AN INTERDISCIPLINARY INTRODUCTION TO SIMULATION

John W. McCredie

Carnegie-Mellon University

ABSTRACT

This paper presents a description of an interdisciplinary introduction to simulation developed at Carnegie-Mellon University. An outline of the course goals precedes a description of the techniques discussed, the teaching methodology used, and the application projects assigned. The final section contains an evaluation of the course with respect to the goals of the first section and the computation and teaching resources needed to support the effort.

INTRODUCTION

"Computer Simulation and Modeling Techniques" is an interdisciplinary introduction to the methodology and application of simulation developed at Carnegie-Mellon University. This one semester course has grown, during the last four years, from a class of eight undergraduates interested primarily in computer science to a class of about 50 students from at least eight departments. The course is not required by any department, nor is it part of a sequence of programming courses even though it is offered by the Department of Computer Science. Most of the students are juniors or seniors but there are always a few sophomores and graduate students. The teaching staff actively encourages a broad spectrum of student interests and skills.

The two prerequisites for the course are: (a) successful completion of the introductory course in programming and problem solving or the equivalent skill in formulating and implementing solutions to a variety of problems on computers; (b) successful completion of an introductory course in probability and statistics (in some cases where a student's mathematical background appears to be very solid, such a course may be taken concurrently with the simulation course).

COURSE GOALS

The major goal of the course is to provide an environment in which students may learn how to use computer simulation techniques in the analysis of the behavior of complex systems from a variety of subject areas. Although all of the students have some previous programming experience, most have not been exposed to the types of data structures and program control features required to represent a broad range of system configurations. The basic features of discrete event and continuous time models are presented in a way that students may form their own taxonomy of system and modeling structures. They learn how to analyze a system and judge what type of simulation approach is appropriate to model it on a computer.

More specifically, by the end of the term students completing the course should be able to:

1. judge when, if, and at what level simulation should be used to help solve a particular problem;
2. select the appropriate technique for a variety of problem types;
3. implement a simulation in at least two programming languages;
4. design and evaluate an original simulation experiment;
5. organize and prepare a written analysis of a simulation study.

COURSE CONTENT

To help achieve the goals outlined in the previous section all students are required to complete four simulation projects which are specified in class and a term project of their own design. Class lectures focus upon points raised by the four required projects. The textbook, System Simulation, by Gordon (1), serves as a reference for the course. The lectures probe selected topics in depth. There is no attempt to cover all of the assigned readings in class. The text provides both continuity to the selected class discussions and an alternative approach to a number of topics. Gordon's book presents a good overview of both continuous and discrete event modeling techniques. Other texts placed on reserve in the library, but not used directly, are: Naylor, Balintfy, Burdick and Chu (2), Meier, Newell, and Pazer (3), Maisel and Gnugnoli (4), and Mihram (5).

The four projects range from a simple, completely specified, first assignment to a partially determined problem which requires extension by the

student. These projects prepare a foundation upon which the term project is built. There is no final exam in the course. Instead all effort near the end of the semester is focused upon the individually designed term projects. A mid-term hour exam points out weak areas and provides early feedback to the students. Since the projects require a large amount of staff time to evaluate, by the mid-term grading period the results from the first project and the hour exam are the only performance measures available. However, they have proven to be fairly reliable indicators of progress. A number of students have improved their performance later in the term, but they rarely perform well at mid-term evaluation and then falter in later work.

After an introductory lecture that attempts to introduce students to the world view of the simulation analyst, we present a survey of the basic structure of simulation languages. An important question for everyone who teaches a course dealing with computers is: What language should the class use? For a simulation course this question has a special dimension. Should students learn a general prupose simulation language (e.g., GPSS, SIMSCRIPT, SIMULA, etc.), or should they learn how to build models in more common procedure oriented languages (e.g., FORTRAN, ALGOL, PL/I, etc.)?

During the first two years of the course the choice was to teach SIMULA and use this language as an example of general purpose simulation languages. As class size grew and computing resources became more scarce our language policy changed. Now students learn how to construct features such as random number generating procedures and list processing routines in FORTRAN, ALGOL, APL, or PL/I. Their choices are based upon previous programming experience. Approximately half way through the semester the class lectures start to emphasize the structures of GPSS and SIMULA. The fourth project asks the students to reprogram the third project in either of these two languages. Thus they have the experience of creating the same model in both a general algebraic language and a simulation language. The experience gained in learning how to construct basic simulation tools in a general algebraic language helps students to learn the specific syntax of a simulation language quickly. They are free to use any language they wish for their own term project. About half of the class use GPSS or SIMULA, and half use FORTRAN or PL/I.

A basic weakness in many students taking this course is an inability to express technical subjects well in writing. Thus each project emphasizes the documentation of the work in a clear, well structured, write-up. A significant amount of class time is spent discussing these presentations and ways of improving them.

The first project is a simple sampling experiment to determine the mean distance of two points randomly distributed in a unit square. An analytic solution to this problem is very hard for students at the undergraduate level. Thus they experience both the power of simulation and the problems inherent in estimating, and specifying confidence intervals for, parameter values from a Monte Carlo experiment. The second project is a model of a continuous system described by a set of non-linear differential equations that include time lags. The focus for this project is traffic flow in particular and the more general topics of data structures and control features useful for continuous system modeling. The third project is to build a model of a queueing system. This problem provides a rich environment for learning about discrete models. The fourth project is to reprogram the third project in either GPSS or SIMULA in order to explore the capabilities of general purpose simulation languages.

Early in the term students are asked to begin thinking about possible term projects. Each student is encouraged to work in an area that is of personal interest and to find some topic from the course that can be applied in this area. If there is a valid reason, they may form teams of two or three people who will work on the same project.

The term project is crucial to the course. Students must integrate many of the topics discussed during the term in this one major effort. The most difficult aspect of this assignment for many is the choice of a topic. The most common error made is the selection of too broad a subject area. Students learn, as they specify their problems in precise terms, how to focus upon the central issues of their problem areas.

COURSE EVALUATION

The most impressive feature of the course is the creativity and sophistication demonstrated on a large number of the term projects. Undergraduates are capable of facing systems problems which until recently were not discussed except in graduate courses. The problem solving capabilities of these students have expanded due to the computing techniques they have learned.

It is clear that students are able to implement more complex models in simulation languages than in general algebraic languages. However, they learn more about the fundamental properties of simulation languages by implementing several basic features in FORTRAN, ALGOL, or PL/I and then learning a specific simulation language which is designed for the type of problem in which they have the most interest.

The average expenditure per student for computing resources for the course is approximately one hundred and thirty dollars of which sixty dollars can be allocated to term project expenses. For purposes of comparison, the introductory course in programming and problem solving requires about thirty dollars per student and the advanced programming course requires about one hundred and eighty dollars per student.

If you are interested in more information about this course, please write to the Department of Computer Science at Carnegie-Mellon University.