Proceedings of the 2004 Winter Simulation Conference
R .G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, eds.

YOU ARE GOING TO TEACH SIMULATION – NOW WHAT? TIPS AND STRATEGIES

Michael Freimer

Department of Supply Chain and
Information Systems
Pennsylvania State University
University Park, PA 16802, U.S.A.

Lee W. Schruben

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

Theresa M. Roeder

Graduate School of Management
University of California, Davis
Davis, CA 95616, U.S.A.

Charles R. Standridge

Padnos College of Engineering and Computing
School of Engineering
Grand Valley State University
Grand Rapids, MI 49504, U.S.A.

Catherine M. Harmonosky

Harold and Inge Marcus Department of Industrial and
Manufacturing Engineering
Pennsylvania State University
University Park, PA 16802, U.S.A.

Ingolf Ståhl

Department of Managerial Economics
Stockholm School of Economics
P.O. Box 6501
S-11383 Stockholm, SWEDEN

ABSTRACT

Facing the prospect of teaching a simulation course for the first time can be a bit overwhelming. This panel shares tips and strategies for teaching simulation based upon a wide variety of experiences, from academic newcomer to many years in the field, and having a variety of student audiences. We hope everyone will come away with a new idea, with our particular focus of helping new academics consider different simulation teaching approaches.

1 MICHAEL FREIMER: OBSERVATIONS FROM A FIRST-TIME SIMULATION INSTRUCTOR

In the spring of 2004 I taught two simulation courses at Penn State. The first, an undergraduate course called *Simulation Models of Business Processes,* was an introduction to simulation modeling and the *Arena* software platform. The second (which had essentially the same name: *Management Systems Simulation*) was a graduate-level course with topics in Monte Carlo and discrete-event simulation. The material in the second course was aimed at Masters and Ph.D. students who need to use simulation as a research tool.

I am a relatively new instructor – this was my second year at Penn State – and though I have taught courses in decision support systems, probability and statistics, I had never handled a course in simulation. Luckily I had been a graduate assistant for simulation courses taught by Lee Schruben and Thanos Avramidis at Cornell, and I had been given an excellent set of undergraduate lecture materials developed by David Kelton.

1.1 The Student Audience

Both courses were offered by the department of Supply Chain and Information Systems in the Smeal College of Business. Of the two groups of students, the undergraduates were far more homogeneous. They entered the course with a semester each of calculus, statistics, and programming. As you might expect, they were extremely computer-literate and were familiar with the Windows environment and Microsoft Office products.

The graduate course, which targeted Smeal Ph.D. students, also drew some students from around the university. There were roughly a dozen Masters students from industrial and civil engineering, a Ph.D. student from hotel administration, and a post doc from forestry. The official prerequisite is a basic programming class and advanced undergraduate classes in probability and statistics, but many of the students had stronger backgrounds.

1.2 Concerns Going into the Courses

My primary concern for the undergraduate course was the amount of time I would spend grading homework. Since I did not have a graduate assistant, I imagined I would spend most of my time debugging *Arena* models. To avoid this, rather than assigning a homework every week, I assigned somewhat longer homeworks spread out every two weeks. This eased the burden on me slightly, but it created a lag between the time material was presented in lecture and when it appeared on a homework. The students found this frustrating.

The process of collecting homeworks and distributing feedback was not as difficult as I had feared. Penn State has an online system called *Angel* (I assume most universities offer something similar) that allows instructors to easily create course websites, collect and grade homework files, and send comments to students. This was invaluable since the assignments primarily consisted of creating and executing *Arena* files, together with generating some written description in a *Word* document.

I would guess that many undergraduate courses also involve a large-scale group or individual project, and this was true of mine as well. I had assigned such semester projects in other courses and discovered that students tend to leave most of the work until the end of the term. (I realize this is not exactly surprising.) This time around I tried the obvious solution, which is to require some deliverables earlier in the semester. The students were allowed to choose their own project topics (the goal was to analyze some business problem), and after a few weeks submitted a 3-5 page project proposal. After another month they wrote a model description and analysis plan. This structure worked well since it gave the students a basis for organizing their team effort and allowed me to provide interim feedback. (It also allowed me to rescue a project that I felt was heading in an inappropriate direction.)

1.3 The Structure of the Courses

The undergraduate course met twice a week: one lecture and one lab. This setup worked extremely well since I could describe a concept (e.g. common random numbers) on Tuesday and have the students practice with it on Thursday. I also tried to leave some time during each lab for homework discussion – either answering questions about the current assignment or reviewing common mistakes in a previous assignment. This was much easier in the lab setting, where I could project the *Arena* model at the front of the classroom, and the students could follow along on their PCs.

The graduate course was entirely lecture based, although I occasionally demonstrated some software-related issue with my laptop and the classroom projector. Since the class was smaller (roughly twenty students), it often evolved into a discussion of the lecture material, and it sometimes took off in unexpected and interesting directions.

Unlike the undergraduate course, the graduate course was not platform-specific, and I spent little time discussing the details of any specific software package. Since the class was geared toward helping students with their own research, I asked them to use a platform with which they felt comfortable. Various students submitted assignments based in *C*, *Matlab*, *Sigma*, and *Arena*.

1.4 Some Lessons Learned

The semester project turned out to be a great tool for evaluating how well the undergraduates absorbed and integrated the lecture material. I found they had the most difficulty with the basic process of modeling. I don't mean the part that involves creating a simulation model that behaves like a real-world system, but rather understanding the reason we build models in the first place. When faced with a business scenario, their immediate reaction was to start building a model, without first considering the business decisions they were trying to address with the model. Since these questions guide all aspects of the simulation study (the level of detail and boundaries of the model, data collection, and output analyses), the students were often left with models that were either too detailed or not appropriate for the decision at hand.

For this reason I think some of the most successful lab sessions involved putting a simple problem on the board (e.g. investigate the difference between the queuing systems employed by McDonald's and Burger King restaurants) and allowing that to guide the discussion. For example, the restaurant problem prompted a discussion of the psychology of waiting lines (and the relative importance of different performance measures), the relationship between simulation and queuing theory, as well as some new *Arena* modeling constructs.

One other idea I found useful to keep in mind with respect to the undergraduate course was something I remembered from Lee Schruben. Although we expect our undergraduates to be able to construct simulation models and perform credible analyses, it is likely that in their professional lives they will more often be expected to understand and interpret a simulation study performed by someone else. Therefore it is important that the students be good consumers of simulation work; when presented with a simulation study, they should know what questions to ask, what assumptions to look for, and how the output can be reasonably interpreted.

2 LEE SCHRUBEN AND THERESA ROEADER

The introductory simulation class we have taught to Berkeley undergraduate engineers could be labeled a "consumer education" course. The overriding goal is for our students to become more demanding and knowledgeable simulation users. We do not expect to create simulation experts with a single course. However, we do want our students to understand the fundamental issues and technologies that are

critical to a successful simulation project. Not all of them will conduct simulation studies after they graduate, however, it is a safe bet that all of them will rely on simulation results to some extent in making critical professional decisions. We want our students to ask the right questions. In short, we want them to be an inept simulation consultant's worst nightmare. There are three interrelated skills we wish to impart to our students to achieve this goal: communication skills, modeling skills and technical skills.

2.1 Communication Skills

In the first lecture, we ask our students "What do you call a person who answers questions?" (we get several answers: an expert, engineer, analyst or consultant). Then we ask them "What do you call a person who *asks* questions?" (silence…). Answer: "the Boss." This sets the tone for the course: learning to ask effective questions effectively. To achieve this goal, we give them some open-ended, non-specific homework assignments where they need to ask questions even to get started. For example: we might simply ask them how to improve service in an airport. It is up to them to come up with a reasonable way to measure customer service as well as discover the constraints, objectives, and trade-offs in the system. We allow each team of students a limited number of questions to do this.

We also try to teach our students to recognize a good answer (even if it is another question!), sometimes by giving them poor ones, occasionally intentionally. We show our students, by examples, that the result of a successful simulation study may well be another question. After years of training in answering questions, it is not surprising that students have trouble formulating solid, well-motivated, and technically explicit questions. There is perhaps no better tool for learning how to do this than by conducting a sensitivity analysis of a simulation model.

In our course, student teams develop web-based simulation tools, which are evaluated by their classmates. This forces them to communicate with their peers on two levels – as colleagues and as customers. We expect our students to explain clearly their modeling assumptions and the results of the study, as well as to describe the model itself. They also must be able to sell their projects. In addition to asking students to tell us *what* they did and *how* they did it, we ask them to tell us *why* they did something. Students are expected to explain and justify their assumptions and disclose any known errors in their code. Simulations are different from other analysis methodologies in that it is often tempting to include an excess of detail; we therefore ask them to also justify *not* making an obviously simplifying assumption. It is not uncommon for us to give students full credit for an assignment even if their code is not working perfectly. (We concede that they could debug a well-designed model given enough time and money.) Students learn to write one-paragraph summaries of their models; several course evaluations explicitly mentioned the value of learning to write these summaries. Being able to explain your work and capture another person's interest quickly is a valuable skill, whether at a critical meeting or a party.

We also show our students how to do animations, but do not stress this and try to keep a reasonable perspective. Animations, while sometimes helpful in debugging, are analytically useless and offer little in the way of realistic validation. As decision makers become more technically sophisticated and more used to simulations, animations become much less a necessary component of a successful simulation project. Experience has also shown that an animation that could take weeks to develop might have an effective lifetime of a few minutes.

2.2 Modeling

Our students are taught *qualitative* aspects of modeling: how to recognize similarities among system entities instead of being distracted by irrelevant differences, how to drive a good bargain between the cost of a simplifying assumption in reduced model validity and its benefits in tractability and interpretation of results, the value and risks of "real" system data that may be outdated or simply incorrect. However, it is unlikely that students will emerge from a single course expert modelers. What we feel we *are* able to teach them is to be educated critics of simulation studies: to question the assumptions, modeling decisions, uses of data, study objectives… again focusing on learning how to ask questions.

In modeling, students learn to exclude unimportant details that not only waste time but also might introduce errors. While some textbooks strongly emphasize data collection and distribution fitting, this is given much less emphasis in our course. Our reasoning is that if a process is not important, there is no need to collect extensive data about it. On the other hand: if a process is important, it is likely to be improved before the simulation project is completed, making the data about the old process obsolete. Rather than fitting distributions to last year's data from making last year's products, we emphasize the importance of sensitivity analysis and examining the impact on performance of changing system parameters. Again the focus of experimentation is on asking questions – in this case, "how important are the data?."

2.3 Technology

We also teach our students *quantitative* simulation modeling and analysis tools. A frequent question facing simulation educators is "what simulation language should I teach?." Our answer is an emphatic "all of them!" There are relatively few fundamental technologies used in simulation software. In lecture, we use one of those little Russian dolls with smaller dolls inside it to introduce each new layer of technology. Once you get past the interface of a particular language – the dolls look alike. While we do

have our students use different commercial languages (and change the languages used often), we feel that these actually form a barrier to a deeper understanding of the fundamental technologies. Simulation tools are designed for a competitive mass market that necessitates hiding the technological details.

We believe that university students are ill served by being taught a single simulation modeling approach using a single commercial software package. It seems much like a driver's education course that teaches students to drive only a 1999 Honda Civic, and no other type of car. University students, and engineering students in particular, should be taught the mechanics and how to drive all cars. There is no excuse for students to leave a university simulation course mystified by *any* simulation software package, even if they have never seen it before. In today's competitive market, simulation software changes all the time. Instead of focusing on a single package, we teach students what goes on "underneath the hood," stressing the similarities between software applications. This allows students to maintain a sense of confidence if they are asked to use, say, GPSS rather than Arena. It also tends to allow us to expose them to a broader range of modeling paradigms, since students are less constrained by the modeling viewpoint or semantics of a particular language.

We teach our students Activity Scanning, Process Interaction, Event Scheduling, using Petri Nets, Process Block languages (like Arena or GPSS) and Event Graphs, respectively. We do this using Sigma (Custom Simulations, <www.customsimulations.com>), software that was specifically designed for teaching the fundamentals of simulation that facilitates modeling with all of the discrete and continuous simulations modeling worldviews. The following is a quote by a member of the first place University of Arizona team (out of 54 entries) in the recent national IIE/Rockwell sponsored Arena language modeling competition.

> *"Our simulation class was taught in Sigma, not Arena, but our instructor did an excellent job in teaching us how to construct simulations so that we could pick up any package and use it. He didn't want us to be so specialized that we would only be able to use one software package."* (University of Arizona 2002)

This indicates that perhaps the best way to teach Arena (or any other language) is using Sigma. By taking care of the fundamentals, Sigma allows students essentially to build their own languages. Examples have included Arena-like (or GPSS-like, etc.) process languages and simulated stochastic Petri Nets.

Another reason that we do not want to design our course around a particular simulation language is that, despite advertising to the contrary, there is no single, one-size-fits-all, best choice language for all applications. Figure 1 illustrates this with a comparison of two modeling approaches to a simple queue. The execution speed of a process interaction approach degrades severely as the system becomes congested; the event scheduling approach used is insensitivity to congestion. Clearly for large-scale highly congested systems, the event approach is preferable. The event scheduling is also better suited for modeling complex systems. The process interaction approach is superior for relatively simple, non-congested, queueing systems where fast code development is more important than fast code execution.

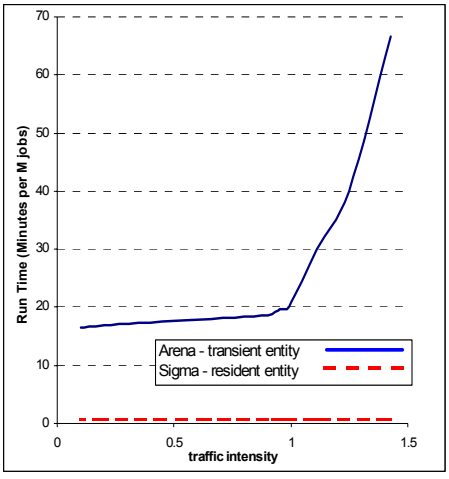

Figure 1: Run Times Using Different Methodologies

Figure 2 illustrates the progression of technologies we taught last year. We start out with several weeks of Activity Scanning (using Petri Nets) and a Process Interaction package (last year we use Arena). This leads naturally to the underlying technology of event scheduling. This, in turn, leads to C-based models automatically generated by Sigma that can be easily called from Excel spreadsheets or web pages using VBA and HTML/ASP scripting. This is a large amount of software; however, we have developed detailed tutorials for all of it so very little lecture time is used (Berkeley Simulation Group, <www.ieor.berkeley.edu/~bsg>). In the past few years all of our student teams have been able to develop their own spreadsheet or web-based simulation tools.

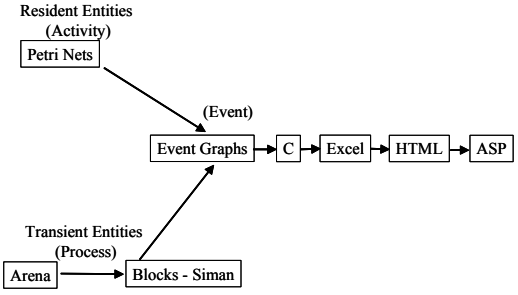

Figure 2: Technologies Taught

As with modeling, it is unlikely that students will emerge from a single course in simulation as technology gurus. This is not our goal. Instead, we would like students to become fearless when confronted with different simulation technologies. We want to give them the tools to learn. At the beginning of our course, most students are unable to create more than basic web pages. By the end, they are designing sophisticated web sites to accept user input, execute simulations, and display the results. They are acquainted with the underlying technology, and are not afraid to use it. They also know where to turn if they encounter problems with a particular technology.

Our course is different from many in that we emphasize communications and asking good questions rather than finding answers to our questions; we de-emphasize real world data and distribution fitting in favor of sensitivity analysis; and we do not teach a single simulation language, but rather how all of them are similar. Also, the course has been paperless for the past few years.

3 CHARLES STANDRIDGE – BY THE END OF THE COURSE THE STUDENTS SHOULD KNOW WHAT?

In designing any course, one fundamental question to ask is: What should the students know as a result of completing the course? I will discuss the answer to this question with regard to an introductory simulation course offered either at the upper division undergraduate level or at the introductory masters level within the context of an applied engineering program. I currently teach in the Product Design and Manufacturing Program at Grand Valley State University.

For me, the fundamental answer is simply: by the end of the course the students should know how to do a simulation project. Achieving this goal requires the following course content:

1. Identification of the application areas of simulation including:
   - Push systems such as lines and job shops.
   - Pull and other lean manufacturing systems such as kanban and CONWIP controls, work cells, and flexible manufacturing systems.
   - Material handling systems such as automated guided vehicle systems, automated storage and retrieval systems, and large-scale conveyor systems, for example package transfer centers.
   - Supply chain issues including automated inventory management, transportation and delivery logistics, and integrated supply chains
   - Business systems including project management, contact centers, office operations, health care delivery, and IT operations.
2. A simulation project process.
3. How to model system components such as workstations, routing, inventories, pull operations, competition for a single resource, choice between resources for a single activity, transportation devices, conveyors, batching and arrival processes where the mean changes in time.
4. How the simulation engine works.
5. Verification and validation approaches.
6. Design, execution and analysis of simulation experiments, both terminating and steady state.
7. Fitting distribution functions to data.
8. A commercial off-the-shelf simulation environment.

Our introductory courses are organized in the following way. The course meets in a computer aided teaching (CAT) studio. A CAT studio has all the capabilities of a classroom and a computer lab. Each student has access to a computer. Each course meeting is an appropriate mix of lecture, active learning, and computer-based laboratory activities.

The first third of the course is devoted to simulation methods, those items stated in items 2 through 8 above. Lectures are used for the presentation of materials. Computer based assignments are used to reinforce the concepts presented in lecture and include the following:

1. A tutorial is used to introduce the simulation environment used in the course. The tutorial provides the students with step-by-step instructions for modeling building and experimentation in the context of analyzing a two workstations in a series system. This allows the students to see how the environment is used for conducting an entire simulation project. Basic modeling constructs such as entities, resources, and queues are employed. The tutorial takes 2-3 hours to complete. ProModel is used for the undergraduate course and AutoMod is used for the graduate course.
2. A tutorial is used to introduce software (ExpertFit) for fitting distribution functions to data.
3. Laboratory exercises allow students to practice the complete analysis of a terminating simulation experiment and of a steady state simulation experiment using provided data. Verification and validation is included.
4. A laboratory exercise allows students to practice the operations of a simulation engine.

The remainder of the course is devoted to simulation projects. Project topics span the application areas listed above. These projects are simplified versions or metaphors for typical industrial applications of simulation as discussed by Shore and Plager (1978) as well as Standridge (2000). For each topic, a completed study is presented. This study follows the simulation process previously intro-

duced. Additional modeling and experimentation topics are introduced as needed. Student activities related to each completed study include:

1. Questions for further discussion of the completed project.
2. Computer exercise for further analysis or extension of the completed project.
3. A related, but significantly different application problem plus discussion questions. The problem is open-ended. Students may perform a simulation project to deal with the issues raised in the application problem or simply address the discussion questions. For a completed simulation project, each student writes a brief report that states and defends recommendations on how to deal with the issues raised in the application problem.

No more than one application problem is assigned from each of the topical groups listed above. Four application problems are assigned in the undergraduate course and three are assigned in the graduate course. In addition, each student completes a student-defined term problem based on undergraduate co-op work experience or graduate industrial experience.

4 CATHERINE HARMONOSKY

My experience teaching simulation has been with Industrial Engineering students, and my comments here focus on strategies for an undergraduate required introductory simulation course for these students. I have learned that you can make both major structural changes to the course and minor modifications in delivery that can have positive impacts, leading to a more successful course experience for all.

When I first taught simulation, it was a straight lecture-based format, with modeling assignments given as homework. A major structural change occurred when we moved to a lecture and lab format, which has worked very well. Both the students and I are less frustrated because I can ensure better synchronization of lecture material with hands-on modeling experiences in weekly labs. The formal labs also provide group time built into students schedules to ensure that assignments get started in a somewhat structured format. Further, because we try to have a "design studio" approach, lab groups have the opportunity to ask questions of an instructor while modeling, which can jump them over a hurdle quickly and get them rolling again. So, I would encourage you to consider this type of structural change if you are still only lecture-based.

A modification in delivery that I have made is to get students modeling in the very first week of the course—but not necessary with a simulation language. Using a single-server system simulated by hand (get out the dice, coins and a spreadsheet to help track system output performance measure statistics) has served me well. It gets the students actively participating early, and it allows me to introduce basic concepts of random variates, the simulation clock, output statistics, multiple replications and confidence intervals. Throughout the semester, I often reference this experience as we learn more about these basic concepts.

But, there are some things that I have learned the hard way that I will share now to hopefully spare someone else pain and suffering.

False assumptions from my past:

- *Myth 1*: I need to give the students 2 weeks to complete modeling assignments so they can start early, work consistently and have ample time to come for help.
  Fact: Students are busy and procrastinate. So, assignments were started 2 days before the due date, which meant they had forgotten the related course material and they blamed me that they "didn't have enough time."
  Resolution: They have a lab section meeting every week with a lab report due the following week and no late labs are accepted. They keep on track and lecture material is synchronized with labs.
- *Myth 2:* By presenting topics and concepts in lecture Monday that students must use in their lab assignment Wednesday, students will appreciate this synchronization.
  Fact: Students still complained, "Lectures have nothing to do with labs."
  Resolution: Adding the phrase "you will get hands-on experience with this concept in lab this week" during lectures and adding the phrase "as we discussed in lecture" to the lab presentation, while changing nothing else in the course, eliminated these complaints.
- *Myth 3:* More frequent exams with less content each will help keep students on track.
  Fact: This increased my workload significantly and distracted the students from focusing on modeling and analysis experiences.
  Resolution: I give one mid-term and one final and the majority of their course grade results from their assignments where data analysis and drawing conclusions is emphasized.
- *Myth 4:* Having students work individually means they will really learn and understand all aspects of simulation.
  Fact: Students study together and help each other anyway, so some students still never truly understand the concepts. Further, we now need to foster our students' abilities to work in a collaborative group.
  Resolution: Make formal student groups to encourage collaboration within but not between groups. Students are less frustrated because they can learn from each other, and with a consistent

group for a semester they typically become comfortable asking questions and offering suggestions within the group.

5 INGOLF STÅHL –TOP TIPS TO A TEACHER

I shall here attempt to provide some tips to a teacher giving a course on simulation for the first time. I shall base my tips of my experience from having taught simulation to 6000+ business students in Sweden and the US during three decades. I must first specify what kind of simulation course I refer to. I exclude simulation courses for computer science majors, but I refer to ones for business students as well as engineering students, e.g. of production, logistics, material handling, supply chain management, etc. I mainly refer to a full course, being e.g. one of 10 courses taught to undergraduates in a year. I also refer to an optional course.

For a teacher of such a course, it is generally important to assure that one is allowed to teach the course at least once again, preferably many times again. It is bad a strategy for a teacher, in particular one on a tenure track, to have to constantly teach new courses. There is a big fixed cost preparing a course for the first time and it is efficient to have this fixed cost spread out over many courses.

With this goal in mind, my tips will be focused on making the course so successful that it will be taught again. An important part of this is to get good student ratings, not only in the formal ratings run by the university, but also by the "grape vine" ratings, where younger students ask older students which optional courses they recommend.

Among the main factors leading to good student ratings of a course, I will focus on the following three:

1. The course is enjoyable, with a simpler word, **fun**.
2. The course is helpful in future studies.
3. The course will be helpful when landing a job.

I have assumed that there is some monitoring of the course so that a teacher cannot "compete" with other courses by requiring less effort than other courses do or by giving significantly higher grades than in other courses.

For making a simulation course **fun**, there are, according to my experience, several factors of special importance.

1. Avoid overwhelming the students with difficult, often unnecessary, technical details.
2. See to it that students at a very early stage get to write simple programs that are not trivial.
3. Proceed step by step, with simple examples in the beginning, so that no students are left behind at an early stage and lose the possibility to catch up.
4. Try to avoid the need for pre-course knowledge requirements, e.g. of computing. Unless one has clear pre-course requirements and knows that the students have passed these requirements recently, one will also here run the risk of losing some students at an early stage.
5. Students should not have to learn a new concept every time that a new and different thing shall be done. It is preferable that the new aspects can be handled using already known concepts, even if the programs become slightly longer. One should carefully restrict the number of concepts taught.
6. One should not sacrifice the ease of introduction for the sake of having sophisticated features for advanced users. It is important to encourage students to forge ahead and experiment on their own.
7. Many students like programming, but hate debugging. Hence, try to minimize the risk of the student making logical errors. Students should not run into surprises and unexpected errors due to not having learnt the full simulation system. Try to choose a simulation system with an easy-to-learn system for debugging and program verification, e.g. in the form of block based animation, and with an extensive error trapping system with clear error codes.
8. It is helpful if input can be made using a Graphical Users Interface, where the student can choose the building blocks of the program from a menu of symbols. The student should then for each building block be able to open a dialog for inputting the operands of this block. This dialog should reveal the main syntax of the block operands.
9. Allowing for graphs and possibly also some form of simple animation will also make simulation more fun.

To have a simulation course **useful also in other subjects**, one can have good examples with applications that are highly relevant for later courses. Let me give some examples. In some simulation courses I have had the students simulate the cash flow, as well as the profit/loss accounts and balance sheet, of a small corporation, dependent on the inventory policy of this firm (Ståhl 1993). This exercise has proved useful for the students in later courses in corporate finance. Likewise, I have in other simulation courses given larger examples dealing with new product development and simulation based costing, helpful in later specialized courses in Marketing and Management Accounting (Ståhl 1995).

As regards usefulness when it comes to **landing a job**, I have found that a simulation course involving a project out in a company in many cases has landed students a job. It should be recognized that in many companies the art of discrete events simulation is little known and many medium-sized company have no expertise at all in this area. Hence, a reasonably good student can even after only roughly 20 classroom hours do something useful in a company that no one in the company could do. The simulation student can hence at an early stage be something of a spe-

cialist. It is also advantageous that the simulation projects can deal with very down-to-earth problems.

I have had good experience of students in groups of two or three doing project work in different Swedish corporation, for example in banking, telecommunications and retailing. Quite a few projects have dealt with "sales support simulation models", where the simulation model is run on a laptop in interaction with a client, regarding e.g. the optimal configuration of a corporate telephone exchange system. Many of the project programs have had continued use in the corporations.

In these projects, the students get a chance to work their way through the whole simulation process as regards some concrete problem, from delimiting the actual problem, formulating the question to be answered, gathering data, outlining the program graphically, coding the program, verifying, validating and documenting the program, running the program a sufficient number of times, doing a statistical analysis for drawing significant conclusions, and presenting the results in a form suitable for a potential user, with a focus on getting the results implemented.

There are also other issues that should be mentioned. It is very important that the students understand that the simulation programs should be **run several times** with different random streams and hence it is essential that it is very easy to make replications of the runs. It is also desirable that the simulation system can automatically carry out a statistical analysis of these repeated runs, e.g. calculating the limits of the universal average, as well as presenting a histogram of the distribution of the results of the runs.

As regards random **distributions** used in the simulation, I have found it suitable to start with teaching how to get data for an empirical random distribution, e.g. by a number of pairs of value and frequency. We would later in the course deal with complicated statistical distributions, since it is more difficult to estimate the parameters for these distributions on the basis of input data.

I should also say something about **animation**. While animation is a factor that tends to make a simulation course more fun for the students, it is easy that too much focus on animation can ruin the whole course. If one starts a course allowing for animation, it can happen that the students spend virtually all time on drawing neat symbols and disregard the fundamentals of simulation. For many types of problems in business, like service systems, where events occur with long time intervals and require very different times, animation might not be helpful at all.

A much simpler form of animation, where simple entity symbols are allowed to move through a block diagram, step by step at each event, can on the other hand be used already early in the course to increase understanding of how the program works and to be used as a simple debugging device, e.g. for program verification.

I would also like to encourage the **teaching in computer labs,** at least in the beginning of the course. The teacher can here, using a projector, show how the simulation software is used and the students can then on the PCs in the lab immediately check that they are able to follow. It is, however, important that the students can read all important items on the projected screen picture.

After such an introduction, it should be easier for the students to carry out self-studies in front of the student's own computer. To support this, one should provide the students with a great many program examples and tutorial lessons. It is highly desirable that the simulation system in the course is available at a very low cost, so that students can afford to buy their own copy of the software.

I shall finally give some advice that directly concerns the teacher. It is very important that the teacher can be the full **master of the course**. This means that he or she must be able to answer all the questions that the students have, e.g. in connection with the project work. This is a factor to consider when choosing the simulation software for the course. If one chooses a complex commercial package, one faces the risk that there will be many student questions that one will be unable to answer.

This choice of software is also connected to the choice of the course **textbook.** The student should not need a manual in order to find features not covered in the textbook or in class. According to my experience, many students have run into great difficulties in their project work when they have attempted to use features that are not covered in the textbook. It is hence important that the textbook covers every aspect of the simulation system. It is for student rating also of interest that the book has a moderate price.

For the teacher it is furthermore of importance to be able to correct and mark the student programs with ease and to sometimes help the student find errors, when they have really got stuck. For this purpose good documentation of the program is very important. Some simulation systems have the advantage of providing both a compact and readable text version of the program as well as an easy-to-read block diagram presenting the logic of the model. Also in this respect, the choice of the simulation software to be used in the course is of importance.

REFERENCES

Shore, B. and D. Plager, 1978. Simulation: a case approach. In *Proceedings of the 1978 Winter Simulation Conference*, ed. H. J. Highland, N. R. Nielsen, and L. G. Hull, 361-370. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Ståhl, I. 1993. Discrete-event simulation for corporate financial planning. In *Proceedings of the 1993 Winter Simulation Conference*, ed. G. Evans, M. Mollaghasemi, E. Russell and W. Biles, 1264-1269. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Ståhl, I. 1995. New product development: when discrete simulation is preferable to system dynamics. In *Proceedings of the Eurosim Simulation Congress 1995,*

ed. F. Breitnecker and I. Husinsky, 1089-1094. Amsterdam: Elsevier.

Standridge, C. R., 2000. Teaching simulation using case studies. In *Proceedings of the 2000 Winter Simulation Conference*, ed. J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, 1630-1634. Institute of Electrical and Electronics Engineers, Piscataway, NJ.

University of Arizona. 2002. UA industrial engineering seniors win software simulation competition [on line]. Available online via <www.sie.arizona.edu> [accessed June 14, 2002]

AUTHOR BIOGRAPHIES

MICHAEL FREIMER is an Assistant Professor in the Department of Supply Chain and Information Systems in the Smeal College of Business Administration at Penn State University. He received a Ph.D. from Cornell University's School of Operations Research and Industrial Engineering (ORIE) in 2001, then spent a year as a visiting assistant professor at Cornell's School of Hotel Administration and a lecturer at the School of ORIE. He also has industrial experience with the operations research consulting firm, Applied Decision Analysis, Inc. in Menlo Park, CA. His email address is <mbf10@psu.edu>.

LEE SCHRUBEN is Chancellor's Professor and Chairman of the Industrial Engineering and Operations Research Department at Berkeley. His research has been on a variety of simulation methodology and simulation application topics most recently in the area of bio-production. His email address is <schruben@ieor.berkeley.edu>.

THERESA ROEDER received her PhD from Berkeley in 2004 and is currently on the faculty at the Graduate School of Management at UC Davis, where she will be teaching MBA students statistics. Her email address is <tmroeder@ucdavis.edu>.

CHARLES R. STANDRIDGE is a professor in the School of Engineering, Padnos College of Engineering and Computing, at Grand Valley State University. He has 30 years of simulation experience in academia and industry. He has performed many simulation applications, developed commercial simulation software, and taught simulation at three universities. His current research interests are in the development of modular simulation environments (MSE). He is working with industry on the application of MSE to supply chain management problems emphasizing strategic and tactical logistics as well as inventory management. His teaching interests are in the concurrent use of factory physics, lean manufacturing, six sigma and simulation in introductory undergraduate and graduate courses using a case-based approach. He also teaches in the area of engineering measurement and data analysis. He has a Ph.D. in Industrial Engineering from Purdue University. His email address is <standric@gvsu.edu>.

CATHERINE M. HARMONOSKY is an Associate Professor in the Harold and Inge Marcus Department of Industrial and Manufacturing Engineering at Penn State University. She received her Ph.D. from Purdue in 1987. Her research and teaching interests are in simulation, scheduling and production planning and control. Her email address is <cmhie@engr.psu.edu>.

INGOLF STÅHL is a Professor at the Stockholm School of Economics, Stockholm, and has a chair in Computer Based Applications of Economic Theory. He was visiting Professor, Hofstra University, N.Y., 1983-1985 and leader of a research project on inter-active simulation in Vienna, 1979-1982. He has taught GPSS for almost thirty years at universities and colleges in Sweden and the USA. Based on this experience, he has led the development of the micro-GPSS and WebGPSS systems. His email address is <ingolf.stahl@hhs.se> and the web address for WebGPSS is <webgpss.hk-r.se>.