "Mathematical Programming representations of discrete event system dynamics", PhD Thesis, Dept. of Industrial Engineering and Operations Research, UC Berkeley.

AUTHOR BIOGRAPHY

LEE SCHRUBEN is a Professor and past Chairman in the Department of Industrial Engineering and Operations Research at the University of California, Berkeley, CA, USA. Prior to coming to Berkeley he was on the faculty of the School of Operations Research and Industrial Engineering at Cornell University, where he held the Andrew Schultz Chair in Engineering. Schruben has degrees from Cornell, the University of North Carolina, and Yale and is broadly interested in simulation modeling and analysis methodologies, most recently as related to the production and distribution of biopharmaceuticals.