—CESL—
The Chemical Engineering Simulation Laboratory

DAVID A. KOFKE, MARC R. GROSSO, SREENIVAS GOLLAPUDI, CARL R.F. LUND
State University of New York at Buffalo • Buffalo, NY 14260-4200

Engineering and science *research* today are conducted within an emerging paradigm in which theory, experiment, and computer simulation play distinct but equally vital roles. Progress is often made in leapfrog fashion as each leg surmounts hurdles that have stalled the other two. Thus application of one technique never diminishes the role of the others, but rather enhances them.

The situation in the realm of science and engineering *education* is somewhat less advanced. Undergraduate instruction for decades has relied on a two-pronged approach of classroom and laboratory experiences. Classroom lectures convey concepts, while laboratory provides the students with physical experience—it exposes them to valves, gauges, flowing fluids, and generally, real-life operating equipment. Laboratory also teaches the students how to perform and analyze experiments, and well-designed laboratory exercises teach them how to plan experiments as well. Laboratories teach the limits of experiments, analysis of error, the importance of significant figures, and application of the models presented in the classroom.

Classroom and laboratory experiences are each irreplacable components of undergraduate engineering education. Nevertheless, they have shortcomings:

- *Space, safety, cost, and time considerations restrict the choice of laboratories*
- *Class sizes often preclude direct participation by students*
- *Laboratories must be maintained*
- *Class demonstrations are operated by the instructor*
- *It is difficult to make laboratory experiences substantially different for each student*
- *Lecture examples and homework often sample only a "small corner" of parameter space*
- *Laboratories are difficult to disseminate; the mere description of a well-designed lab does not suffice for someone else to implement it.*

A more fundamental drawback of the classroom and laboratory is their uncertain ability to instill physical *intuition* (as opposed to physical experience, which laboratory does well). Rarely is the laboratory a truly interactive exercise. The student conducts a series of pre-planned experiments and heads home to perform the analysis. This experience does not leave the students with an intuitive feel for the nature of the process. Likewise, classroom instruction is interactive in the sense of instructor-student, but it is not in the sense of student-process; classroom instruction is akin to teaching bicycling through the use of force and torque balances.

Shortcomings of the classroom and the laboratory can be alleviated through proper use of computer-based instruction. In chemical engineering, substantial progress has recently been made in this direction. A group at the University of Michigan[1] has produced a set of tutorial modules that addresses topics across the chemical engineering curriculum. This we view as a valuable tool directed at the shortcomings of the classroom. Software that focuses on the laboratory also exists. In particular, a group at Purdue University[2] has created a suite of modules that lets students perform pilot-

David A Kofke *is Associate Professor of Chemical Engineering at SUNY Buffalo. He earned his PhD in chemical engineering from the University of Pennsylvania and his BSChE from Carnegie-Mellon University. His research interests are in molecular thermodynamics.*

Marc R. Grosso *served as manager of the CESL project. He earned his PhD in Learning and Instruction from SUNY Buffalo in 1994. He also holds an MSE in Computer and Information Science from the University of Pennsylvania, a BS in Information Systems Management from Buffalo State College, and BSEd and MA degrees in Secondary Education. His professional interests are in the application of computing in instruction.*

Sreenivas Gollapudi *holds a MS in chemical engineering from SUNY Buffalo and is presently pursuing a MS in Computer Science. His BS degree in chemical engineering is from IIT Bombay. His research interests are parallel systems and multimedia.*

Carl R.F. Lund *is Associate Professor of Chemical Engineering at SUNY Buffalo. He earned his PhD in chemical engineering from the University of Wisconsin and his BSChE from Purdue University. His research interests are in catalysis and reaction engineering.*

CEE 30(2) Spring 1996

scale laboratories on the computer. Additionally, a host of packages has been developed for more specialized topics in chemical engineering, such as process control. Industrial simulation packages (*e.g.*, Hysim, Aspen) are used routinely and effectively, although this software has not been developed with an eye toward pedagogy. We feel that the potential of simulation as a tool for education is largely unfulfilled.

Indeed, the recent literature in engineering and science education journals has highlighted the tremendous potential of computers as a pedagogical tool, while at the same time lamenting the degree to which this potential is not being met. Seider[3] (prior to the efforts at Michigan and Purdue) noted that in chemical engineering, instructional computing has kept pace with the profession only in the areas of process design and control. Many authors[3-6] have noted two obstacles to the complete integration of computers within the engineering curriculum: the absence of powerful but inexpensive computers with strong graphics capabilities and the high cost (in terms of faculty time) of software development. The steady improvement of computing hardware has made the former a problem no longer. The latter obstacle is our target.

WHAT IS CESL?

At SUNY Buffalo we have developed a detailed plan and completed early development of software that enables education via simulation; we call our package CESL (pronounced "Cecil"). Seven department faculty have been active on this project (these include, in addition to two of the authors of this report, Scott Diamond, Johannes Nitsche, T.J. Mountziaris, Tom Weber, and Mike Ryan). CESL is designed to perform three functions:

- **It is an authoring tool—CESL provides instructors with the ability to construct simulations with relative ease.**
- **It is an environment for conducting simulations—CESL permits the student to explore a process with a minimum of unnecessary effort.**
- **It is an instruction and class management tool—CESL allows the instructor to monitor, guide, record, and sometimes restrict the student's actions.**

While all these features are inherent in its design, to date CESL has been developed only to a level that has permitted three prototype simulations to be implemented. In particular, many of the capabilities related to classroom use are designed but not yet coded.

SIMULATION

Once one considers simulation as part of the educational paradigm, one begins to realize how naturally and substantially it complements the laboratory and classroom experiences. As a simulator, CESL is much more than a simulated laboratory; it does more than just port the traditional lab to the computer (although it can be used in that way too). There are obvious features such as time compression or expansion, a unique laboratory for each student, etc., enabled by simulation. But beyond this are more the pedagogical features of

Many authors[3-6] have noted two obstacles to the complete integration of computers within the engineering curriculum: the absence of powerful but inexpensive computers with strong graphics capabilities and the high cost (in terms of faculty time) of software development. The steady improvement of computing hardware has made the former a problem no longer. The latter obstacle is our target.

- *Random events can be programmed to occur, to which the student must respond appropriately; less dramatic but just as useful, equipment can be programmed to "age" as it is used.*
- *Students can be quizzed in a number of ways, and their responses can be recorded; this can occur before, during, or after the simulation, and subsequent operation of the simulation can be based on their performance (e.g., the student cannot begin until satisfactorily completing a pretest).*
- *Simulations can be conducted intermittently over several days, with the student being given a fixed period of time to return to the simulation; this time can be used by the student to reflect on subsequent actions.*
- *Access to a simulation can be restricted; likewise, the student may be given a fixed number of practice runs before conducting one or more runs for grading.*

No doubt many more novel features can be conceived. Our goal in developing CESL is to enable the simulation author to program and implement these features and the instructor to use them.

Clearly, there is great potential for diversity in design of simulations. It is helpful then to have an organizing principle when considering the options. We have identified the following four categories of simulation:

Laboratory • The student is presented with a piece of (virtual) equipment, or an entire process, and he or she conducts "experiments" on it to characterize its operation. This simulation is meant to mimic as closely as possible an actual laboratory experience; it is the mere simulated laboratory.

Steady-state simulation • The student must choose conditions that optimize the operation of some equipment when run at steady state. The parameters under which the simulation proceeds vary through the course of the experience, building an intuitive sense of cause-and-effect. The student is given a period of time (ranging from minutes to days, per simulation design) to reflect on each action.

Unsteady-state simulation • The students operate equipment in "real" time (which may be compressed or expanded time, if needed). They must respond to regular or random changes in process operating conditions, relying mainly on an acquired intuitive "feel" for the equipment's operation. Many actions are demanded of the student, each by itself being of small consequence, but together adding up to success or failure (perhaps catastrophic) in operating the equipment.

Design • The student is given equipment, or a budget with which to "purchase" equipment, and must assemble, test, and operate a process that is in some sense optimal.

Presently, only the first three categories of simulation can be constructed with CESL, and we have developed a prototype module for each: a simple tank-draining *laboratory;* a pump-sizing *steady-state* module; and an *unsteady-state* continuous stirred-tank reactor (CSTR) module. We will describe the last of these prototypes.

THE PROTOTYPE CSTR MODULE

The display for the prototype CSTR module is presented in Figure 1 (the actual display is in color). Series-parallel reactions take place within the reactor:

$$A + X \rightarrow B + Y$$

$$B + X \rightarrow C + Y$$

The reactions are exothermic and the reactor is not isothermal. Reactant X is presumed to be a gas that dissolves very rapidly in the liquid-phase reaction medium, so that the dissolved concentration of X is always proportional to its feed partial pressure (*i.e.,* Henry's law is obeyed). Normally, the supply pressure of X is essentially constant (though there may be small fluctuations). The other reactant, A, is supplied from three tanks: one containing A in relatively high concentration, one containing A in "medium" concentration, and one containing A in "low" concentration. Deliveries are made at random times to replenish the three tanks.

The students must operate the system attempting to maximize the yield of the intermediate product, B. At the same time, they must prevent any of the tanks from overflowing and must keep the reactor temperature under control. The students can manipulate the flow rate leaving each tank, the feed pressure of X (which must be less than or equal to the supply pressure), the flow of steam to the reactant preheater, and the flow of coolant to a coil within the reactor. A reactor quench can be used in an emergency if the student needs it. As already noted, the simulation will provide random deliveries to the feed tanks. The concentration and temperature in the tanks may fluctuate slightly due to these deliveries and seasonal conditions. Other potential problems that the simulation may invoke include a reduction in or loss of steam pressure for the pre-heater, a reduction in or loss of supply pressure of reactant X, a loss of cooling water flow or increase in cooling water temperature, and a gradual decrease in the heat transfer coefficients for the two exchangers. Because the reactions are exothermic, the student will need to exercise care whenever the feed concentration of reactant is increased or a thermal runaway may occur. Similarly, a large decrease in feed concentration may result in a significant temperature drop and thereby a loss of conversion.

The module is designed to develop within the students an intuitive feel for how conversion, yield, selectivity, and outlet temperature (call them response variables) are affected by changes in operating variables (feed composition, feed temperature, feed flow rate, and heat exchange) for series-parallel reaction networks. There are several stages or levels,

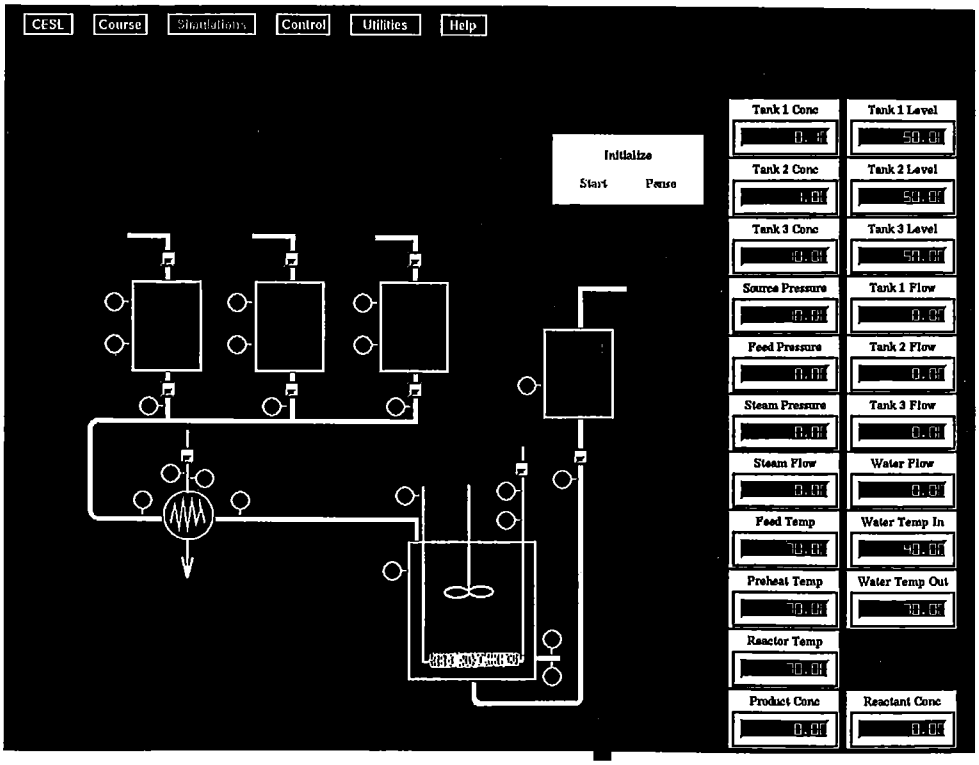

Figure 1. *Display presented to student while operating CSTR prototype module. The actual display is in color.*

of attaining such an intuition. The module allows the student to progress through these levels.

At the most elementary level, the student intuitively knows which operating variables to change and whether to increase or decrease them, in order to effect a specified change in one of the response variables. Often, a desired change can be brought about by manipulation of more than one variable. At the next level of intuition, the student knows which of the operating variables will be most effective in bringing about the desired response (*i.e.,* he or she knows which operating variable will cause the least change in the other response variables). At a yet higher level, the student knows how all the responses will change (at least direction and qualitative magnitude) when a given operating variable is changed.

At a still higher level of understanding, the student can explain why the system responds as it does to a given change in operating variables. Here the student should be able to formulate the explanation lucidly without the use of equations and mathematics. Finally, the ultimate objective of the module is that the student knows how all the above would differ if other parameters of the reactor were changed (*e.g.,* if the reaction was endothermic instead of exothermic, if the kinetic order of one or the other of the reactions increased or decreased, etc.).

OVERVIEW OF CESL DESIGN

A schematic of CESL and its role in the development and implementation of simulation laboratories is presented in Figure 2. In the upper-left corner of the figure is the process to be simulated; perhaps it is too large, dangerous, expensive, etc., to expose to the student. CESL comprises the elements within the gray-shaded region. The white-on-black components have not been implemented (or even designed) in the present version of CESL, but they will be developed as part of future work.

The simulation author is responsible for identifying the appropriate model for the physical system and for programming it using an established language (let us say that this is done in FORTRAN). We delegate this task to the module author for several reasons. First, quantitative modeling and programming form part of the undergraduate and graduate training of chemical engineers, so an instructor should have some competence here, at least for sufficiently simple experiments. Second, general and robust process simulators already exist, so any efforts expended by us in this direction would be inefficiently placed and thus detract from the important task of developing the novel features of CESL. Third, by making the model and simulation code separate from the core of CESL, we introduce a large element of flexibility in a

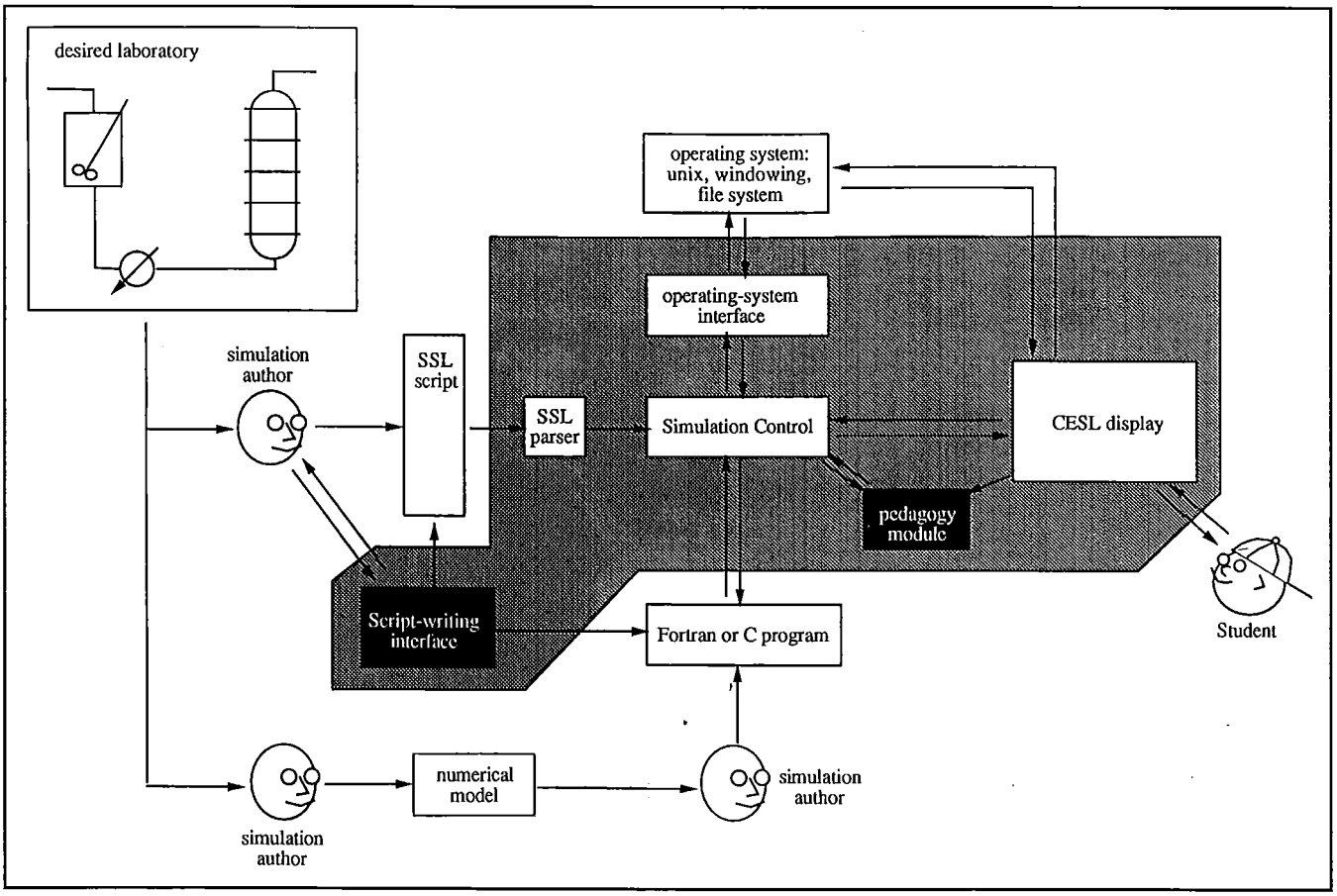

Figure 2. *Schematic of simulation laboratory and CESL's role in implementing it.*

module, and indeed in CESL itself. Instructors may modify or even replace the modeling code (presumably to improve it) while retaining the general simulation design and interface. More significant, extension of CESL to disciplines other than chemical engineering is well facilitated by this design.

The simulation author is also responsible for creating a graphical display through which the student interacts with the modeling code. Our goal has been to make this task as simple as possible. To this end, we have devised the *Simulation Script Language* (SSL), a declarative language through which the module author "equips the simulation" and specifies rules concerning how the equipment may be used and how it interacts with the numerical modeling code. The module author prepares the script using any text editor. The script is parsed by CESL when the student calls for the laboratory to be loaded into the system, and the "Simulation Control" element of CESL is thereby programmed with the laboratory. The Simulation Control element interfaces with the FORTRAN modeling program (to gather data and make sure that solution of the numerical model is proceeding synchronously with the wall clock), the operating system (*e.g.*, to record data to file), and the CESL interface (through which the student conducts the simulation).

There are two elements presented in Figure 2 that were not needed to implement the prototypes but which are important to the ultimate success of CESL. First is a "pedagogy module," which monitors the activity of the student and reports to the Simulation Control the actions needed to improve the student's understanding of the lesson. Second is a script-writing interface. In simplest form, this interface will enable the author to prepare the SSL script using mouse-oriented actions; it will also guide the author in creating modules that are pedagogically sound. We plan to incorporate these features over the coming years.

MODULE WRITING

The SSL is a declarative language comprising a set of keywords and qualifiers that the module author uses to construct a simulation. Declarations may be categorized into the following three types:

Object *statements declare "variables" and place corresponding graphical elements on the screen; these graphics can display or allow user-specified changes to the value of the variable. Simple examples include a temperature gauge or a valve that may be opened and shut.*

Procedure *statements declare the numerical routines that model the system's behavior. Included in these statements is a specification of the object-declared variables that are passed to the routine, and when or how often the routine is called. Routines may be called at fixed points in the experiment (e.g., immediately after the student initializes the laboratory), at regular intervals (of 0.1 sec, for example, if the routine is integrating unsteady equations in time), at random intervals (to cause random events to which the student must respond), or at the behest of the student (by clicking on an appropriate graphic button). As programming the routines is completely up to the module author, they can make anything happen (e.g., a pressure loss is programmed by having the routine simply set the appropriate pressure variable to the newly desired value).*

Controls *declare restrictions and monitors of student's actions. The design and implementation of these features is in an early stage.*

There are only two basic conceptual matters that a module author must grasp to construct a laboratory. The first deals with how CESL, the student, and the modeling routines change and communicate values of the laboratory parameters (*e.g.*, temperatures, flow rates, status of valves). This is done using the "shared memory" concept. The idea is simple: there is one "official" repository of all parameter values, and they may be accessed or changed at any time by CESL, the student, or the modeling routines. Thus, once the modeling routine has computed a set of updated values (perhaps by completing a time-step calculation), it makes a simple call to a library routine that updates the shared-memory values.

The second conceptual matter concerns how CESL keeps in sync with the wall clock (an issue only with Laboratory and Unsteady-state simulations). The SSL script specifies how often a procedure is to update process variables. After computing its values, the routine suspends itself (again using a simple library call), until restarted by CESL (after a period of, say, 500 msec). When restarted, the values in shared memory may have been altered (*e.g.*, a valve may have been shut off). When the routine next uses such values, it will produce results that reflect the changes. In particular, while the routine is suspended, CESL can update the "time variable" using the system clock. Thus the routine can be programmed to blindly update its variables to whatever time it reads from shared memory, without any concern about whether or how that time matches the wall clock.

The simple calls to routines that read and write shared memory, or suspend subroutine execution, are the only additions that the FORTRAN-routine author must include to interact with CESL. Everything else is familiar and standard.

PLANNED FEATURES OF CESL

CESL is a work in progress, and the following features have been designed in some detail but not yet incorporated in the software:

- *Experimental error may be introduced to an arbitrary extent and in two ways: the first is what we call "gauge error," and it describes the simple addition of normally distributed stochastic noise to the values reported to the student; the second is what we call "fluctuation," and it involves random perturbations to the process variables themselves. In contrast to gauge error, fluctuations are propagated through the system. They may in fact be viewed as part of the model that describes the*

physical behavior.

- *Any process variable may be alarmed, with setpoints specified by the module author or the student, and with notification made audibly or via a visual indicator. It may prove interesting to observe which variables the students decide to alarm.*
- *Any process variable may be subject to automatic control using a PID 'device' that is tuned by the author or the student.*
- *The variability of the simulations is easily controlled; each student may be provided with a unique piece of equipment, or all students may be presented with the same equipment, or either of a pair of pieces of equipment, etc. Equipment may also be programmed to 'age,' with its operating characteristics changing in an appropriate way as it get older.*
- *The instructor may schedule the "availability" of the (virtual) equipment, restricting its use to, say, a particular one-week period. Also, the number of practice and grading runs may be specified, along with separate time periods for each.*
- *Data output by CESL for analysis by the student may be presented in any of several pre- or student-defined formats. The predefined formats are chosen to make them suitable for immediate input to popular graphing and analysis programs. This specification reflects a general design principle of CESL to exploit pre-existing software to the fullest extent possible. We do not wish to re-invent software that already exists and functions well.*

Phillips,[5] Koper,[7] and Wankat and Oreovicz[8] each emphasize the importance of the team approach to educational software development. Phillips notes the need for both curriculum and computer specialists on such a team, and Koper stresses the additional role of the educational technologist. In addition to the expertise offered by computer science majors and over half of our department's faculty, we have recently recruited to the project experts in education technology (Prof. Thomas Shuell of our Graduate School of Education) and human-computer interfaces (Prof. Valerie Shalin of our Department of Industrial Engineering). Their impact will be felt particularly in our subsequent efforts.

An interesting application of CESL concerns the development of new modules. We plan to offer to our students, in the form of an elective Projects course, the opportunity to develop new modules that could be used for instruction of subsequent classes. As part of this project, the student will be given the task of creating a working module. This will entail the concept for the module, considering carefully the instructional goal (provided by a faculty advisor), design of the module, writing of the script, programming the model, and testing the product. In this manner CESL will provide to the student a unique experience in pedagogy and design that simply could not be offered by other means.

DEVELOPMENT PLATFORM

We have chosen to develop our software on a Unix platform. We have been careful to employ development tools for which there exist industry standards. Thus all of our code is written in ANSI-standard FORTRAN and (predominantly) C. We use the X-windowing system because it is widely portable and freely available. Because CESL itself interacts very well with the Unix operating system, we can readily introduce file-handling and classroom-management features that will underlie many of CESL's capabilities. This capacity also will facilitate the introduction of pedagogical functions that contribute to the realization of a complete computer-based instructional environment.[9] For example, a record of student achievement and errors can be designed and maintained, allowing CESL's activities to be tailored to the student's progress.

Alternatives to our choice include the use of the C++ programming language and the traditional personal computer platforms. C++ is object-oriented and thus very well suited to our needs, so we are giving serious consideration to its eventual use. There is, however, no present ANSI standard for this language, and it is not as widely available or portable as C. The Macintosh and PC platforms are appealing because of their wide availability. These platforms are capable of running Unix and X-windows, so our present approach does not preclude porting to them.

ACKNOWLEDGMENTS

CESL was developed with the support of a Leadership in Laboratory Development grant from the National Science Foundation (DUE-9352500) and from the SUNY Buffalo School of Engineering. We wish to thank both Mr. Rich Alberth for very important contributions during CESL's formative stages and Dr. Nitin Ingle, who provided programming assistance to the project. Finally, we thank Sun Microsystems, Inc., for substantial equipment discounts and other support, and Mr. Corky Brunskill and his staff for their many contributions to our efforts.

REFERENCES

1. Fogler, H.S., and S. Montgomery, "Interactive Computer Modules for Chemical Engineering Instruction," *CACHE News,* **37**, 1 (1993)
2. Squires, R.G., G.V. Reklaitis, N.C. Yeh, J.F. Mosby, I.A. Karimi, and P.K. Andersen, "Purdue-Industry Computer Simulation Modules: The Amoco Resid Hydrotreater Process," *Chem. Eng. Ed.,* **25**, 98 (1991)
3. Seider, W., "Chemical Engineering Instruction and Computing: Are They in Step?" *Chem. Eng. Ed.,* **27**, 134 (1988)
4. Shacham, M., and M.B. Cutlip, "Authoring Systems for Laboratory Experiment Simulators," *Computers Educ.,* **12**, 277 (1988)
5. Phillips, W.A., "Individual Author Prototyping: Desktop Development of Courseware," *Computers Educ.,* **1**, 9 (1990)
6. Carnahan, B., "Computing in Engineering Education: From There, To Here, To Where?" *Chem. Eng. Ed.,* **25**, 218 (1991)
7. Koper, R., "Inscript: A Courseware Specification Language," *Computers Educ.,* **16**, 185 (1991)
8. Wankat, P.C., and F.S. Oreovicz, *Teaching Engineering,* McGraw-Hill, New York, NY (1993)
9. Wenger, E.L., *Artificial Intelligence and Tutoring Systems,* Morgan Kaufmann, Los Altos, CA (1987) ❐