Proceedings of the 2001 Winter Simulation Conference
B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, eds.

VARIOUS WAYS ACADEMICS TEACH SIMULATION: ARE THEY ALL APPROPRIATE?

(PANEL ON EDUCATION IN SIMULATION)

Tayfur Altiok (Moderator)

Department of Industrial Engineering
Rutgers University
Piscataway, NJ 08854, U.S.A.

W. David Kelton

Department of Management Science
and Information Systems
The Pennsylvania State University
University Park, PA 16802-1913, U.S.A.

Pierre L'Ecuyer

Departement d'informatique
et de Recherche Operationnelle (I.R.O.)
Universite de Montreal, C.P. 6128
Centre-Ville, Montreal, H3C 3J7, CANADA

Barry L. Nelson

Department of Industrial Engineering
and Management Sciences
Northwestern University
Evanston, IL 60208-3119, U.S.A.

Bruce W. Schmeiser

School of Industrial Engineering
Purdue University
1287 Grissom Hall
West Lafayette, IN 47907, U.S.A.

Thomas J. Schriber

Business School
University of Michigan
701 Tappan Street
Ann Arbor, MI 48109-1234, U.S.A.

Lee W. Schruben

Department of Industrial Engineering
and Operations Research
University of California
Berkeley, CA 94720, U.S.A.

James R. Wilson

Department of Industrial Engineering
North Carolina State University
Raleigh, NC 27695-7906, U.S.A.

ABSTRACT

This panel discusses goals and educational strategies for teaching simulation in academia. Clearly, there is considerable material to cover in a single course or a sequence thereof in, say, an undergraduate program. The issue is how to motivate and empower students to analyze complex problems correctly and to prevent the pitfall of misusing the concept.

1 INTRODUCTION

Computer simulation modeling has been taught in engineering schools, business schools and computer science departments in academic institutions worldwide for decades. This discipline teaches students how to develop computer models of systems, both real-life or on the drawing board, depending on the area. It is comprised of theoretical aspects (e.g., statistics and time series) as well as practical aspects (programming). In view of the complexity of large-scale problems that engineers, scientists and business managers face today, simulation modeling has grown into possibly the only practical approach of analyzing them. Indeed, the simulation methodology is widely accepted and broadly applied in many application domains. Consequently, a tremendous growth has been achieved over the past two decades in both theoretical and technological facets of simulation, both edu-

cational and commercial. Simulation systems are increasingly evolving from general-purpose languages to direct-engagement, visual programming environments supported by powerful graphics tools.

Nowadays, we are faced in academia with the baffling decision of what to cover in a course or a series of courses in simulation. The main problem is how to select varied simulation-related topics into a one-semester course in view of the knowledge the educator wishes to impart to the students. This problem is complicated by the constraints imposed on the educator who must factor in the students' skill set.

Fundamentally, there are two basic approaches to teaching simulation: one emphasizing theoretical aspects and the other emphasizing hands-on aspects. Accordingly, a simulation course may include the following basic topics:

- Discrete-event simulation
- Random number and variate generation
- Input and output analyses
- Verification and validation

In the theoretical approach, these topics, which are largely based on probability and statistics, can be comfortably taught in some depth either without regard to any simulation tool, or by using an educational software tool for demonstration purposes. This approach teaches students the proper simulation methodology (the "right" things to do in a simulation project) as well as theoretical underpinnings. However, it would not teach them the hands-on skills to simulate a complex system with a strict deadline, which is what the marketplace mandates. Furthermore, the students may suffer from modeling-skill deficiencies, leading them to produce invalid or oversimplified models.

The hands-on approach would briefly review the fundamentals, and spend most of the time in the lab, practicing with a (typically commercial) simulation tool, to teach programming details. In this approach, students will definitely acquire extensive hands-on knowledge of such a tool and most likely the capability of modeling complex problems under strict deadlines. However, it is fraught with serious dangers, including failure to verify and validate the model, misinterpretation of results, too much detail in the model and last but not least GIGO.

An important question facing this panel is how to mix and match the two approaches above to the goals of the course and the capabilities of the attending students. Furthermore, can such mixtures be spiced with practices such as having groups of students modeling real-life systems involving data collection and validation, and mentored by individuals working in the corresponding application domains.

This panel consists of educators with many years of experience in preaching, teaching and practicing simulation. It is expected that the panel discussion will touch upon many relevant and controversial issues. Below are the position statements of the member panelists.

2 PANEL MEMBERS' STATEMENTS

2.1 Undergraduates Should Not Take a Course in Simulation (B. L. Nelson)

This is my thesis: Undergraduate industrial engineers and management scientists should not take a course in computer simulation. Here is a quick quiz that illustrates why: Is the following a "simulation problem?"

- How many parking spaces should there be in the parking garage of a large shopping mall?

I claim that the answer depends on a number of factors beyond the problem statement itself, including:

- How quickly do we need an answer?
- Are there any data available?
- What performance measures will we use to evaluate a solution?
- How accurate do we need to be (or, stated differently, how will the answer be used)?

For instance, suppose that the mall has yet to be built, so there are no data available on how much customer traffic the mall will generate beyond the marketing department's estimates. However, studies at similar malls have given us some idea how long people spend shopping. The mall developers tell us they want the chance that there are not enough spaces for everyone to be quite small, but the type of structure we have in mind comes in floors of 200 spaces, so whatever our answer is it will be in multiples of 200. Finally, we need the estimate quickly to roll it into our financing proposal.

In this situation, a relatively simple M/G/∞ queueing model may be adequate for the rough-cut analysis needed to size the structure.

On the other hand, suppose we are replacing surface parking at an existing mall with a parking garage. Since our traffic volume is heavy, we are worried about how long it will take for drivers to find open spaces in a large parking structure, not just whether there are enough spaces. Therefore, we want to consider different designs for the facility as well as different sizes. Maybe we also want to locate spaces especially for patrons with disabilities, with compact cars, etc. We have lots of data on this mall and others we own. Further, since we have surface parking now, we are willing to invest substantial time into getting this garage design right.

In this case, simulation might be the only tool capable of the analysis we desire.

Clearly the same problem may require very different solutions, and this is obvious to most of us. Unfortunately, when students take distinct course in stochastic processes (analytical or numerical solution of certain well-studied classes of models) and simulation, their natural inclination is

to try to map problems one-to-one with solutions. They end up with a thought process something like this: Parking garage problems require Markovian queuing models, but evaluating the use of cross-trained workers requires simulation, *because we solved a parking garage problem in the queueing class, and a cross-training problem in the simulation class.*

I am a proponent of generic courses in stochastic modeling and analysis, in which mathematical, numerical and simulation solution techniques all appear. I have been teaching a two-quarter (20-week) sequence in this way for over six years, and I am convinced that there are at two features that are critical to making it work:

1. For *every* stochastic modeling problem, start working on it by thinking about how to simulate it. Simulation (inputs, events, states, etc.) provides the formulation language, much like the decision-variable, objective-function and constraint concepts do for optimization. Simulation is also intuitive. We then we teach students to recognize those situations in which a mathematical or numerical solution is possible or appropriate.
2. When a large-scale simulation is required, force students to do a rough-cut model prior to simulating. (I am pretty sure I stole this idea from Lee Schruben.) Sometimes the rough-cut model is just plugging in mean values for all the stochastic stuff, or deriving best-case and worst-case bounds. More often it involves using some sort of simplified model, such as an M/M-type queue. This allows students to understand that both approaches apply to the same types of problems. They also see that the numbers that come out of the simulation typically do not match the rough-cut model---demonstrating that there is a reason for simulation---but they also see that the best solution, as determined by the rough-cut model, is often identical to the one indicated by the far more detailed simulation.

2.2 Teaching Modeling and Analysis Together (W. D. Kelton)

When teaching a simulation course, there is usually the issue of how to divide the available time between modeling topics and analysis topics. Seldom is there time to do a complete job with both in a single course.

By "modeling topics" I mean how to construct a model of a static or dynamic system, issues about level of detail, and how strong the simplifying assumptions can be before model validity starts to come under question. Model validation and verification are also taught under this heading as well. Of course, there is the issue as well of whether to use a high-level modeling package or a lower-level programming language (but I won't get into this debate).

By "analysis topics" I mean the methodological underpinnings of simulation, which usually pertain to probability and statistics. These subjects include specifying input distributions and processes, methods for generating random numbers and processes, statistical analysis of output, variance reduction, gradient estimation, experimental design, and perhaps optimizing simulation models.

Now it's of course impossible to imagine a first course in simulation (at either the undergraduate or beginning-graduate level) not to cover modeling topics. The issue, rather, is how much of the analysis topics should be covered.

Sometimes the answer is none. This is, in my opinion, unfortunate since students walk away thinking this is all about computer programming with some piece of software or in some language. They may be able to build nice models, but they won't know what to do with them. And they won't be able to react to challenging or difficult problems if they've had no exposure to the underpinnings.

The "none" answer is not only unfortunate, but also unnecessary. In fact, I feel that it is possible and helpful to teach at least the basics of analysis alongside the modeling in an integrated fashion, and not treat analysis as a separate set of topics included (maybe) at the end of the course. This can be done almost from the very beginning, even if one chooses, as I do, to start the first course with a simulation done "by hand." It can be redone a couple of times to illustrate the variance in the results, or at least described without taking the class time to do several by hand. Then when we move to computer-based simulations, it is easy to replicate across different configurations and start doing valid two-sample or paired statistical tests. As richer models are developed, the more advanced analysis topics like variance reduction, gradient estimation, and optimization can be introduced in the context of these richer models.

I like to call this method of presentation "tutorial style," and my ideal model for it is *AMPL: A Modeling Language for Mathematical Programming* by Fourer, Gay, and Kernighan (1999). Though not on simulation, this book skillfully crafts a sequence of increasingly complex examples that simultaneously teach how to model with the AMPL system, how to interpret the results, and to a large extent what's available in AMPL. I feel that we should be able to do very much the same kind of thing when teaching simulation.

2.3 How Not to Teach a Simulation Course (L. W. Schruben)

"I wish I didn't know now what I didn't know then" – Bob Seger.

I am going to share some bad ideas I have had for teaching simulation courses. These are *real* bad ideas (also really bad) – I have tested them all in my classes, sometimes repeatedly. Most of these ideas will take little effort to avoid. Avoiding others may require abandoning some conventional teaching methods and traditional simulation course content.

I have taught simulation for 26 years at several universities, to students with widely different backgrounds – from Freshmen to Advanced PhDs. I started teaching simulation at the University of Florida where I taught Simscript to graduate students (on Television) and GPSS to a class of IE juniors - using the original notes from Tom Schriber's now-famous text. The following year, I moved to Cornell where, for decades, I taught a very large class (up to 170 students) of seniors and Master's students along with an occasional PhD research course. Once, I tried to teach SLAM to Freshman (a bad idea). For the past few years at Berkeley, I have a taught a Master's course, a casual PhD research seminar, and a required junior course.

I am currently happy with about 90% of the content in my courses. I think that is about as well as one can do if they care about teaching.

The objectives of my courses are to teach students how to ask worthwhile questions and to recognize legitimate answers. I want them to become more demanding consumers of simulation technology - and of knowledge in general. *Simulation is the means, neither the topic or context.* I expect my students gain a comfortable understanding of uncertainty and how to model it. I want them to learn how to create and use dynamic models to make better decisions – and how such models might mislead them. I also hope they find building and experimenting with discrete event simulations easy, fun and useful, but not mysterious! In short, I take a "consumer education" approach to teaching simulation. I should warn you that this can backfire: several of my former students have later become my consulting clients.

My elements-of-a-simulation-study flow chart also includes a step called 'identify prejudices'. I think it is important for my students to realize that political agendas can have a greater influence on decisions than the very best simulation studies. (See Figure 1.)

Some of my particularly bad ideas follow. If you want to teach the worst simulation course I can imagine just follow the advice that appears in boldface type.

Homework: To try and keep my students busy, I used to **give students extended homework assignments spanning several weeks.** This doesn't work: most students put off doing homework as long as possible, then try to grind it out the night before it is due. I did it; my students do it; so did you. I once thought monster homework sets were a swell idea. I stopped doing this after seeing an informal teaching evaluation gouged into the men's room wall. This was neither constructive or complimentary, but he spelled my name correctly (It's probably still there)[2].

I give students complex problems, but keep them small scale. There is no way that I can replicate the magnitude of a significant, industrial-strength simulation study like those I have seen in my consulting work. I tell students about these projects, but give them the simplest exercises that effectively illustrate the point. I also tell them why I am asking them to do something, this is more for my benefit then theirs. Justifying a homework is sometimes harder than writing it.

I require my students to explain clearly, concisely, and neatly *why* they did something, not just *what* they did and *how*. I also ask them to try and make it interesting reading and try to sell their ideas to us.

Another way I have tried to keep my class busy is to **insist that students collect 'Real World' data.** Lately, rather than have students collect real data, I ask them to tell me specifically what data they would collect, how they would collect it (with data collection forms), what this might cost in time and money, and how they would use the data. This is not the same as doing it; on the other hand, students can be very creative in inventing data. Furthermore, I think data is highly rated (more later). As seen in Figure 1, I emphasize not collecting data until one has a pretty good idea what data is necessary after doing some sensitivity analysis.

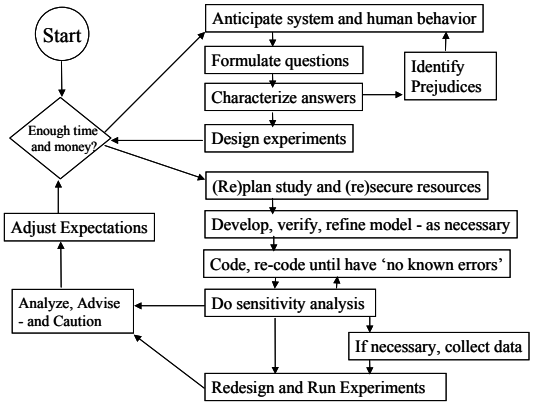

Figure 1: Concurrent Steps in Simulation Project[1]

Incidentally, I also do not require my students' computer codes to be error free – or even running. How they design their model is more interesting to me then how they code it. I believe that the best a simulation program can achieve is 'no known errors'.

I have found that no matter how easy the homework assignment, the more persuasive students will try to get me or my Teaching Assistants to do the work for them – some succeed (future managers?). Which leads me to a closely related bad idea. **Hold extended office hours the day before a big assignment is due.** My posted office hours read: '2:00pm until the queue is empty'. If an assignment is due the next day, my queue might not be empty until the next day when they follow me to lecture[2].

Motivating Students: I once received, and followed, the following bad advice from a senior colleague: **don't waste class time motivating students.** My undergraduate courses have always been required for graduation. This is an automatic strike against motivation for most students and the only motivation for some. My courses also take a lot of work, some hard analytical thinking, use mathematics and statistics, and involve some serious computation. (As if I needed another strike.) Still, I have managed to get consistently high course evaluations and have even won a

few teaching awards. I think this is largely because I spend a great deal of thought on how to motivate each concept and place methodology in a problem- solving context.

I once shamelessly used fancy animations from software vendors to motivate my students. Now, I don't like to spend any more than part of a single lecture on animation, and that is mostly cautionary. My consulting practice has taught me that animation is an analytically worthless activity that is often a waste of study resources (but it *is* fun!). Maybe animations were more important before Nintendo, but today's better managers want quantitative results.

My Berkeley undergraduates have had two years of calculus, physics, chemistry, thermodynamics, mechanics or electronics; they are eager to learn something they can 'do'. I discovered the best motivator for simulation students is to point out that this is probably their first college course about something someone might actually pay them to do. The summer after taking my class, a student once was offered more money than an average Stanford MBA to do simulation studies.[2] That gets a Berkeley student's attention – think of the tuition savings!

Peer pressure is also a terrific motivator. Last year, my students did their simulation projects over the internet[3]. On the web, they can run and evaluate their classmates' simulations. I am keeping the really strong projects to motivate next years' class – did I say motivate? maybe intimidate is a better word.

While I am on motivation: I use a grading method that keeps kids who flunk the midterm exam from giving up. I use multiple weighting schemes for exams, projects, and homework. One of the schemes assigns effectively no weight to the midterm exam. After the course is over, students retrospectively get the weighting that gives them their highest grade – thank goodness for spreadsheet macros. Caution: if you try this, make sure that all weightings include the final exam; if not, the better students may all cut it giving some slackers an easy A in the course[2].

Attendance: Most students and some faculty feel that **lecture attendance should be optional.** I have a twist on an old trick that assures nearly perfect class attendance. I give random in-class quizzes. Wait! This is key: I no longer call these quizzes, I call them "advanced placement (AP)" questions. A student's AP score is added to their score on the next exam. This takes pressure off the exams. In fact, students usually *request* more 'pop quizzes' as exams approach. This also tells me if they understand a point I tried to make, before the exam. That reminds me: I once wrote a "million point exam" to try to get students to stop hounding me for partial credit – ten thousand points here, ten thousand points there.

Fairness: This is a big deal to me. The worst advice I can give here is to **base grades, at least in part, on subjective measures** like "class participation". Of course, to treat students fairly is easy, just do it. More important, and harder, is to *appear* to be fair. Occasionally I have had to explain the difference between unfairness and irrationality. Students sometimes accused me of the former when they meant the latter, of which I have been guilty. I let students pick 'secret course IDs'. They put these and not their names on all their work. These IDs are useful for posting grades and handing back assignments. But, I primarily use them to curve the course without looking at students' names – and I do this with my TAs as witnesses. After curving the course, we look at the names below each grade. If a student put in an exceptional effort, we sometimes lower the line – but preserve the ordering. Once my TA argued successfully that a student had worked hard enough for an A, he later married her[2]. Whoa.

Cheating: you could **rely on the honor system**. I believe most people are honest; however, there are always a few students who would rather spend 10 hours working a scam then 1 hour studying. A single successful cheater can drain their classmate's morale and ruin a course. I have tried everything I can think of (shuffled multiple choice exams, a bounty, common random numbers…) I have concluded that prevention is the only approach to cheating that can work. Once you catch a student cheating, it is too late.

Teaching Assistants: maybe the worst thing you can do here is **ignore the opinions of your teaching assistants.** At Cornell my TAs almost always won the outstanding TA award (several have won it twice and one won it three years straight)[2]. Since moving to Berkeley, both my TAs have also won TA awards. Some if this is probably a manifestation of the "good cop/ bad cop" effect. My TAs are important, and not just for their work. I depend on their judgment and count on them to tell me when something is not working – this is actually how I found out that many of the ideas presented here were such losers. I also adjust the course content to the background and skills of my TAs. If a TA knows a particular simulation language, I invite them to present it to the class. I always require and pay TAs to attend my lectures. If I have strong TAs, I try something new; if not, I back off a bit.

Relationships to other courses: A bad idea here is to **present simulation as an alternative to analytical modeling.** As seen in Figure 1, I feel it is important to anticipate system behavior with analytical models before starting coding. Depending on a student's background, this might be in the form of an analytical queueing model, or maybe only a rough-cut qualitative (unscaled) response surface drawing. This has the side effect of forcing abstraction to suppress the natural tendency to put too much detail in a simulation model. I always introduce basic queueing models and concepts and present Little's law from a variety of different perspectives. I am not sure a better name for my course might be applied stochastic modeling. Probably the most misleading thing you can do is **tell students that simulation is the tool of 'last resort.'** Your credibility will be gone after they graduate and find it is often the tool of choice.

Software: I think probably the worst thing one can do is **focus your course around a commercial software package.** When I am asked: 'what language do you teach in your simulation course?' I respond: 'All of them'. It's true. The success of my students in using a wide variety of commercial simulation packages is solid evidence that this works.

Teaching a so-called, language-based simulation course has no place in a University. For better or worse, most simulation software is changed regularly, making such a course obsolete. It is like teaching a class in drivers' education, after which students can only drive a '94 Oldsmobile Cutlass. Simulation education, as opposed to training, should teach what goes on "under the hood" where all simulation packages have fundamentally the same engines. Commercial simulation software training is best left to the short courses offered by the software vendors who have developed and sell the product – I have helped teach such courses and academia can't compete.

Nevertheless, I try to introduce one or two commercial simulation packages to my students. This is primarily to make them better consumers. I emphasize language similarities rather than their differences. My students get an overview of the language, a demonstration and usually an exercise to do. For the homework I ask them to model a highly congested system; they are surprised how quickly most simulation software packages come to a virtual standstill when simulating a simple queue with heavy traffic! (See Figure 2.)

There are several programs that are specifically designed for teaching the fundamentals of simulation. I use one to teach all three classical modeling world-views. The software can be learned in a few minutes and can be used to model any discrete or continuous dynamic system. What I did not like about it, I changed.

Modeling uncertainty: I used to **spend much of the course on input distribution estimation and random variate generation.** This material contributes nearly half the weight of several popular simulation textbooks and I once thought it was important. Unless changing the system has no effect on its environment, then fitting distributions to real-world data for modeling exogenous simulation input is nonsense. If a system has no effect on its environment, why study it? If one were not thinking about changing a system, they probably should not be simulating it in the first place. If we **include distribution-fitting software in a simulation course**, we show students how easy it is to *remove* most of the useful information from a data set - such as trends, dependencies, and cycles. I am continually amazed in my consulting work to find people who collect a few weeks' worth of system data, fit it to the wrong scalar iid distributions, and then simulated a *different* system for years of operation to get narrow confidence intervals. The likelihood that their confidence intervals will actually include the performance measures they are studying gets worse the more they run their model. I am glad these were not my students.

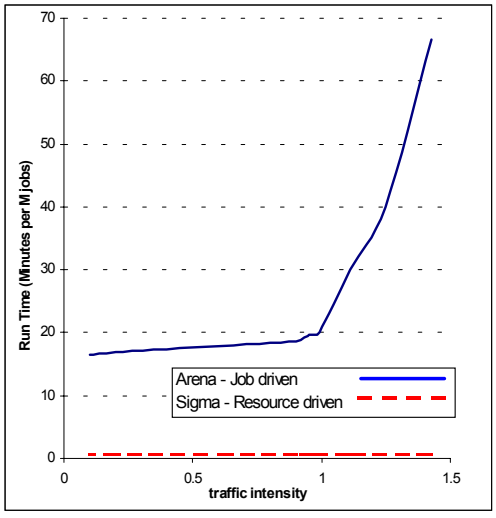

Figure 2: Run Times for a Simple Queue Simulation Using Different Methodologies[4]

How to model uncertainty? I tell my undergraduate students to start with a Beta distribution and then show them how they might create dependencies in and among the input processes and tell them try a fractional factorial design of distribution shapes. (I also tell them to never use an exponential distribution in an actual simulation study – who waits in line for three months anyway? – only someone memoryless.)

I do introduce basic algorithms for generating pseudo-random numbers and the four fundamental techniques for generating scalar variates identified by Bruce Schmeiser. I regard these as cultural, useful mainly as building blocks for modeling dependent processes.

Lest I give the wrong impression: understanding uncertainty is a major goal in my course. I use the time I save by not going through the litany of variate generation algorithms to emphasize different approaches to sensitivity analysis. I want my students to ask how a system would behave if the inputs were shifted, skewed, increasing, accelerating, cycling, less variable, more dependent, constant, chaotic… On every exam I give, an answer to a question is to perform a sensitivity analysis – and I tell students this in advance (Remarkably, some still manage to get it wrong – maybe they thought it was a trick?)

Finally, I think the point about teaching students how to ask better questions and recognize valid answers is worth re-iterating. Much of Engineering education focuses on teaching students how to answer questions; little on how to ask good questions, and rarely to question the questions. (Ironically, university professors are usually pretty good at this - particularly when refereeing each other's research papers[5].)

1 Contrast this with similar figures in most popular simulation textbooks.

2 All anecdotes are unfortunately true.

[3] Arlen Khodadadi wrote an excellent set of tutorials on how to do this.

[4] From Theresa Roeder.

[5] Research support from NSF, SRC, and ISMT has made a significant difference in how I teach my simulation courses.

2.4 Some Thoughts on Teaching Monte Carlo Simulation (B. W. Schmeiser)

Rather than focusing upon a single issue, I comment briefly on various aspects of teaching Monte Carlo (that is, stochastic-model) simulation. My comments, like the other panel members, are supported (and tainted) by being old enough to have begun my career in an era in which "doing" mathematical modeling (whether discrete-event simulation, system dynamics, probability, statistical, or optimization) required an understanding of first principles, including computer programming (for most of us, on punch cards). I also am similar to the other panel members in that my professional life bounces among trying to extend the research frontier to working with practitioners (as a consultant) to teaching students ranging from young college undergraduates to Ph.D. students. In addition, our careers have centered on stochastic models analyzed via digital-computer simulation. Our approaches, styles, interests, and abilities to teach simulation courses (at various levels) might have little relevance to the (common) situation where a faculty member is teaching simulation only because someone has to do it. Nevertheless, here are some thoughts, which I have organized into six points.

Point 1. University classes should focus on topics that are both timeless and difficult to learn in practice. Learning new concepts is difficult for a practitioner, who typically lacks time and guidance. A practitioner will learn, almost automatically, many of the practicalities of the real world. Real-world data collection and extended projects are inefficient use of a student's time. In an engineering curriculum, some exposure to the real world is traditional, at least via a senior design course. Much more efficient and useful is a sequence of relevant summer jobs. The other simulation practicality is the need to know a relevant simulation language, but this need is best met by the relevant company's one-week short course.

Point 2. Courses (in general) should integrate topics. Simulation courses, at many levels, can combine modeling constructs such as system dynamics, discrete-event, probability, statistics, and optimization, as well as mathematical, numerical, and Monte Carlo analysis methods. Course evaluations from my introductory dual-level "simulation" course regularly include comments that the learning of computer programming, probability, and statistics was a non-negligible course benefit. Using Microsoft Excel, I use simple Monte Carlo sampling in my sophomore-level introduction to probability and statistics. Event-probability problems are solved both mathematically and with Monte Carlo, foreshadowing the inferential statistical thinking that underlies the second probability and statistics course. I mention to the students, but don't emphasize, that the same thinking underlies the dynamic-simulation analysis that the IE students see later in their course work.

Point 3. The particular combination of modeling and analysis methods should depend upon the course's context. Course contexts differ dramatically: undergraduate/ graduate; stand-alone/sequence; management/technical. Purdue IE has the luxury of two parallel dual-level (undergrad/graduate) introductions to simulation, IE 580 (which is language based) and IE 581 (which is concepts based), as well as IE 680, a graduate-level research-frontier course. In IE 581, which I discuss more below, I tell the students that they may stop me at any time to ask me to explain why a course concept is relevant to practice; in IE 680 I make no such claim.

Point 4. To the extent allowed by budgets, courses should be segmented by student backgrounds and goals. With a homogeneous student background and goal, determining a good course structure is easy. Heterogeneous backgrounds and goals are far more challenging. This past spring, for example, IE 581 contained undergraduate industrial engineering students, Ph.D. students from mathematics and statistics, as well as M.S. students from departments such as forestry, construction management, traffic systems, management, and industrial engineering. Several of the students took the class because it is required for Purdue's Computational Finance certificate. Few of the students had extensive background in computer programming, probability, and statistics, but most had a solid background in one or two of these three topics.) Far too often the instructor with heterogeneous students faces the uncomfortable situation of boring part of the class while "blowing away" the rest. Law and Kelton's SimLib (see Law and Kelton (2000)) can be a challenge for a mathematics Ph.D. student while being straightforward for an IE undergraduate student; a few days later a discussion of random vectors (with arbitrary marginal distributions and dependency structure) lights up an entirely different set of faces. Fortunately, some topics work well for most students, for example, a discussion of how simulation analysis can fail.

Point 5. Integrate simulation topics within a course. The structure of IE 581 is based on increasing model complexity rather than on a sequence of topics. Topics such as U(0,1) random numbers, random-variate generation, input modeling, output analysis, and variance reduction arise multiple times. Dynamic models (those with a state that evolves over time), arise only during the last few weeks of the fifteen-week course. Only three models are used throughout the semester, although with some variations. The first model, which can serve for three or four weeks of the course, has no time component, is analytically solvable, and has only indicator variables as output. Typical would be a four-component reliability network, at first with inde-

pendent components and later with dependent components. This simple model motivates the need for random numbers (with a discussion of what that means on a deterministic computer), the need to convert those random numbers into coin flips, the need for standard-error calculations, the effect of statistical dependencies within a model, a simple intuitive variance reduction idea or two, the ease of doing some Monte Carlo simulations in a spreadsheet, the conceptual point that simulation's natural talent is high-dimensional integration, and a sense that analysis can be based on a combination of methods. The second model is more complex, requiring continuous random variables, logic complicated enough that a spreadsheet is unwieldy, and non-binary output data. Typical would be a muffler-warranty simulation, in which distributions must be specified for car lifetime, muffler lifetimes, and muffler costs. Such a model motivates the need to move beyond a spreadsheet, the need for some random-variate-generation ideas, the need for standard-error estimation for non-binary data, and the effect of another one or two variance reduction ideas. The course's third model requires discrete-event modeling. Typical would be a cleaning-and-patching station, two servers in tandem. New issues include the need for maintaining system state, the concept of event, time-based statistics, transient versus steady-state behavior, a sense of the problem of autocorrelation, the effect of non-stationary behavior (via a non-homogeneous arrival process), and the general modeling power of the simple next-event framework (as in GASP II and SimLib).

Point 6. Use an appropriate textbook as background reference. Such a reference includes far more information than can be included in the class. I spend little time on topics such as the structure of simulation projects or model validation; students can easily read the textbook author's ideas, which will differ from other people's ideas and for which simulation practitioners (and modelers in general) need to create their own philosophies. A good reference also helps the student later when he or she comes across a particular need, such as properties of various statistical distributions or ideas for model validation.

2.5 A Key Topic: How Simulation Software Works (T. J. Schriber)

Some other members of this panel have addressed the trade-offs involved and balance needed in teaching the elements of both modeling and analysis in a first course (which unfortunately is often also the only course) in discrete-event simulation. It's not possible to provide a "master's degree in simulation" in a first course, and so the many important topics falling under the umbrella of simulation have to compete with each other for some degree of inclusion in the first course. Each of us (who teaches simulation) has his or her own take on how the candidate topics rank in relative importance. For example, some might think that random number generators are not of enough *relative* importance in a first course to give more than maybe 20 or 30 minutes of airtime. Others might think that the logical foundations of discrete-event simulation software are not of enough relative importance to be included in a meaningful way (or even at all) in a first course. I myself am a believer in giving students the substantial *intellectual satisfaction* and *practical benefits* that result from bringing them to the point of truly understanding how discrete-event simulation software works (both in generic terms, and also in terms of the specific software they are using). I would like to argue here for meaningful inclusion of that topic in a first course.

In practice, a "black box" approach often seems to be taken in teaching and learning a discrete-event simulation software package. The external characteristics of the software are studied ("arrange the blocks or statements in this sequence and, voila, you have a model for three serial stations in a production line"; or "to model line switching in a multiple-line, multiple-server situation, arrange the blocks or statements this way"), but the logic on which the software is based is ignored or is touched on only briefly. Choices made on the part of the *language* designer (not the *model* designer) in implementation of the foundation might not be studied at all and related to *step-by-step model execution*. The modeler therefore might not be able to think things through when faced with needs like these:

- developing good (or even correct) approaches for modeling complex situations;
- using interactive tools (provided as part of the software) to come to a rapid understanding of error conditions arising during model development;
- using interactive tools to verify that complex system logic has been captured correctly in a model.

How feasible is it to include such material in a first course? In my experience, it is eminently feasible. A generic (language independent) model for the logical foundations of discrete-event simulation can be introduced and discussed in one (50-minute) class. In another single class, a fundamental subset of the corresponding step-by-step particulars (for the software being used in the course) can be introduced, and use of the software's interactive tools to support step-by-step tracing of execution of several fundamental models can be demonstrated (via use of an in-class computer and projection of the computer screen). Exercises stimulating additional learning on the part of the students can then be assigned. The fundamental understanding of what the software is doing can then be incrementally refined and exercised as additional features of the software are introduced.

The benefit of this is that students come to *understand* what happens at a step-by-step level when a discrete-event simulation is performed. This puts them in a powerful position to be intelligent and aggressive users of the software. Having learned only a subset of the software, this also brings

them to the point where they can *predict* what additional capabilities the software probably offers and *how* those capabilities might be supported at an underlying level (e.g., how does the language likely support features like these: modeling resource unavailability to reflect breakdowns, scheduled maintenance, or shift workers; modeling pre-emptive use of resources; and modeling inter-entity communication?)

Of course, the inclusion of any topic (even at only an introductory level) in a course does take time, with the possible (and maybe inevitable) effect of crowding out other topics. In terms of the discussion of software, the extent of coverage of a simulation package can be scaled back to free up time to include the "how simulation software works" topic. It is arguably better for a student to be familiar with, say, only 40% of a language *and understand how it works* than to be familiar with 75% of a language *but not understand how it works*.

Those who might want to delve further into "how simulation software works" are referred to Schriber and Brunner (1998) and Schriber and Brunner (2000). The philosophy and particulars of teaching "how it works" is given in Schriber (1991) for the case of GPSS/H.

2.6 Modeling, Programming, and Analysis (P. L'Ecuyer)

2.6.1 Modeling vs. Programming

I will use the same definitions as David Kelton for modeling and analysis, except that I put the specification of input distributions and processes in modeling instead of in analysis. Stochastic modeling thus involves probability and statistics. I also want to emphasize the distinction between modeling and programming. People often refer to their simulation program as "the model"', which I believe is unfortunate and sometimes misleading.

A model of a system is a purely mathematical abstraction, usually with many simplifying assumptions about the system. Building an appropriate model, realistic enough for answering the questions of interest but otherwise as simple as possible, is the most important and difficult part of a simulation project. Once the model is clearly specified, programming it is usually routine. Too often, however, assignments in simulation courses and exercises in standard simulation textbooks concentrate only (or mostly) on the programming aspects: The model is fully specified and the students are just asked to program it. This is a bit like entering a 10 km race at the 9 km mark. And I must admit that I have often been one of the culprits as a teacher. Why? Maybe because it is easier to assign an exercise taken directly from the book than to come up with data from which a model is to be built. Perhaps another reason is that appropriate tools for stochastic modeling are not always easily available to the students. Such tools involve the use of statistics, which students often find rebarbative, but which I believe is necessary especially in computer science departments such as mine. In my opinion, unless they are for computer science students, simulation courses should put more emphasis on how to build models than how to program them. This does not mean spending time collecting data on a real system, but rather starting from a small incompletely specified model, together with some data, perhaps with an open possibility of collecting more data, and learn about what to do from there. Students should come out of a simulation course realizing that modeling is really the difficult and important part, and get at least a bit of intuition about appropriate ways of modeling uncertainty. For this, teaching material (both textbooks and software) that better supports the stochastic modeling activity would be welcome.

2.6.2 What kind of Simulation Software to Use?

Most of the attendance of the undergraduate simulation course in my department are computer science students, so it is natural that simulation programming remains an important topic (more than if I was in industrial engineering, say) and that some emphasis be put on the design and implementation of simulation software. Our graduate course in simulation, on the other hand, targets mostly students from operations research and computational finance, and is more oriented towards analysis and advanced topics, with very little emphasis on programming. For both courses, I find it important that the students have access to the internals (computer code) of the simulation software. Giving assignments that ask to add or modify facilities in the software (e.g., adding non-uniform variate generators, adding statistical tools or even changing the event-list implementation, etc.) is a great way to teach how simulation software is built. This is one of the reasons why simulation courses in my department are taught with homemade simulation libraries, written in general purpose simulation languages (currently Java and C). I never use commercial simulation languages or environments in my teaching and research. The use of graphical simulation environments is low on my list of teaching priorities. It is not hard to write libraries or frameworks for discrete-event simulation, in widely used high-level languages such as Java or C++ for example, with the same powerful simulation tools that are offered in specialized simulation languages. In my opinion, there is no need for distinct simulation programming languages, not only for teaching, but in the industry as well. Using a standard language has important advantages: it removes the need to learn another programming language with its own syntax (this saves time), and it facilitates the use of the standard compilers, the software engineering tools, and the rich software libraries that are available for this language (e.g., for optimization, statistics, computer graphics, etc.). By "standard", I mean languages that are well supported over all kinds of operating systems. This excludes Visual Basic, for instance. High-level simulation environments with graphical interfaces, such as those

currently on the market, could be adapted to fit on top of such frameworks.

2.6.3 Analysis, etc.

Analysis issues should be addressed at least to a certain extent in a first simulation course, with emphasis on principles and ideas rather than recipes. What are the important issues to address early? The students must certainly understand the philosophy behind the design of (pseudo)random number generators (RNGs). They should be given concrete examples of why not to trust the RNGs available in commercial software, in general, and be provided with concrete alternatives. (See my tutorial paper on RNGs in these proceedings.) There is no need to cover theorems about the period lengths of LCGs, the spectral test, bad ideas such as the mid-square method, etc. Regarding non-uniform variate generation, one should explain the main ideas such as inversion, acceptance/rejection, etc., and give some examples.

For output analysis, concentrate on giving intuition about the noise in the simulation results due to all sources (including the model building). Show how to estimate it and to compute confidence intervals, while insisting that the normal approximation is not always appropriate for this. One can forget about steady-state analysis in a first course. Stochastic optimization via simulation is a very important topic in practice, but also very difficult to realize. The students should be made aware of this. Finally, they should get some intuitive idea of efficiency improvement, (e.g., using common random numbers, control variates, etc.) via concrete examples.

2.7 The Interplay of Practice and Theory in Teaching Simulation (J. R. Wilson)

The title of my contribution is a shorthand for general considerations of balance in (a) the topics to be presented in a university course on system simulation; and (b) the relationship of those topics to the broader curriculum in which they are found. In my experience, especially at the undergraduate level, it is essential to maintain an effective balance between the hands-on, practice-oriented aspects of a simulation course and the theoretical principles of engineering, computer science, and probability and statistics that underlie simulation.

In recent years, I have found that in all courses I teach (not just the simulation-related courses), it is critically important to motivate students strongly at the outset by showing them the direct practical applications of the material that will be covered in the course. For students in my undergraduate simulation course, this involves a brief discussion of recent high-visibility senior design projects that involved simulation (and nearly all do) as well as recent consulting applications that involved the research of my master's and doctoral students. Showing the associated animations is particularly effective in stimulating the students' interest. For graduate students, the corresponding motivation is to discuss my recent work with other professors' graduate students that involved the effective use of simulation modeling and analysis techniques in their research.

As a follow-up to the motivational pep talk, in my undergraduate course I begin the hands-on portion of the course with a lab exercise that involves a manual simulation of a simple queueing system to introduce the students to the entities, attributes, events, lists, and resources that comprise a discrete-event stochastic simulation. This is rapidly followed by construction of a simulation model of the same system using a commercial simulation package. I try to weave all of the point-and-click bells and whistles into the discussion as unobtrusively as possible while emphasizing the correspondence to the elements of the manual simulation.

In the presentations and labs that follow, I introduce an increasingly complex series of examples that (I hope) illustrate good modeling practice as well as the specific features of the simulation package that enable the students to implement the simulation model, animate it, debug it, plan an experiment exercising it, and then finally to analyze the results that it generates. Interspersed with the hands-on labs are presentations on input modeling and transient and steady-state output analysis that are much more "theoretical," linking directly to previous courses on probability and statistics and applied stochastic models. I have found that is it necessary to "remind" the students about a number of basic results from these courses that they have had little opportunity to use until they take the simulation course. To maintain the students' motivation, I use the previous homeworks and labs as the vehicles for illustrating various issues that arise in input modeling and output analysis. Although I doubt that many students would say this is their favorite part of the course, on the whole the students find the alternation between practice and theory to be (tolerably) interesting. For many of the students, this is their first real opportunity to see the direct application of the various probability distributions, statistical methods, and queueing models that they learned earlier. Thus the simulation course provides many students with an opportunity to integrate and internalize many topics that are essential for practicing engineers, scientists, and managers.

In both my undergraduate and graduate courses, I assign a term project that is designed to tie together much of the material covered in the course. In the undergraduate course, the students are organized into teams, and they are responsible for arranging a project in a local industry. If this activity is carefully monitored, it can give the students invaluable experience in conducting a successful simulation study with an industrial client; and in some cases this project has formed the basis for a larger senior design project or even a master's project.

Although I find that in short order the students are able to point-and-click rings around me, they often seem to need an inordinate amount of help in (a) conceptualizing (model-

ing) some aspects of the systems they encounter in their term projects; and (b) debugging the resulting simulation models. For this reason, I have made a concerted effort to provide in-class examples and homework problems that require the students to think more creatively about how to model systems that do not appear in the familiar form of a queueing network model in which work pieces enter the network, move among the various work centers (nodes) in the network, then exit the system. Moreover, I have augmented the in-class discussion and labs on model verification (debugging) in an effort to make the students more self-sufficient.

Turning to the issue of graduate courses in simulation, I want to make two main points. The first is that I believe an introductory graduate course in simulation should include a detailed, comprehensive treatment of the basic principles of discrete-event stochastic simulation so that the students are well grounded in current techniques for random-number generation, (nonuniform) random-variate generation, and discrete-event programming as well as the usual material on input modeling and output analysis at the level of Law and Kelton (2000). Depending on the mix of students in the course, their interests, and the available time, it is also highly desirable to provide a comparable treatment of the basic principles of combined discrete-continuous simulation.

In moving beyond the topics in an introductory graduate-level simulation course, my second point is that there seems to be a significant gap in textbooks of an appropriate level and scope of coverage. This gap should be closed. All these points will be elaborated in the oral presentation.

REFERENCES

Fourer, R., D.M. Gay, and B.W. Kernighan. 1999. *AMPL: A Modeling Language for Mathematical Programming*, second edition. Duxbury Press.

Law, A. M. and W. D. Kelton. 2000. *Simulation Modeling and Analysis*, third edition, , New York, New York: McGraw Hill.

Schriber, T.J. 1991. *An introduction to simulation using GPSS/H,* New York, New York: John Wiley & Sons.

Schriber, T.J. and D.T. Brunner. 1998. How Discrete-Event Simulation Software Works. Chapter 24 in *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*, ed. J. Banks. New York, New York: John Wiley & Sons.

Schriber, T.J. and D.T. Brunner. 2000. Inside simulation software: how it works and why it matters. In *Proceedings of the 2000 Winter Simulation Conference*, ed. J.A. Joines et al., 90-100. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

AUTHOR BIOGRAPHIES

TAYFUR ALTIOK is Professor and Chair of the Department of Industrial Engineering at Rutgers University. He has co-authored the recent book *Simulation Modeling and Analysis using Arena*. His email is <altiok@rci.rutgers.edu>.

W. DAVID KELTON is Professor of Management Science and Chair of the Department of Management Science and Information Systems in the Smeal College of Business Administration at Penn State. His interests are in simulation methods and applications, stochastic modeling, and applied statistics. In 1987 he was Program Chair of the WSC, and in 1991 was WSC General Chair. He served on the WSC Board of Directors from 1991 through 1999, co-representing INFORMS. His email is <david.kelton@uc.edu>.

PIERRE L'ECUYER is a professor teaching simulation in the Departement d'Informatique et de Recherche Operationnelle, at the University of Montreal. He received a Ph.D. in operations research in 1983, from the University of Montreal. From 1983 to 1990, he was with the Computer Science Department, at Laval University, Quebec. He obtained the prestigious E.W.R. Steacie grant from the Natural Sciences and Engineering Research Council of Canada in 1995—97 and a Killam Grant in 2001. His main research interests are random number generation, quasi-Monte Carlo methods, efficiency improvement via variance reduction, sensitivity analysis and optimization of discrete-event stochastic systems, and discrete-event simulation in general. He is an Area Editor for the ACM Transactions on Modeling and Computer Simulation. His email is <lecuyer@IRO.UMontreal.CA>.

BARRY L. NELSON is a Professor in the Department of Industrial Engineering and Management Sciences at Northwestern University, and is Director of the Master of Engineering Management Program there. His research centers on the design and analysis of computer simulation experiments on models of stochastic systems. He has held many positions for the Winter Simulation Conference, including Program Chair in 1997 and current membership on the Board of Directors. His e-mail and web addresses are <nelsonb@northwestern.edu> and <www.iems.northwestern.edu/~nelsonb>.

BRUCE W. SCHMEISER is a professor in the School of Industrial Engineering at Purdue University. His interests lie in applied operations research, with emphasis in stochastic models, especially the probabilistic and statistical aspects of stochastic simulation. He is an active participant in the Winter Simulation Conference, including being Program Chair in 1983 and chairing the Board of Directors during 1988-1990. His email is <bruce@purdue.edu>.

THOMAS J. SCHRIBER is a Professor of Computer and Information Systems at The University of Michigan. He is

a Fellow of the Institute of Decision Sciences and is a recipient of the INFORMS College of Simulation's Distinguished Service Award. He teaches modeling, decision analysis, and the design of business-application software in Michigan's MBA program. He is a member of ASIM (the German-language simulation society), DSI, IIE, and INFORMS, and is listed in Who's Who in America. His email is <schriber@umich.edu>.

LEE SCHRUBEN is Professor and Chair in Industrial Engineering and Operations Research at the University of California at Berkeley. His research interests are in statistical design and analysis of simulation experiments and in graphical simulation modeling methods. His simulation application experiences and interests include semiconductor production, dairy and food science, health care, banking operations, and the hospitality industry. His email is <schruben@ieor.berkeley.edu>.

JAMES R. WILSON is Professor and Head of the Department of Industrial Engineering at North Carolina State University. He is currently a co representative of INFORMS–CS to the WSC Board of Directors. His e-mail address is <jwilson@eos.ncsu.edu>, and his web page is <www.ie.ncsu.edu/jwilson>.