COMPUTER-AIDED INSTRUCTION

Learning engineering, as a student or as a practitioner, has always required intense participation by the learner. Engineering faculty traditionally have assigned homework problems, conducted problem sessions, answered questions, and tutored students one-on-one, and of course given quizzes, all as means of providing feedback for their students, and thereby increasing the intensity of interactivity in the learning environment. Despite the recurring impression that laboratories mainly develop practical technical skills in students, the fundamental purpose of laboratories arguably is to provide students with an intensely interactive learning environment in which feedback occurs realistically and, often, immediately.

Although these traditional routes to providing interactive learning environments have proved effective (some more than others), it is easy to recognize two difficulties. First, most of these interactive approaches, especially those that provide prompt feedback, require the learner and the teacher to meet together for the approaches to be effective. This requirement for spatial and temporal coincidence of learners and teachers is particularly inconvenient because much, if not most, learning in typical engineering courses occurs outside the classroom and beyond the presence of the teacher. A second difficulty, not independent of the first, is that providing extensive interactivity to students is labor intensive and hence quite expensive.

In view of these constraints, engineering educators began to envision how to provide on-site on-demand interactivity through the use of computers almost as soon as practical computers became available. Looking into the future of engineering education in 1962, for example, W. L. Everitt, Dean of Engineering at the University of Illinois, foresaw the time when every student and practicing engineer could access interactive and adaptive learning environments through inexpensive personal computers of moderate capacity connected, as needed, to more complex computers by wire or radio (1).

Dean Everitt's vision was not developed in isolation. In December 1961, the *IRE Transactions on Education* had published a special issue, guest edited by Mager (2), on *automated teaching* with contributions that addressed the questions, "Why Automate Instruction?" and "How Effective Are the New Auto-instructional Materials and Devices?" and presented "A Rational Analysis of the Process of Instruction" and "A Method for Preparing Auto-instructional Programs" (2). These questions and issues remain contemporary. Doubtless, Dean Everitt was influenced especially by the programmed logic for automatic teaching operation (PLATO) project on his own campus, led by Professor Donald L. Bitzer in the Department of Electrical Engineering (3). In 1960, Bitzer demonstrated the first version of PLATO as one terminal connected to Illiac I (2–6). The initial stated purposes of the project were to investigate the potential role of the computer in the educational process and to design an economically and educationally feasible educational system. Bitzer and his colleagues were therefore among the first to address what remains the fundamental question in computer-aided instruction: "How can we use computers to improve education effectively and inexpensively?" (In this article, *computer-aided instruction* is a broad term that encompasses almost any reliance on computers and networks in learning environments. It encompasses such terms as computer-assisted instruction and computer-based learning.)

From the beginning, PLATO served both on-campus and distance learners. By 1975, the PLATO IV system consisted of over 900 terminals at 146 different sites, some across the United States and Canada and some on campus. The PLATO terminals consisted of a special transparent plasma display panel (512×512 dot matrix) with a touch screen that could superimpose computer-generated output on photographic slides or movies projected through the rear. Audio and even laboratory apparatus could be incorporated, as well. PLATO

J. Webster (ed.), *Wiley Encyclopedia of Electrical and Electronics Engineering*.

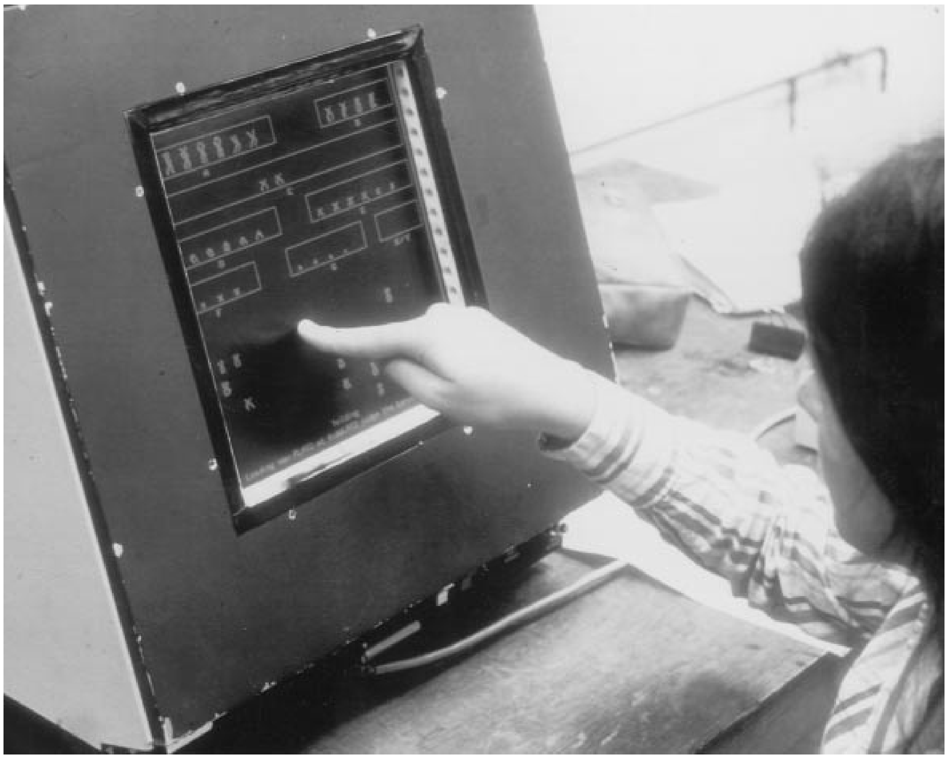

Figure 1. A learner interacts with the seminal PLATO network (PLATO IV) through a graphical interface and a touch screen, circa 1975. (Courtesy of Prof. Donald L. Bitzer.)

IV, funded in part as a demonstration project for computer-assisted learning by the National Science Foundation (NSF) and commercialized by Control Data Corporation, ultimately offered approximately 8,000 h of instructional material, prepared by about 3,000 authors, in subjects that included electrical engineering, computer science, classical mechanics, accounting, astronomy, geometry, biology, chemistry, algebra, foreign languages, law, medical sciences, library science, agronomy, and elementary reading (Fig. 1).

The scope of the PLATO vision is indicated by the proposal for 1,000,000 terminals, most in elementary and secondary schools, in a PLATO V system by 1980–81. To attract enough users to be economically feasible, PLATO V was to include e-mail, on-line library card catalogs, graphics, and games, as well as access to on-line computation and interactive learning environments. As far as functionality is concerned, the PLATO V proposal resembles an early version of the World Wide Web (WWW), though confined to a few mainframes and, therefore, tiny in size by comparison and more starlike in its connective topology.

PLATO V was never implemented, however, because of two evident concerns. The simplest to address is cost. From very early in the PLATO project, the goal had been to provide learning through the use of computers at costs comparable to the cost of classroom instruction. The PLATO strategy was to employ centralized mainframes for the necessary computing power and rely on relatively simple terminals, connected to the mainframes by high-speed telephone links, to give students access to the PLATO learning environments. This strategy sought to exploit the economies of scale available at that time in purchasing and maintaining large computers in comparison with those costs for smaller computers. The PLATO strategy became less appropriate, however, as communication costs proved larger than expected and minicomputers, first, then personal computers undermined the cost and capability advantages of mainframes. Simply stated, PLATO proved to be too expensive.

The second concern was that lessons for use on PLATO, and another large computer-aided learning NSF demonstration project known as time-shared, interactive, computer-controlled information television (TICCIT), took too long to prepare. When completed and implemented, most failed to achieve the results anticipated for interactive computer learning environments (5,6). Despite notable exceptions (6,7), much of the educational material developed for PLATO and TICCIT used the computer mainly to check specific answers entered as a learner moved along an inflexible learning path. Such a format, mainly drill and practice, was straightforward to program, but achieved little of the complex interactive learning environments good teachers were accustomed to providing, albeit labor intensively, for their students. For their part, learners found much of the software for PLATO and TICCIT boring.

The most worrisome problem in developing computer-aided instructional materials for PLATO and TICCIT was that psychological and educational theories at that time gave little guidance about how to use computers to construct intensively interactive learning environments. The disappointing results from these large and well-funded projects, as well as disappointment in the richness of the educational materials they produced, was thus doubly frustrating because the available theoretical framework offered few suggestions about how results might readily be improved.

After the reported investment of more than $900 million by Control Data Corporation, PLATO never became profitable (5). PLATO nevertheless represented a grand vision of what might be. Some forces and concepts it spawned and the continuing elaboration and development of its vision were central to developments in computer-aided instruction decades later. Some, for example, point to the interaction between thousands of PLATO authors and the PLATO staff as the first sizable on-line community (8).

As far as computer-aided instruction is concerned, however, the main legacy of PLATO (and TICCIT) is the beginning of focused and continuing efforts to address the fundamental question, "How can we use computers to improve education effectively and inexpensively?", still the central question for computer-aided instruction. The question never has been answered definitively and, even if it had been, the answer would have been short lived, outdated by rapid developments in technology and in the improving understanding of the nature of learners who use computers in the hope of better progress. Grappling with a question whose answer changes constantly is a task not all would favor to address. A confluence of economic, social, and political factors, however, makes the application of computers in education increasingly inevitable and therefore makes addressing the fundamental question, first addressed by the PLATO workers, inevitable as well (9,10). Contemporary approaches to the fundamental question, fortunately, benefit considerably from the very factors, developments in technology and improving understanding of learners, that drive changes in the answer. The following sections explore developments on these fronts.

LEARNING ENVIRONMENTS ON DESKTOP COMPUTERS

The emergence of desktop computing can be viewed as a firestorm much like the one for automobiles and roads initiated by Henry Ford's Model T automobile. Early desktop computers relied upon inexpensive mass-produced components to achieve modest functionality at low cost. Just as people often preferred independent control of a personal automobile to typically speedier mass transportation, people often preferred a desktop computer that they controlled to the inconvenience of accessing the power and speed of a mainframe. The firestorm effect occurred as increasing demand for desktop computers drove prices down and functionality up, which in turn attracted numerous talented authors of software whose software increased demand even more.

Perhaps the most important impact of desktop computing on computer-aided instructional was to increase dramatically the number of workers and hence the amount of experimentation. The availability of increasingly powerful and inexpensive desktop computers led to almost startling penetration of computing into industries and universities, as well as elementary and secondary schools and homes. The sheer number of desktop computers, far larger than the seemingly incredible 1,000,000 terminals in the unimplemented PLATO V proposal, presented unprecedented opportunity for computer-aided instruction. However, development of computer-aided instruction with desktop computers, beginning in the late 1970s near the end of the PLATO and TICCIT projects, also benefited little from psychological and educational theories as far as guidance about how to construct effective learning environments with computers was concerned. As a consequence, the learning environments produced for desktop computers suffered basically the same complaints and criticisms as those directed at mainframe learning environments: they were either ineffective or effective only unpredictably, they were mainly drill and practice and hence neglected a large number of important interactive learning possibilities and, perhaps most damning, both students and teachers often found them boring once the newness wore off.

Earlier efforts to strengthen the theoretical foundations of computer-aided instruction continued and gained momentum with the participation of more people, attracted by the tantalizing possibilities of desktop computers (11). In practice, much of the work that produced materials for computer-aided instruction on desktop computers, especially in engineering, simply ignored theory and relied instead on authors' intuitions, not necessarily a bad approach under the circumstances. In engineering education, two broadly applicable tactics emerged: *drill and practice* and *simulation.*

Drill and Practice

The drill and practice approach, popular during the PLATO and TICCIT projects, uses the computer for direct checking of specific responses entered by learners. The sheer volume of computer-aided instructional material that abuses the drill and practice technique, admittedly of limited power and applicability, has given the drill and practice approach a reputation that is worse than it deserves. Drill and practice exercises often can help learners achieve rudimentary competence in a subject before moving on to its more challenging aspects.

Contrary to abundant folklore, drill and practice exercises need not be boring. In electrical engineering, a drill and practice program that is widely used with good effectiveness is CircuitTutor®, developed by Oakley and marketed by a commercial publisher (12). CircuitTutor integrates drill and practice exercises into tutorials on various aspects of elementary circuit theory, as typically taught in the first course on circuits in an electrical engineering undergraduate curriculum. After selecting the tutorial on writing node equations, for example, learners either can choose to enter numerical answers to a number of problems with parameter sets generated by the computer or can choose a step-by-step approach that begins with their entering the appropriate node equations in symbolic form. In the step-by-step route, the tutorial leads the student to write the equations in symbolic form, to find numerical solutions to the equations and, perhaps, to use the results to find the maximum power that can be supplied by a pair of terminals of the circuit. In the step-by-step approach, incorrect answers must be corrected before proceeding to the next step. Even this brief description of CircuitTutor reveals a degree of flexibility and sophistication that helps account for its widespread acceptance and success despite the inescapable limits of its drill and practice approach.

In a portion of ELECSIM, a learning environment for undergraduate electrical engineering students enrolled in the first course in analog electronics, Marcy and Hagler have extended the drill and practice approach to evaluation of answers to problems that do not have unique answers (13). Specifically, learners enter component values for a specified circuit (such as a voltage amplifier) that cause the circuit to meet computer-generated performance specifications (frequency response, gain, and input and output impedances, for instance) and to satisfy certain design rules available to the learner. In one example, the computer checks that the transistors in an audio amplifier are appropriately biased (according to the design rules), that the open-circuit voltage gain and the output resistance for the amplifier are appropriate for the design constraints, and, again according to design rules, that coupling capacitors are neither too large nor too small. If the learner enters parameter values that lie outside ranges required by the design rules, advisory messages to the learner appear on the screen. Checking the consistency of entries with a set of design rules in this manner requires no more than simple branching constructs in the software.

Simulation

The simulation approach to computer-aided instruction often has relied on simulation software to simplify a complex process so that a learner could concentrate on only a few salient aspects during the learning process. In engineering, laboratory simulations have proved useful in preparing students for physical laboratories, although it is possible to use them apart from a physical laboratory as well. Mosterman, Campbell, Brodersen, and Bourne developed the electronic laboratory simulator (ELS) system, for example, which includes simulations of nine laboratories for a course in electronics (14). Topics range from Thevenin's theorem to operational amplifier implementations of multiple-pole filters. In each laboratory, learners assemble circuits for simulation by dragging and dropping graphically realistic components onto a graphically realistic breadboard. Learners make measurements on the circuit by connecting graphically realistic instruments, such as an oscilloscope, to the circuit. Learners use their mouse to adjust the instrument controls on a close-up image of the instrument. Each laboratory includes an introductory tutorial and context-sensitive on-line help.

More generally, powerful and realistic simulators can enrich the learning environment by helping the student to learn self-assessment of solutions to open-ended problems. The circuit simulator PSpice, for example, is used widely in both industry and academia for simulation of analog and mixed analog and digital circuits. A free evaluation version limited to small circuits is available for use by students on their personal computers. Using PSpice, a student can check the performance of a design attempt against design specifications and revise the design as necessary to achieve the specified performance without direct involvement of the teacher. The learner not only receives immediate feedback about the success of the design (at least within the accuracy limits of the simulator), but also gains invaluable experience in self-evaluation of work on open-ended problems that have no unique solution. The iterative interplay between the learner and the simulator creates a powerful real-time interactive learning environment without real-time participation by the teacher.

Any topic for which a useful simulator is available is a good candidate for interactive instruction in engineering education. Possibilities beyond circuit and logic simulators include any numeric simulation software, as well as compilers (for computer language instruction), which, strictly speaking, are not simulators but indeed the real thing. Symbolic manipulation programs are, in a sense, symbolic simulators in that learners can compare their results from symbolic calculations with the results from the symbolic software. In circuits, for example, a Thevenin equivalent circuit might be derived analytically by the learner and checked with a symbolic manipulation program. If the result is incorrect, then the learner can reprise the derivation until the disagreement is resolved. The learners thus take responsibility for assessing and correcting their work, a most valuable and productive activity. Engel developed SPLICE software that combines the schematic capture capability of PSpice and the symbolic manipulation capability of Waterloo Maple V to provide a simple means for learners to obtain analytical answers with which to compare their own results (15). By simply drawing a circuit schematically and selecting a pair of terminals, SPLICE generates mathematical expressions for the Thevenin voltage and impedance of the circuit for those terminals, for example. SPLICE thus provides learners a convenient means of evaluating their own symbolic analyses of circuits.

Graphical User Interfaces and Multimedia

In addition to multiplying the number of people exploring the use of computers in teaching, the emergence of powerful inexpensive desktop computers with graphical user interfaces (GUIs) and multimedia capabilities has transformed the learner interface to computers. The learner interaction offered by contemporary desktop computers was simply unthinkable during PLATO's heyday. Furthermore, the extensive application of desktop computers for presentations in the business community has stimulated the development of powerful and inexpensive software tools for easy creation of complex interactive presentations. The difficulty is that this power can be dangerous for creators of learning environments. Just as in the early days of desktop computing the newfound ability to experiment with countless fonts led to rampant publication of nearly unreadable newsletters in the proverbial ransom note font, the multimedia capabilities of contemporary desktop computers can ruin the effectiveness of learning environments by confusing and distracting, rather than supporting, both creators and learners. Avoiding such pitfalls in developing a learner interface is a matter of designing to provide as much richness as the learning environment can support without confusing the learner. Fortunately, principles of graphical user interface design have emerged that can guide the development of effective interfaces (16,17). Perhaps the best simple rule in interface design is to make an interface to the learning environment that focuses the learner's attention on the learning environment, not the interface. If the interface either distracts the learner with unnecessary bells and whistles or obstructs attention by frustrating or confusing the learner, then the learner interface is a failure and must be reworked.

Electronic Books and Tutorials

Electronic books and tutorials for desktop machines typically rely on some combination of drill and practice, simulation, and multimedia to provide learners a considerably more complex learning environment than is possible with printed material. Doering, for example, has combined circuit simulation, video, audio, graphics, and text in a tutorial that helps learners visualize the dynamic behavior of circuits (18). Harger constructed a hyperlinked interactive book in which Mathcad, an inexpensive but powerful and widely used commercial mathematical analysis package, is used to achieve an interactive learning environment for a course in digital signal processing (19). Learners can experiment by modifying the examples and seeing the consequences immediately. Wood developed interactive tutorials for digital logic design, digital signal processing, and engineering mathematics that aim specifically at helping learners integrate conceptual fragments and thereby achieve a deeper understanding of basic concepts (20).

LEARNING ENVIRONMENTS ON NETWORKS

Although a number of successful interactive engineering learning environments for stand-alone desktop computers became widely available, the collection of available environments failed to exhibit anything like the variety in interactive strategies traditionally provided to students by teachers of engineering. On a more pragmatic level, developers of interactive learning environments for desktop computers encountered two inhibiting limitations. First, a learning environment on an isolated desktop machine, or even one on a local area network, could access and incorporate only a limited variety of resources in comparison with a mainframe environment. Second, the mixture of MS-DOS Windows, Macintosh, and UNIX operating systems on desktop computers required preparation of multiple versions of a learning environment if it were to be readily accessible to the majority of desktop machines. These limitations significantly impeded the development of interactive computer-aided instructional environments for desktop computers.

The explosive development of networks, especially the Internet and the WWW, provided a widely available means of addressing the limitations of both isolation and operating system incompatibility for desktop computers. For example, the Internet permits users of Oakley's CircuitTutor tutorial software described earlier to submit homework assignments and quizzes electronically to an Internet server, which automatically runs a grading program and returns the results to the user (12). Bulletin board software permits students, faculty, and teaching assistants to discuss problems and questions related to the class and therefore function as a virtual learning community.

The availability of unlimited access to the Internet at low monthly rates, of inexpensive powerful desktop computers, and of multimedia WWW browsers for all popular operating systems means that an increasing fraction of students and practicing engineers have ready access to the WWW, regardless of the operating system for their machine. The WWW thus provides a de facto standard for writing and providing access to computer-aided learning environments, independent of the operating system. Such a cross-platform standard provides access for a large number of potential users and hence establishes a strong incentive for the development of learning environments (21). The number of potential users on the WWW dwarfs even the erstwhile incredible 1,000,000 terminals envisioned for PLATO V. Moreover, it makes available a variety of resources for incorporation into computer learning environments that is larger even than that available, before, on mainframes. At the same time, the WWW restores the promise, dating from the days of PLATO, of efficiency that results from using the same computer-aided instructional materials to provide on-site on-demand access to complex learning environments for both on-campus and distance learners.

Because of these advantages, WWW-based tutorials and interactive learning environments began to appear in the mid-1990s. Schodorf, Yoder, McClellan, and Schafer, for example, established a WWW home page for a digital signal processing course taken as the first course in electrical engineering by students at Georgia Institute of Technology (22). The home page gives students access to demonstrations with video and audio files, Matlab quiz problems for review and drill, and interaction with each other via a newsgroup devoted entirely to the course. Material from the WWW site for the course, recorded on a CD-ROM that can be viewed with a WWW browser, accompanies a textbook (23).

Sears and Watkins developed a multimedia manual for a telecommunications setup and placed it on the WWW (24). The manual makes extensive use of hypertext markup language (HTML) image maps to permit users to click on a component in a photograph to access, for example, a close-up view of a printed circuit board from a different perspective, as well as obtain further technical details about the component. An interesting feature is the possibility of incorporating on-line information made available by the manufacturer of the equipment.

CyberProf was initiated at the University of Illinois by Hubler in 1994 with the notion of updating and expanding the functionality of PLATO through WWW technology and a more robust and intelligent human-computer interface based on complexity theory (25). Specifically, CyberProf is designed to support teachers' efforts to integrate lecture notes, laboratories, and homework into a cohesive package. Students, who access the learning materials with a standard WWW browser configured to accommodate common multimedia file formats, can solve problems and receive immediate feedback "from a sophisticated grading package that makes use of the latest complex systems data analysis tools to handle ambiguous input in an intelligent manner." The grading package analyzes the learner's work and determines errors in arithmetic, sign, and units and then provides hyperlinks to appropriate help. The grading package also can check symbolic expressions including, for example, differential equations. The pages with problems for students contain hyperlinks to relevant lecture notes and help files. Special tools permit the students to draw images and create animations. Integrated conferencing software provides communication among the students, the teacher, and teaching assistants. Controlled access to an on-line grade book indicates the student's progress. CyberProf provides specialized HTML editors for constructing lecture notes and problem sets. CyberProf is implemented on a server as a package of Perl scripts and C routines that respond to input from students and teachers submitted via ordinary HTML forms. By 1997, courses at the University of Illinois that use CyberProf included offerings in physics, chemistry, biology, and economics, with courses under development in several other areas, including electrical engineering.

In 1997, commercial publishers began to offer on-line course packages for several computer languages. The purchaser of the package, sold at bookstores, receives a textbook, an ancillary CD-ROM, and, at no additional charge, entry to a WWW site that provides access to peer discussion groups, a list of frequently asked questions and responses, a tutor (who answers a fixed maximum number of additional questions for each subscriber), and on-line examinations. Some universities began to offer degree credit for courses offered over the

WWW. In the late 1990s, a few complete degree programs became available over the WWW and offerings were increasing rapidly.

The National Engineering Education Delivery System (NEEDS) maintains a database of noncommercial curricular materials in electronic form on the Internet and hence provides a means of finding and obtaining materials for computer-aided instruction in engineering as they are developed. NEEDS includes material for stand-alone desktop computers and, increasingly, materials for use on the WWW (26).

The World Lecture Hall, maintained on the WWW, contains links to pages created by faculty worldwide who are using the WWW to deliver class materials. Information is classified according to subject and includes course syllabi, assignments, lecture notes, exams, class calendars, and multimedia textbooks (27).

Perhaps the most important conceptual contribution of the WWW to computer-aided instruction is the perspective of an open learning environment in which there are few limits on the types, quantity, or location of materials that can be incorporated. Before the WWW, it was easy to think of learning environments as circumscribed by a few computers on a local network or even by a single machine, but this is no longer true.

APPROPRIATE APPLICATION OF COMPUTER-AIDED LEARNING ENVIRONMENTS

Widespread availability of desktop computers and pervasive access to the Internet and the WWW give teachers unprecedented choice about how to deploy computer-aided learning environments. Once, computers at universities were available mainly in special on-campus computer laboratories. This location limited the use of computers both in the classroom and at home. Access to desktop and laptop computers and to the Internet removes these constraints and presses the recurring question about where and how computers best can aid learning. These dramatic changes in technology, the demands for more student-oriented approaches to learning on campus and indeed fundamental changes in the structure of engineering education, as well as calls for improved opportunities for distance learners, all mean that the fundamental question of computer-aided instruction, “How can we use computers to improve education effectively and inexpensively?”, must be approached with a careful understanding of basic strategies in engineering education. Otherwise, it is easy to lose perspective and, as a consequence, spend lots of time and money to achieve disappointing results.

Responsibility for Learning

Teachers at the college level adopt, intentionally, a fundamentally different strategy of instruction than teachers in secondary school, or high school. In high school, the teacher lays out during class what the students are expected to learn and then directly supervises the learning, most of which occurs also during class meetings. Although work outside class (homework) is a part of high school education, most learning occurs during class. In college, the teacher still lays out during class what students are expected to learn but expects that only a small part of the learning takes place during the class meetings. Teachers expect college students to accept the responsibility for learning most of what they need to learn outside normal class meeting times. This approach is a major reason why college students typically spend less than half the time in class than high school students spend. In college, therefore, the teacher spends much of the class time explaining to students what they should learn. Then the students leave class and learn the material, usually with the help of other students and perhaps with the help of the teacher or an assistant, study groups, and, perhaps, interactive tools such as computer-aided instruction. This approach leads to the oft-stated rule of thumb that college students should spend approximately two hours outside class studying and learning the course material for each hour spent in class.

The principle that underlies this approach is that post-secondary-school students should learn to take responsibility for learning the specified material as a step toward lifelong learning in the real world. In the world of practice, they must assume not only the responsibility of learning whatever they need to know, but of deciding what they should learn as well. From this perspective college serves students as a kind of half-way house between learning in the highly structured high school environment and learning in the real world after they leave college.

Note that focusing on changing engineering education by changing the classroom (for which the need for improvement seems inarguable) misses the largest target for reform: the two-thirds of the course time that students spend studying and learning the material outside class. It is this time outside class that should be a prime target for interactive instructional environments, computer-aided instruction, collaborative learning, teamwork, and many other approaches that focus on the learner. It is, after all, outside class that most learning occurs during college courses.

An obvious rejoinder is that teachers in college should spend less class time laying out the material to be learned and more time conducting learning activities related to the content of the course. Such a change suffers from two difficulties: one pragmatic and one philosophical.

From a pragmatic perspective, encroaching on class time for extensive in-class learning activities means decreasing course content or interfering with other functions of the class meeting, discussed later, unless either efficiency in presenting the material to be learned increases dramatically or the time spent in class increases. Universities are unlikely to invest more resources in providing increased time in class in view of widespread calls on them from many of their customers to increase institutional productivity. Even dramatic gains in efficiency of presenting the material are unlikely to make available enough time to reduce significantly the two hours or so of time students need to spend on learning outside the classroom after each class meeting.

From a philosophical perspective, implementing these learner-centered approaches to a significant degree during class time emphasizes learning supervised by the teacher rather than helping learners accept responsibility for their own learning activities. Directing instructional reform primarily at class meetings, therefore, not only misses the larger part of the course (outside class) during which most learning in college occurs, it also jeopardizes the success of efforts to help students learn to accept responsibility for a lifetime of learning on their own.

The Class Meeting

If most learning in a collegiate course occurs outside the classroom, a logical question is, "Why ask the students to meet together at all?" Perhaps the major attractive feature of a class meeting is the efficiency it affords the teacher in dealing simultaneously with a large number of students. As mentioned earlier, the class meeting permits the teacher to describe to the students, simultaneously, what it is they should learn. In addition, class meetings can offer efficient means of communicating to all students in the class answers to questions of common interest, of administering examinations simultaneously, of collecting student work, and of handling necessary organizational details. The class meeting also provides special opportunities for motivating students through, for example, presentation of supplemental material by the teacher, class participation by the students, and interaction with the teacher. Class participation can range from informal group discussion to the once-popular approach of requiring several students to work example problems, simultaneously, in front of the class and the teacher (as a reasonably efficient, if somewhat intimidating, means of demonstration, correction, and motivation).

The fundamental purpose of the class meeting, however, is to prepare students to learn efficiently after they leave the class meeting. From this perspective, the class meeting is analogous to a staff meeting in the business world: the meeting itself accomplishes little of the work that needs to be done, but permits communication, planning, and coordination that help the participants accomplish their work after the meeting.

From this viewpoint, nothing is fundamentally wrong with the oft-criticized lecture method. Until recent times, the lecture method arguably has been one of the most time-efficient means available to teachers for communicating, within a limited time, to students what they are to learn. Much of the criticism directed toward the lecture method apparently stems from belief that the primary purpose of class meetings is to provide a time for students to learn rather than a time to prepare them to learn outside the classroom. From another perspective, much of the criticism of the lecture method seems to be based on confusion between the supervised learning strategy practiced by teachers in high school and the strategy, used by teachers in college, of helping students (indeed requiring students) to become independent learners.

The recent widespread availability to teachers of inexpensive, and easy-to-use, software for word processing, graphics, presentations, simulation, spreadsheets, symbolic manipulation, and multimedia and that of desktop and laptop computers to students means that placing material to be learned in files on disks or on networks (especially the WWW) permits distribution of more detailed, accurate, and timely information than could possibly be accomplished through the traditional chalk and blackboard use of the lecture method. The traditional lecture method's days as the predominant teaching mode, therefore, are threatened not so much by its lack of interactivity in comparison with collaborative learning and other student-centered approaches to learning, but by its recent relative inefficiency in presenting the material to be learned in comparison with techniques that exploit information technologies. Exclusive use of the traditional lecture method during class meetings is, therefore, increasingly difficult to justify. In particular, it is exceedingly difficult to justify the use of class time for the teacher to transcribe notes on the board as the students transcribe them (incompletely and inaccurately) into their notes. Any modest cognitive benefit learners might derive from the transcription process surely can be outstripped by suitably challenging problems or projects they pursue outside the classroom.

It is important to note, however, that a judicious combination of the lecture method and the distribution of information electronically can permit the teacher to devote much more time during the class meeting to motivating students, discussing relevant issues, and answering questions than use of the traditional lecture method alone. The class meeting time saved also can permit the teacher to introduce some student-centered learning approaches during the class meeting time. The objective in doing so, however, is not to make it unnecessary for students to use these approaches outside the class meeting time, but to improve the efficiency with which students use them outside the classroom.

Accessing and reviewing some, but certainly not all, of the course material located on servers can prove useful during the class meeting in a suitably equipped classroom. Given that most learning occurs outside the classroom, however, investment in classrooms that provide a computer in front of each student is difficult to justify, especially given the continuing expense of maintaining up-to-date computers in such classrooms. Students' computers, in contrast, are a renewable resource.

Learning Outside the Class Meeting

If most of the learning in a collegiate course occurs during the two-thirds of the course that lies outside class meetings, teachers must consider how to design learning experiences for students outside the classroom that help them make the best use of that time. Traditionally, most engineering teachers have paid far less attention to designing, carefully, learning activities for students outside the class meeting time than they have to designing the class meeting time, despite the fact that students are supposed to spend twice as much time on the course outside the classroom as they do within it.

Common approaches to fostering learning by the students outside the classroom include assignments of reading and homework. Unembellished, both of these approaches fail to provide intensely interactive learning experiences that students appreciate and, increasingly, demand. In practice, homework assignments often turn out to be largely a waste of time for both the student and the person who evaluates them. Consider the following situation. The teacher assigns a homework problem, due one week later. The night before the assignment is due, students consider the problem for a time, take a shot at providing a solution and submit it the next morning to see if they got it right. The teacher carefully studies, corrects, annotates, and grades the papers in time to return them to the students during the very next class meeting. By that time, the students have focused their attention on other matters and lost almost all interest in the problem. They probably never read the careful corrections and admonitions provided by the teacher. In short, few students assume ownership of typical homework to the point of committing to self-evaluation of their work. Instead, they make a quick pass at a solution and send it off for evaluation by someone else whose evaluation cannot possibly reach them in time for due consideration.

What students need to improve their efficiency in learning outside the classroom are highly interactive learning environments in which they receive feedback about their work immediately, while their interest is high. How can such environments be achieved in practice? Formal and informal problem sessions have been conducted by engineering educators or their surrogates for years as a means of providing interactive learning environments in which students can ask for, and receive, immediate response to more questions than the teacher can entertain during the class meeting. In contrast to an apparently widespread assumption that laboratory activities are mainly a means of inculcating practical skills in students, the main purpose of laboratories is to provide a highly interactive learning environment in which learners receive immediate feedback about their work directly from experimental apparatus. Especially for electrical engineers, the cost of sophisticated electronic components has become low enough that construction and testing of surprisingly complex circuits and systems assembled from parts that the students purchase at local stores can be required as *hardware homework* that demands no allocation of laboratory space or personnel (beyond a grader for the homework) by their academic department (28). In recent years, engineering educators have begun to encourage formation of study groups among students of a class. Such groups provide a potentially highly interactive, and effective, learning environment at little cost to the institution. One-on-one learning sessions with the teacher, or a hired tutor, can be quite effective in providing a highly interactive learning environment but, unfortunately, at a high cost. Few of these approaches offer as much promise for improving the learning environment outside the classroom as computer-aided instruction. Current widely available computer and network technology afford intensely interactive learning environments at any time on desktop computers almost anywhere and, via networks, offer learners opportunity to interact with the teacher and each other, without the necessity of overlapping exactly in either time or space.

In the mid-1990s, the Alfred P. Sloan Foundation initiated a program specifically directed at learning outside the classroom to explore “new outcomes in science and engineering higher education” made possible by affordable technology, including desktop computers, network access, CD-ROMs, and video tape (29). They began with the perspective that lectures and study groups, and indeed most on-campus learning activities, nowadays could occur without the learners and the teacher gathering at the same place at the same time through e-mail and conferencing, for example. They termed this concept asynchronous learning network (ALN). ALNs are intended to serve both on-campus and distance learners. The Sloan Foundation does not envision asynchronous learning networks to require specially constructed software learning environments of the kind often associated with computer-assisted instruction, but considers ALNs mainly as a means of facilitating connections among teachers and learners. Possible outcomes of asynchronous learning networks include self-paced learning, lower cost to the learners, and pursuit of degrees or certifications at home. The Foundation also is interested in the effects of ALNs on the time required to complete a degree and on student retention.

Implications for Distance Learning

If two-thirds of a typical on-campus course occurs outside the class meetings, the good news for distance learning is that much of the material prepared to help on-campus students learn outside the classroom applies to distance learners as well. Computer-assisted learning environments, in particular, can be readily available and useful to on-campus and distance learners alike. Especially since on-line study groups are becoming common, much of a course designed to provide interactive learning environments outside the classroom for on-campus students should be useful to distance learners with little adaptation.

The main difference for distance learners is in accomplishing the functions that are carried out during class meetings for on-campus students. Specifically, the question is how to accomplish (1) dissemination to the students of the information that they are to learn during the course, (2) communication of answers to questions of common interest to all students in the class, (3) administration of examinations, (4) collection of student work, (5) organization of the course, and (6) motivation of the learners, perhaps through interaction with the teacher and other students.

It is easy to see how the increasingly ubiquitous Internet and WWW (in combination with surface mail, fax, and telephone) can provide all of these functions without re-creating a classroom environment for the distance learner. Indeed, the Internet is profoundly changing distance learning (30). Perhaps providing motivation is the most difficult function to furnish distance learners. Fortunately, distance learners, by the virtue of the fact that they have taken the trouble to become distance learners, usually are highly motivated and may not need as much classroomlike motivation as typical on-campus students. The on-line experience itself provides motivation for some students.

Given the perspective that the class meeting is not the primary occasion for learning during a course and given that most functions of the class meeting can be accomplished by other means, it seems difficult to justify heavy investment in two-way video links to recreate a classroom environment for distance learners. Investment in enriching the environment in which they learn on their own seems more appropriate and productive. Recreating a classroom environment is not only expensive and accomplishes little that cannot be accomplished by alternative means, it also demands that distance learners congregate at a specific time and place—a severe disadvantage for many distance learners. Distance learners seem to view on-site, on-demand, highly interactive learning opportunities as the ideal. With computers on the desktops (or laps) of most distance learners, on-site on-demand, highly interactive learning seems much more nearly achievable via the Internet and surface mail materials such as videos and printed items than via real-time video and/or audio links.

Student and Teacher Responsibilities

Student responsibilities in a course of study are simpler to describe than to accomplish. Student responsibilities are threefold: (1) find out what is to be learned during the course of study, (2) assume responsibility for doing whatever is necessary to learn it, and (3) learn it.

Teacher responsibilities are more complicated to describe and, perhaps, to accomplish. The most visible responsibility of a teacher in a college course is to introduce students enrolled in the class to what is to be learned. This responsibility may be carried out, for example, through delivering lectures, assigning readings in textbooks and supplemental material, distributing handouts, posting files on network servers, and providing interactive computer-aided learning environments. Explaining the material to the members of the class, discussing it, and answering questions are other important responsibilities. A most important related responsibility for teachers is the following: do not assume responsibility for a student's learning. That responsibility must lie squarely on the student to avoid subverting perhaps the most important single objective for college students: learning how to learn without supervision. Another obvious responsibility of the teacher is to evaluate the achievements of students enrolled in the class.

Arguably the most important single responsibility of the teacher, however, is to plan, actually to design, the course of instruction for the class. Although the most obvious part of this responsibility is planning the class meetings, the most critical, most demanding, and most easily neglected part is designing that portion of the course of instruction that involves the students when they are not in class meetings but learning on their own.

Unfortunately, most experienced teachers have the distressing impression that the time students spend outside of class is singularly unproductive. Thus, an important responsibility of the teacher is to design learning activities for students that help them make the best use of their efforts outside the classroom instead of wasting precious time. Indeed, the design of that part of a course that occurs outside the classroom is potentially the most productive single opportunity that is available to contemporary teachers for improving learning by students. Elements available to teachers for designing this part of the course of instruction include, as already mentioned, study groups, student-teacher conferences, problem-solving sessions, hardware homework, interactive computer software, and network communication between students and the teacher.

Faculties of education at universities have long advocated explicitly designing a course of instruction subject to constraints such as time, financial resources, student capability, and accreditation requirements. The general process consists of (1) determining what students already know, (2) deciding what students should learn, (3) identifying specific instructional approaches (lectures, collaborative learning approaches, interactive software, and laboratories, as examples) that may be useful, (4) synthesizing a coherent plan for student learning that exploits these approaches, (5) selecting and/or developing appropriate materials, (6) developing examinations, projects, portfolio requirements, and other means of assessment that measure the effectiveness of the plan in practice, (7) trying in practice what has been conceived, and (8) modifying the course of study, based on the assessments, to improve its effectiveness (31). This process of designing a course of instruction subject to constraints can be conveniently termed *course design* to distinguish it from the term *instructional design,* encountered later.

Although engineering faculties at universities certainly can recognize the parallels between course design and what they term *engineering design,* only a few seem to apply the design process, consciously and routinely, in developing courses of study for engineering students. In the absence of widely accepted theories of learning and instruction, it might seem likely that engineering teachers would adapt familiar approaches from engineering design to develop courses that, subject to constraints, help students achieve such fundamental objectives as problem solving and learning to work with others as a team. Perhaps, however, the relative stability, until recently, of the instructional paradigm for engineering education that emerged and persisted during the decades that followed World War II permitted the design of a course of study to mean little more than ensuring that students are exposed to a minimum set of technical topics. Approaches based on folklore and customs learned during the teacher's student days too often substituted for careful explicit design of courses of instruction. Such a naive approach to designing courses of instruction is always risky, but is especially likely to be ineffective during times of rapid changes in educational approach, student preparation, industry expectations, and student expectations. Careful course design is vitally important as courses rely more on computer-aided instruction, an unfamiliar tool that, without careful forethought and planning, can fail either by running amok or alienating learners.

Widespread application of the course design process to develop courses of study in which the learning activities outside, as well as inside, the classroom form a coherent and effective whole could improve the resulting course designs dramatically by (1) focusing faculty attention on the underlying fundamental learning objectives for their students instead of simply the topics to be covered and (2) incorporating a greater variety of learning approaches in engineering courses than has been customary. Indeed, one answer to the fundamental question, "How can we use computers to improve education effectively and inexpensively?", is to use computers as an excuse to convince engineering faculty to plan carefully the entire course of study, within and without the classroom, as a coherent whole. If such a ploy were successful, computer-aided instruction would have succeeded nicely even in courses in which computers play no explicit role. However, successfully incorporating computer-aided instruction, with the additional variables related to the relatively unexplored attendant networking and multimedia environments, into courses of instruction becomes very hard without some explicit design strategies suggested by theories of learning and instruction.

CREATION OF LEARNING ENVIRONMENTS

The major impediment to progress in computer-assisted instruction is less and less the cost of networking and powerful hardware, but rather incognizance about how to design effective interactive learning environments. From a naive perspective, psychology would provide a theory of learning on which to base a theory of instruction that would prescribe how to design a successful computer-aided instructional environment. Reality is far different. Theories of learning from psychology certainly are available (32). Indeed, a major difficulty is too many theories of learning. Most are rooted in movements in psychology, such as behaviorism or cognitive science. Seemingly endless conflicts appear among the various theories, however, mainly from uncertainties about the domain of their respective applicability. Such uncertainty clearly hampers adequate verification, and widespread acceptance, of the theories. Imagine the conflict and confusion that would result among circuit theorists and electromagnetic theorists if the domain of applicability of each theory were not understood. Circuit theory is a simplification of electromagnetic theory that applies only when the wavelength that corresponds to the highest frequency at which the circuit will be used is much larger than the largest linear dimension of the circuit. Without this knowledge, the violation of Kirchhoff's current law along conductors in antennas, for example, could produce endless confusing arguments.

Doubtless, all of the available learning theories apply under some circumstances. Not understanding what those circumstances are means that discovering conflicting conclusions from the different theories sheds little light on how to improve the theories, or on the question as to whether one or the other is incorrect in some fundamental way. Perhaps because of the seemingly countless, inconsistent, unverified (unverifiable?) theories of learning and instruction, engineers traditionally have been largely ignorant, indeed skeptical, of theories of learning and instruction. As a consequence, the application of theories of learning and instruction to the development of instruction in engineering and science at the university or professional level has received scant attention compared with the development of instruction in kindergarten through grade 12.

Fortunately, theories of learning and instruction can provide considerable insight during course design by engineering educators, despite the absence of a single widely accepted theory. Exploring the available insights and exploiting them in addressing the fundamental question, "How can we use computers to improve education effectively and inexpensively?", however, requires familiarity with the various approaches, vocabularies, and patterns of thought characteristic of these areas.

Learning Theories and the Creation of Learning Environments

PLATO and TICCIT, the large computer-aided instruction pilot experiments boosted by substantial funding from the National Science Foundation and other sources, followed quite different approaches to designing the enormous amount of computer-aided instructional materials that these large demonstrations required. Authors of learning materials for the PLATO project had complete freedom as far as the types of instructional strategies and design procedures they employed (33). A basic assumption initially was that developing computer-aided instructional materials was just a matter of automating instructional techniques commonly used at that time. As a consequence, learning materials developed for PLATO included a diverse combination of drill and practice exercises, simulations, games, and tutorials. Although the basic assumption ultimately proved wrong, the collection of PLATO materials showed that diverse design approaches could yield useful environments and that there is no best approach, even for a particular discipline. Authors also found it convenient that PLATO permitted small modules to be constructed and evaluated easily and then, later, combined with others to form a larger unit. Not surprisingly, results for a specific PLATO lesson depended dramatically upon how the teacher who used the lesson felt about it and upon how the teacher chose to implement the lesson.

In contrast, the TICCIT project, conducted by the Mitre Corporation, used a much more structured production approach to developing large quantities of learning materials. Individual teams consisting of a subject matter expert, a psychologist, an instructional designer, an evaluator, and a packaging specialist designed learning environments that carefully controlled learner activities (5,33). Partly to achieve production efficiency, the project chose, initially, a rules-examples-practice pattern as the single instructional strategy for developing materials designed to ensure that learners mastered the information in the lessons. This approach made it easy to generate templates that permitted new lessons to be developed merely by adding subject matter to the template rather than spending the time and effort to create a new design for each lesson. In TICCIT, again, the attitudes and approaches of the teachers strongly influenced the success of the lessons.

The approach to lesson development in TICCIT clearly was more systematic than that for the PLATO project and hence offered the possibility of further development into a broadly applicable procedure that authors could rely on for guidance in developing computer-aided instructional materials. The TICCIT approach, which ultimately led to what is called *instructional systems design* or simply *instructional design,* relied heavily on concepts from the movement in psychology known as *behaviorism* for its initial development.

Behaviorism. Before the mid-1950s, behaviorism was a dominant force in psychology (5,32). Behaviorists held that psychology should concern behavior without consideration of consciousness or mental models and constructs. Specifically, they maintained that psychology should deal only with prediction and control of observable behavior. Without much oversimplification, behaviorism can be viewed as a reaction against not only the earlier psychological concepts of mind and consciousness, which, to behaviorists, seemed too close to the religious idea of the soul to be the subject of proper scientific inquiry, but also a reaction against Freud's preoccupation with the unconscious, the id, and the libido, which seemed too fanciful to be the subject of scientific research. From investigations with animals in simple experimental configurations, behaviorists came to believe that essentially all human behavior, including learning, is the result of conditioning. Classical *Pavlovian conditioning* sought the means of achieving a desired response after application of a stimulus. Specifically, the approach was to begin with an existing stimulus-response pair, such as a dog's salivating in response to the stimulus of seeing food, and then to add a simultaneous neutral stimulus, such as the ringing of a bell that initially would not produce the response. After sufficient repetition of the stimulus pair (food-bell) and the ensuing response (salivation), application of the initially neutral response (bell) alone, without the original stimulus (food), stimulated the original response (salivation).

B. F. Skinner, whose views came to dominate psychology in the United States for several decades, introduced an alternative approach known a *operant conditioning.* In operant conditioning, the frequency of a desired result, called an *operant,* is increased if it is followed by positive reinforcement, or alternatively, if undesired results are followed by negative reinforcement. In animal experiments, a rat that pressed a bar (operant behavior) would receive a pellet of food (positive reinforcement) or a pigeon that pecked a dot (operant behavior) would receive some grain (positive reinforcement). In operant conditioning, notice that a stimulus can be absent or, if present, may be unknown or ignored.

From a Skinnerian perspective, traditional instruction emphasized providing stimuli, through content, to the learner. In contrast, *operant behaviorism,* often called simply *behaviorism* because of its ultimate dominance of the behavioral perspective in psychology, shifted the emphasis to reinforcing desired operants, or behavior, of the learner. With this perspective, Skinner developed *programmed instruction,* in which competence is developed in a learner by dividing the learning process into steps sufficiently small to be easily achievable and by providing reinforcement when each small step is accomplished successfully. The size of the steps is chosen to be small so that the learner experiences positive reinforcement as frequently as possible. The small steps in the programmed learning approach also mean that the information can be presented and the learner response can be checked and correct responses reinforced automatically by what Skinner called *teaching machines.* Skinner viewed teaching machines as more effective in providing reinforcement than teachers because they could provide reinforcement more quickly. The concept of teaching machines shifted the focus of applying technology to instruction from presentation alone, as with films, for example, to reinforcement as well.

Empirical results for programmed instruction compared with those from conventional approaches were disappointing. Moreover, many students found the programmed instructional materials boring. As an illustration of the dynamics that typify the interplay between psychological theories of learning and their consequent theories of instruction, however, the programmed instruction movement persisted until the late 1960s, more than a decade after adherence to behaviorism in psychology had waned. Thus, the concepts of programmed instruction and the teaching machine, obviously ready-made for implementation on digital computers, were available to influence significantly early large computer-aided instructional projects, such as PLATO and TICCIT, despite rising dissatisfaction with these concepts among psychologists at that time.

Another behavioral theory of instruction that attracted the attention of some engineering educators during the 1970s was the Keller plan, which focused not on programmed instruction or teaching machines, but on personalized, self-paced, mastery-oriented instruction (5). Although performance on final examinations by students who used the Keller approach typically exceeded that for students in traditional courses and students liked the flexibility of self-paced instruction, critics charged that it inevitably taught students a subservient approach to learning that, in the long run, is quite unfortunate because it inhibited later independent learning. Some studies indicate increased time requirements for learning and higher dropout rates for students as well.

Cognitive Science. In contrast to behaviorism, cognitive psychology, or *cognitive science,* emphasizes understanding the role of consciousness, thinking, and reasoning in behavior (5,32). The consequent development of cognitive models for mental processes considerably expanded the possibility of providing insight useful in understanding and constructing successful learning environments. Jean Piaget, as an early example, developed models of cognitive development that helped teachers understand the cognitive readiness of students for different types of learning, such as dealing with abstractions and hypotheses (5,32).

During a decade beginning in the mid-1950s, the cognitive perspective eclipsed behaviorism in psychology, and cognitive theories of learning therefore, in time, became dominant. In contrast to behaviorism, the cognitive perspective not only emphasizes the study of mental constructs and organization of knowledge, it concentrates on knowing rather than responding and considers people to be active, problem-solving learners rather than passive subjects of conditioning. Although psychologists such as John Dewey (5) and Kurt Lewin (5) had advocated cognitive views earlier, the influences that favored eventual predominance of the cognitive perspective included the translation of most works of Piaget into English by the 1960s, the influence of information theory as developed by Claude E. Shannon and Norbert Wiener, and the advent of the computer and artificial intelligence (5). The architecture of computers, for example, suggested to cognitive scientists that cognitive processes were as real as physiological processes. Such information processing analogies led to early models of memory and of cognitive algorithms for making sense of sensory information.

The cognitive perspective took longer to affect theories of instruction than it did theories of learning, however. Even those who embraced at least some elements of cognitive science constructed theories of instruction that produced learning environments that emphasized learning by conditioning rather than learning by problem solving and exploration. Benjamin S. Bloom, for example, proposed a classic learning taxonomy based on cognitive concepts but developed a mastery learning approach for instruction that was essentially behavioral and hence proved ill-suited for encouraging divergent thinking and creativity (5). Robert M. Gagne also developed a taxonomy starting from cognitive concepts and from it developed an approach for accomplishing learning by achieving carefully predetermined behavioral objectives (5,32). Despite conceptual cognitive influence, the initial emphasis on behavioral objectives resulted in learning environments that suffered the limitations typical of behavioral approaches. David P. Ausubel developed a cognitive theory that foreshadows a number of later developments and, based on the manner in which he believed learners build cognitive structures, indicates that learning environments should begin by introducing learners to general concepts and then proceeding to the specific (5,32).

Over time, the choice of learning strategies has expanded far beyond those available in the initial restrictive templates of TICCIT by relying more and more on cognitive perspectives and relying less and less on pure behaviorism. Developments based on cognitive concepts have proceeded along several different paths, however.

Instructional Design. Much of the particular design procedure known as *instructional design* or *instructional systems design* stems from the work of Gagne, and hence includes cognitive aspects together with a substantial behavioral component (34–37). Over the years, Gagne and others have incorporated more and more cognitive concepts into the instructional design process to achieve additional dimensions in the learning environments developed from this perspective. One detailed account of instructional design, the most widely used single approach to designing computer-aided instruction, is given by M. David Merrill, who originally worked on the TICCIT project (37). Beginning with concepts from Gagne and Ausubel, authors Mengel and Adams more recently have modified the conventional approach to instructional design by adapting and including concepts from software engineering to develop a design methodology specifically for hypertext computer-aided instructional materials (38). Incorporating hypertext (a concept sometimes attributed to Vannevar Bush, an electrical engineer) (39) offers the possibility of increasing the flexibility of learning environments created through instructional design and yet maintaining some of its best features. Ausubel's approach of moving from the general to the specific is compatible with the top-down design approach central to software engineering.

Despite impressive expansion of its scope, critics of instructional design complain that the resulting environments achieve too little of the complexity and diversity that is readily envisioned for interactive learning environments after, say, browsing the WWW. Perhaps the difficulty is that instructional design emphasizes highly controlled and therefore in some sense closed, or at least highly circumscribed, learning environments.

Intelligent Tutors. Developers of *intelligent tutors,* based on applications of artificial intelligence in learning environments, seek to apply cognitive science directly to computer-aided instruction (5,6,16,33,40–42). The tutors typically maintain separate complex internal models of the learner and an expert and include a tutoring component that relies on the learner and expert models in selecting a course of interaction with the learner. In one simple approach, the learner is modeled as knowing a strict subset of what the expert knows. Learning progress is indicated by the size of the subset in comparison to the set of what the expert knows. Such a model obviously cannot take into account the mistaken knowledge that the learner "knows." One means of avoiding this and other problems in constructing a model for learners is to eliminate the learner model and permit the learner to interact with the expert model through a mutual exchange of questions and answers. An early classic example of this approach, the sophisticated instructional environment (SOPHIE), sought to teach troubleshooting of relatively complex electronic circuits as a means of transforming classroom knowledge about electronics into intuitive, experiential knowledge (6,33,42–44).

SOPHIE I and SOPHIE II combined a powerful natural language interface, an inference engine and an early version of the circuit simulator called simulation program with integrated circuit emphasis (SPICE) to realize a learning environment that contributed significantly to the credibility of intelligent tutoring systems. The circuit simulator SPICE functioned as the expert module. To SOPHIE I's capability for addressing natural language questions posed by learners, SOPHIE II added an articulate expert to give demonstrations. With this capability, a learner could insert a fault (replacement of a good component with a faulty component) into the circuit and watch the articulate expert explain its strategy as it tracked down the problem. During this troubleshooting process, the learner was asked to make qualitative predictions (higher, lower, or roughly the same) about the result of each measurement on the circuit into which the fault had been inserted, in comparison with the circuit without the fault, before the expert actually took that step. If the learner made a wrong prediction, SOPHIE took steps to help the learner understand the result of the measurement.

SOPHIE II also included a game in which two learners took turns inserting faults for each other to find. The learner who was looking for the faulty component was assessed a cost for each measurement. That cost increased approximately according to the difficulty of conducting the measurement in a practical setting. The learner who inserted the fault, in contrast, was called upon to predict (higher, lower, or roughly the same) the result of each measurement on the circuit into which the fault had been inserted, in comparison with the circuit without the fault, before the other learner actually made that measurement. The game was complex in that the score of the learner who inserted the fault depended upon the product of the percentage of successful predictions (made by the learner who inserted the fault) about the measurement and the cost of the measurements (made by the learner who must find the fault). If you were the learner who inserted the fault, the winning strategy was to insert faults with the most complicated consequences that you could understand and, of course, increase your understanding of the circuit so that you could insert faults with more complicated consequences to better thwart scoring by the other learner. When teams of two rather than individuals played the game, debates between the partners regarding the next move provided valuable insight into the thinking strategies of the learners that otherwise was difficult to obtain.

SOPHIE III represented an attempt to move away from simulation-based (SPICE) expertise about circuits to a more flexible representation of such expertise. For pedagogical purposes, explaining the basis of a result to a learner can be critically important, but simulations provide little information about the causality on which their inferences lie. SOPHIE III explored replacing the circuit simulator with qualitative, or causal (rule-based), models so that it could better follow the activities of a learner and then provide coaching rather than merely answering specific questions posed by the learner or by giving demonstrations. In the end, the expertise for electronics troubleshooting proved to be difficult to accommodate successfully in a causal model, and workers pursued application of such models to develop coaches in other fields. SHERLOCK, a more recent intelligent tutoring system for electronic troubleshooting, employs software object techniques for managing complexity in the expert component of the tutor (45).

Although SOPHIE II was used in an actual short course, application of SOPHIE and subsequent intelligent tutoring systems for instruction has been limited. While some educators question whether intelligent tutors, by their very nature, can provide the learner-centered environments that promise greater effectiveness in learning, the use of intelligent tutors in actual learning environments appears to have been limited mainly by the immense effort required to write the necessary software. The rare deployment of intelligent tutors in actual learning environments also reflects the circumstance that studies of intelligent tutors are directed as much towards improved understanding of certain aspects of cognitive science as towards successful instruction.

Learning Styles. Beginning with the work of Piaget, Dewey, and Lewin, David A. Kolb developed the concept that different learners prefer different learning styles in building the mental or cognitive constructs associated, in the cognitive perspective, with learning (46,47). For example, some students find it easier to relate to facts and data while others prefer to deal with theories and mathematics. Some find it easier to deal with information visually, in pictures or diagrams, for instance, although others learn more easily from written or spoken information. Some students prefer to learn by interacting with other students. Some prefer to learn alone. Kolb holds that learning improves when learners pursue all styles of learning, not merely their individually preferred styles. Some engineering educators conclude that, in practice, professionals must learn in all of the styles (48). From either perspective, it follows that learning environments should help students develop learning skills in the learning styles they do not prefer as well as in those that they do. This presumption leads to the concept of *teaching around the cycle* to help learners experience learning in the various styles. Computer-aided instruction seems to offer the promise of applying, and then evaluating, the Kolb approach by structuring learning environments that accommodate various learning styles and by measuring and studying the response of learners to the approach of teaching around the cycle. In principle, the learning environments even could be changed adaptively, depending upon the performance of the learner in the environments corresponding to the different learning styles. That promise has not yet been realized in practice, however.

Biological Bases of Learning. Just as knowledge about computers stimulated developments in cognitive science, so too, has increasing knowledge about the biology of the human brain. Neurophysiology, while still maturing, identifies several characteristics of the brain that relate to cognition (32). Modularity and plasticity, for example, may have important consequences for learning. *Modularity* means that different parts of the brain correspond to memory for different cognitive functions. The left and right sides of the brain, for instance, seemingly correspond to different functions, and certain functions seem correlated with even more specific regions. The different regions, or modules, seem to function with some degree of autonomy and appear to be stimulated, or accessed, by different senses (vision, hearing, and so forth). The ease with which the various modules can be accessed seems to vary significantly from individual to individual.

A consequence of modularity for learning and instruction is that different individuals have preferred modes of processing information, depending upon which brain modules are most easily accessible to them through corresponding sensory stimulation. Some may prefer to learn through listening, for example, and some through seeing. Thus, a learning environment that involves several sensory stimuli probably helps learners access more of the brain's different modules and, presumably, enhances learning through construction of more complex cognitive structures.

Plasticity means that brain structure continues to develop even after birth, rapidly at first, but continuing throughout life. Continued development seems to mean increased capacity for learning. Moreover, development appears spurred by complex interactive environments. Thus, plasticity opens the possibility that appropriate learning environments can stimulate increased capacity for learning, even in adults.

Modularity and plasticity of the human brain, taken together, may imply increased effectiveness of complex interactive learning environments that encompass manifold stimulation of the learner. Elaborate theories of instruction ostensibly based on current knowledge of brain biology, however, are premature and not supported by experimental understanding of the brain (49). Specifically, classification of learners as either right-brain (analytic-verbal) or left-brain (holistic-spatial) is simplistic, although learning environments designed to stimulate and exercise both sides of the brain may be useful merely because of the inevitable complexity they involve.

Neurophysiology holds great promise for ultimately clarifying and providing bases for theories of learning and instruction. For now, however, the links between it and theories of learning and instruction are tenuous.

Constructivism. Although instructional design provides authors with fairly specific design procedures for computer-aided instruction, the resulting lessons tend to lack the flexibility and cognitive complexity that are sometimes realized in lessons composed with less systematic approaches. Adherents of *constructivism,* a cognitive perspective that traces its origins back to Piaget, Dewey, and Vygotsky (32), criticize the instructional design process and even other cognitive approaches as being fundamentally flawed by their basis in *objectivism,* a traditional philosophical perspective that ultimate reality exists, independent of any observer. In objectivism, reality is absolute. *Constructivism,* in contrast, considers ultimate reality to be the cognitive interpretations or mental constructs that each individual builds as a consequence of perceptions and learning. In constructivism, therefore, ultimate reality can be different for different individuals (16,32,50).

For constructivists, learning becomes the construction of individual interpretations of stimuli experienced by learners. Teaching becomes the creation of environments that help learners construct their individual interpretations of what they perceive rather than transmission of information to the learners and reinforcement of what the teacher views as appropriate responses. Because different learners construct different interpretations, learning necessarily becomes learner centered. Indeed, because each learner must construct an individual interpretation, learning is impossible unless the learners take direct action to construct the interpretations. Constructivists often speak of *student-centered learning* and *active learning.* When teaching is viewed as the transmission of information and reinforcement of responses that the teacher considers appropriate, learning is much more teacher-centered and the learner assumes a less active role. Constructivists call such an approach *passive learning.*

Constructivists emphasize *situated,* or *authentic, learning environments* (learning based on the real, or at least realistic, settings in which the learner will apply it) because abstractions or simplified models of reality may distort or otherwise impede the development of the individual mental constructs that are the ends of learning. Because realistic settings are complex, constructivists stress *learning from multiple perspectives* to promote more complete understanding. Encountering abstract concepts in several different contexts, for example, can help learners advance their understanding. Constructivists advocate *collaborative learning* both to assist the learner in achieving multiple perspectives through interactions with others and, perhaps more fundamentally, to accomplish a kind of social validation of the learner's perspective that is necessary in the absence of an objective reality that objectivists rely on as a comparative standard for learners. Because situated learning should occur in a realistic environment, constructivists assert that evaluation and testing should be an integral part of the learning process, not a separate activity. Such *integrated evaluation and testing* might be accomplished through projects or portfolios, for example.

From the constructivist point of view, numerous shortcomings of traditional instructional design become apparent. Breaking up the material to be learned into incremental steps small enough that learners can complete them successfully with little probability of error requires methodical dissection of the material and careful reassembly into logical paths of learning. By design, learner involvement mainly requires the persistence to follow a learning path developed by the teacher. Thus, the environments that result from traditional instructional design are more teacher-centered rather than learner-centered. As a consequence of their limited involvement, learners in such environments fail to gain experience in developing their own approaches to learning unfamiliar complex material. In short, environments that result from traditional instructional design fail to help the learner learn how to learn independently, a central goal of learning, certainly at the university level. Moreover, environments produced with traditional instructional design tend to avoid the complexity of situated learning environments and use simplified, often prematurely abstract, models of what is to be learned to make tractable the dissection of the material into small steps and its reconstruction into paths of learning. Being based on simplified models of reality, what the learner learns may be distorted significantly by convenient but ultimately misleading simplifications. Reassembling small learning steps into multiple learning paths requires considerable imagination, and perhaps even more time and effort, by the teacher. Thus, learning environments produced with traditional instructional design often offer the learner few alternative paths through, and hence perspectives of, the material to be learned. Because the learning steps are so small and the paths of learning are so well-defined in learning environments that result from traditional instructional design, little collaboration among learners is needed, nor indeed is possible, during use of the environment. Learners in these environments therefore miss out on possible enrichment of their learning by the ideas of other learners. Integrated evaluation and testing of a very narrow kind is certainly a part of learning environments created with traditional instructional design, but implementation of the concept as advocated by constructivists usually is impossible because the learning is not situated in sufficiently realistic environments.

Constructivism certainly suggests several routes of escape from the somewhat delimited domain of traditional instructional design. Constructivism itself, however, is open to criticism on several points, not the least of which is its sacrifice of the objectivism that lies at the heart of modern engineering and science to accommodate a diversity of perspectives constrained only by social negotiation. A socially negotiated approach to electromagnetics that is inconsistent with Maxwell's equations ultimately would prove counterproductive, perhaps at considerable cost to the learners. Situated learning environments can degenerate into a mimetic apprenticeship approach that neglects development of abstract concepts necessary for applying learning in alternative topical domains and yet consumes and distracts the attention of the teacher. The complexity of realistic situated learning environments can hamstring beginning learners. Practical collaborative learning environments can allow uneven participation by learners, and hence uneven learning, to a degree that makes evaluation and testing difficult and the results fruitless. Integrating evaluation and testing into situated learning environments can make focusing assessment on learners' understanding of concepts that are implicit in the task difficult. For example, a teacher who assesses a fuzzy-controlled robot that successfully balances a vertical rod by appropriate compensating movements may find it difficult to decide if the learners who collaborated on the design and implementation of the robot understood important concepts in fuzzy control or merely implemented a ready-made algorithm that they found. Supplemental means of assessment are necessary.

Review of the criticism by constructivists of learning environments created with traditional instructional design reveals that many of their complaints are directed at behaviorism rather than at objectivism. As a result, the insights provided by constructivists are being adopted by both constructivists and cognitive objectivists to enrich the approaches in contemporary instructional design (36,51). Modification and replacement of conventional instructional design by constructivist concepts began in the mid-1990s (52–54).

Although computer-aided learning environments developed specifically from a constructivist perspective for use by engineering students are not yet common, existing environments exhibit some of the attributes that constructivists advocate. ELECSIM, already mentioned, illustrates one approach to structuring a computer learning environment that includes features advocated by constructivists. In the basic approach, a learner encounters a collection of connected simulated "rooms." Each room concerns a topic with scope roughly comparable to a subsection in a conventional textbook. Learners basically are free to interact with the material in each room and indeed the entire collection of rooms, in whatever sequence they choose. If teachers wish, however, they can limit possible destination rooms accessible from each room and can insist that learners successfully complete a quiz or other work before leaving one room for another.

A collection of rooms is called a simulation implementation of multifaceted peripatetic learning environments (SIMPLE). A SIMPLE room contains, at the pleasure of the designer, informative notes and explanations, drill and practice exercises, project assignments, (software) tools, and network access. While touring the complex of rooms, the learner can develop understanding, accomplish and document tasks, learn new tools, and demonstrate competence.

A major difficulty with implementing any computer learning environment is, of course, software development. The strategy in constructing SIMPLEs is to rely primarily on existing commercial software that is widely used and thus authentic to some degree. ELECSIM, a SIMPLE that concerns a course in analog electronics for undergraduate electrical engineers, can include readily available software components such as circuit simulators (PSpice), math packages (Mathcad, Matlab, Maple, Mathematica, and Macsyma, for example), and logic simulators, as well as WWW browsers, programming languages, word processors, spreadsheets, and databases. Custom software and multimedia also are readily accommodated. The ELECSIM software serves mainly as "glue" to join disparate existing software. The rooms can be easily replicated, edited, modified, and combined to form other rooms or SIMPLEs on a particular subject, whether or not they relate to analog electronics. The rooms thus are a kind of structural template into which existing materials, such as notes and examples of almost any sort, can be integrated, without undue effort, to form flexible interactive learning environments.

The rooms in ELECSIM rely on strong visual images for two purposes. First, the learner develops a strong mental picture of where everything in the environment resides. The central screen in an ELECSIM room displays a graphic view of an office and helps learners recall the location of the various available resources. For example, it is easy for a learner to remember to click the mouse on the file drawer to find supplemental notes about the subject of this room, click on the index file on the desk to find interactive drill and practice exercise problems and open-ended homework problems, click on the computer to find software tools needed to do work in the room, click on the pile of mail to send work to the teacher, click on one of the shelved books to find references or links to related material, and click on the road in a picture on the wall to leave the room. The strong graphical context therefore permits the learner to access directly much greater functionality, without maneuvering through several levels of screens or menus, than would be convenient with a more conventional interface.

The second purpose served by the visual environment is to give each screen in the room a very definite mental context. When the learner is concentrating on a particular screen, the teacher can know with some certainty what the learner is thinking about and doing at the moment. The sequence of screens chosen by the learner therefore tells much about the learning strategy and process that the learner is using. The teacher can choose to record automatically the learner's activity to any level of detail. All of this information can be available to the teacher on-demand over a network or from a diskette for processing and interpretation.

As far as constructivism in ELECSIM is concerned, learners control their learning paths and utilization of the various elements in each room and construct individual solutions to complex open-ended problems. The environment is thus manifestly *learner-centered* and requires *active learning.* ELECSIM provides a *situated,* or *authentic, learning environment* by utilizing the tools of practice, rather than tools built especially for the learning environment. ELECSIM stimulates *learning from multiple perspectives* by providing ready access in the room to notes and other resources, as well as requiring the learner to move beyond drill and practice exercises to synthesis of concepts through design. Video, other multimedia, and hypertext could provide perspectives beyond those possible with text and simple graphics. Because ELECSIM is essentially a consistent user interface to available resources, e-mail and chat rooms can be incorporated easily, if the computer is networked, to provide a framework for *collaborative learning.* Learners, for instance, could collaborate on a design without being coincident in space or time. Submission of documents that record and describe the learner's activities, on open-ended homework problems, for example, provide *integrated evaluation and testing* in ELECSIM.

The SIMPLE paradigm, illustrated in ELECSIM, provides a consistent graphical user interface to tools and other resources related to particular topics and collects them in "rooms" that can be visited at the learner's convenience and provides, as well, one metaphor for creating flexible learning environments that can incorporate the elements advocated by a variety of cognitive and constructivist theories of instruction. A key to the flexibility is incorporation of existing software. From a different perspective, the scope of the rooms in the SIMPLE approach is much larger than the small Skinnerian steps of instructional design, and the learner in a SIMPLE environment has much more choice of learning paths. On the other hand, a teacher who develops a SIMPLE dissects the material and exerts control over the learner more than most constructivists deem appropriate. In this sense, SIMPLE represents an intermediate approach to that of extreme behaviorism and constructivism.

Sun and Chou describe the role of constructivist ideas in the design of the cooperative remotely accessible learning (CORAL) system, a multiyear effort in distance education funded for development by the National Science Council in Taiwan (55). The specific constructivist ideas considered during the design of CORAL were to (1) make courseware learning materials as rich as possible, (2) give students authentic tasks on which to practice (building a local area network system in their computer laboratory, for example), (3) encourage students to provide different solutions to given problems (brainstorming innovative methods for preventing computer viruses, for example), (4) encourage students to navigate through instructional nodes and construct their own learning paths, (5) encourage students to interpret new learning situations based on their existing knowledge and experiences, and (6) encourage students to discuss, debate, and work cooperatively. Access to the hypermedia CORAL courseware is possible with a special browser that permits video conferences, exchanges of text and graphical information via an electronic whiteboard, and discussions, all as means of promoting communication and interaction between teachers and learners. Students access CORAL with standard personal computers, although video conferencing requires a high-speed network card and extra equipment such as a camera, microphone, and so forth. The CORAL system records details of learner activities that permit construction of student models. Such records open the possibility of rewarding students for helping each other. Ultimately, CORAL will include a student tutor system (56).

Collins advocates *cognitive apprenticeship* specifically as an approach for developing cognitively rich computer-aided instruction that helps learners develop the mental constructs that are the essence of learning from a constructivist perspective (7). This approach, based on a generalization of the traditional apprenticeship concept, identifies six components that leverage and exploit available computing technology in implementing learning environments.

Situated learning involves reliance on actual, or at least realistic, representations of the environments in which what the learner learns will be applied. For example, a circuit simulator used in industry, such as PSpice, could form the basis of a situated learning environment, as could a C++ software development environment. Beyond the domain of software, computers can help create situated learning environments with multimedia to bring realism to the desktop.

Modeling represents a complex process in simpler terms for the purposes of explanation. In addition to utilization of mathematical models and simulators, computers can deploy animation and other forms of multimedia to help learners grasp complex material. Computers can even help learners understand how to solve problems by showing (modeling) how experts solve problems. The articulate expert in SOPHIE, mentioned earlier, is an example of modeling.

Coaching provides personalized hints or assistance to learners as they need them. Relatively simple rule-based branching programs can provide coaching in limited contexts, such as within a single example, without the difficulties associated with authoring and implementing a more ambitious traditional intelligent tutoring system.

Reflection requires learners to reconsider their activities and, in so doing, evaluate their own performance. Comparison of the simulated performance of a learner's design with performance requirements can stimulate reflection. Submission to the teacher of word processing files, with attachments, that document and explain a learner's work can stimulate and document reflection, as can certain types of examinations.

Articulation requires learners to describe and explain their learning activities as a means of making their tacit knowledge explicit. In addition to word processing files with attachments mentioned earlier, chat rooms and e-mail via networks permit learners to interact with others and thereby practice articulation and, in addition, gain new perspectives from their peers.

Exploration requires learners to try out different hypotheses, methods, and strategies to see their effects. Because exploration puts the learners in control, they must learn how to explore productively. Computers offer the advantage of permitting rapid examination, in limited time, of wide-ranging alternatives, through simulators, for example.

As far as cognitive apprenticeship is concerned, ELECSIM provides *situated learning* through the use of tools, such as the circuit simulator PSpice and other software, that are widely used for electronic design and communication in industry. As discussed earlier, PSpice permits learners to carry out realistic iterative solutions to open-ended design problems of the type encountered in practice and, thereby, develop the important ability to assess their own work, with only modest direct involvement of the teacher. As appropriate, additional realism can be incorporated into the environment with video clips or other multimedia.

In the stand-alone version of ELECSIM, *modeling* is provided to some degree by drill and practice exercises in which a learner views a circuit and must enter component values that cause the circuit's operation to meet certain design rules. The exercise thus presents a simplified, or modeled, view of the circuit's operation and thereby permits the learner temporarily to focus entirely on the design rules. The answers entered by the learner are checked to determine if they satisfy appropriate design rules, as explained earlier. Violation of the rules produces informative messages to the learner, as described later. The entries into the exercises are not checked with the circuit simulator PSpice, which does not operate on the basis of the design rules. Instead, branching comparisons are used to check whether or not the answers satisfy the design rules. In the open-ended homework problems, learners use the circuit simulator (a different model) to evaluate the performance of their design with respect to the given performance specifications. Neither audio nor video were included in the learning environment for analog electronics, although they certainly could be central to SIMPLEs on other topics and incorporating them is straightforward. In the stand-alone version of ELECSIM, modeling in the sense of showing how an expert solves problems is present only in the explanatory notes available in the rooms. In a networked environment, e-mail, chat rooms, or conferencing could show the teacher solving problems in action, complete with pursuits down blind alleys and mistakes. Over time, selected transcripts of these interchanges could be posted as notes in appropriate rooms as one means of preserving the teacher's initial approach to solving an unfamiliar problem, an approach that likely will be lost in an inevitable polishing process, otherwise.

The implementation of *coaching* in ELECSIM is kept simple by limiting the context to which coaching applies to the particular context to which the learner is directing attention at the moment. Specifically, consider as an example a drill and practice exercise from an ELECSIM room that deals with the design of simple bipolar junction transistor (BJT) audio voltage amplifiers. In a standard four-resistor bias configuration for a typical BJT, the learners are required to specify values (not unique) of the resistors, the power supply voltage, and the coupling capacitors that will bias the BJT at a specified operating point and achieve a specified open-circuit voltage gain over the audio range (57). A simple branching structure checks learners' entries against design rules explained in notes available in the room in which the exercise appears and provides either a message that the design is acceptable, in that it satisfies those rules, or if it does not, provides on-screen messages that indicate which components have inappropriate values and whether the values entered by the learner seem too high or too low. Coaching in ELECSIM is thus fairly specific but easy to implement because the rules for coaching apply only in a limited context.

Although even drill and practice exercises such as the one just described can stimulate *reflection* in learners to some degree, solution of open-ended homework problems that require iterative solution absolutely demand reflection. In the room that deals with the design of simple BJT audio voltage amplifiers, learners are required to design a BJT common emitter audio amplifier that gives a certain voltage gain to a specified load, subject to the constraints of a given Thevenin impedance for the driving circuit and a given power supply voltage. This problem involves enough variables and constraints to require an iterative approach to the design. No simple set of equations employed step-by-step suffices. The learner employs approximate analytical results and design rules for an initial design and then evaluates the design by simulating its performance with the circuit simulator and comparing the simulation results with the performance specifications. If the performance specifications are not met, the learner can then refer back to the approximate analytical results and reflect on how the design might be changed to bring its performance within the specified ranges. This iterative reflective process may be repeated several times before the learner achieves a successful design. In the process, the learner practices, as mentioned earlier, self-evaluation and, through reflection, experiences prompt feedback about relatively complex open-ended problems without direct involvement of the teacher.

Articulation in ELECSIM is required when learners prepare and submit documents, using a standard word processor, that describes their learning activities. For the open-ended audio amplifier design problem described above, the teacher might require the learner to submit a document file that includes (perhaps as software attachments) the PSpice schematic capture file to show the circuit of the design, design calculations, samples of the simulation output (values, graphs and so on) for the circuit, and perhaps most important a discussion about how the simulation results demonstrate that the design satisfies performance specifications, or if it does not, a discussion of why the performance specifications could not be achieved. Such a document provides considerable insight to the progress of a learner and provides the teacher a convenient means of investigating the learner's work in more detail. For example, a double-click on the imbedded PSpice schematic capture file by the teacher or an assistant not only displays the schematic file, but makes possible an immediate simulation of the schematic displayed and subsequent investigation of the output from the learner's design.

Exploration of the various rooms and their topical contents is the essence of ELECSIM, which can be viewed as one instance of a consistent graphical user interface to a diverse collection of tools, notes, problems, media, and network resources assembled by the teacher to assist the learner. From a different perspective, solutions to the open-ended problems in ELECSIM require learners to formulate hypotheses and test the consequences.

Jonassen (58) discusses how teachers can implement cognitive apprenticeship on stand-alone desktop computers, without specialized computer software, but with widely available and inexpensive software such as databases, spreadsheets, semantic networks, and expert systems. In addition to self-assessment documentation, summary statistics about performance, and portfolios for assessing learning outcomes, his suggestions include learning logs, student rankings of course objectives, think-aloud protocols, documented problem-set solutions, brief autobiographical essays on a specific learning experience, cognitive interviews, directed paraphrasing, analytical memos, classification/decision matrices, diaries and journals, experiments, concept maps, and debates. Specific applications of spreadsheets in engineering that can employ elements of cognitive apprenticeship include simulation of computational and sequential logic circuits and the solution of ordinary and partial differential equations, as well as easy evaluation of complicated equations and generation of graphs that display the results for various parameter values as part of, for example, an iterative process (59–61).

Evaluation of Learning Environments

Development of evaluation criteria for learning environments, electronic or not, is complicated by the absence of consensus about an underlying theoretical framework for theories of learning and instruction, as well as by several different expectations for the evaluation process (62). A particular difficulty is that approaches to evaluation that adopt, explicitly or implicitly, the viewpoint of a particular theory of learning or instruction can give negative results for a project developed from a different theoretical perspective. That is, the result of the review may follow more from the nature of the learning environment than from the degree of success it achieves according to the theoretical perspective with which it was developed and implemented. At first glance, focusing evaluation directly on the learning accomplished by participant would seem to circumscribe this problem. Alas, deciding what should have been learned depends directly on the theoretical perspectives chosen. The matter is usually not as simple as deciding whether the focus of learning should have been facts or process, but that dichotomy illustrates the point. Moreover, application of evaluation approaches developed for a different context are not always easily adapted to the evaluation of materials for computer-aided instruction, especially those for engineering education.

Evaluation, therefore, demands careful planning and work. In practice, it is not easy. To the extent that evaluation is neglected, however, the iterative approach to design that is so traditional in engineering endeavors is not possible in the development of interactive learning environments for engineering education.

The National Engineering Education Delivery System (NEEDS), mentioned earlier, developed criteria for the review and classification of software it receives for posting on its WWW site, as reported by Eibeck (63), which can serve as a useful guide. The evaluation and classification criteria used at first span nine categories. *Engineering content* deals mainly with whether or not the material is free of errors and corresponds to the level of the intended users. *Engagement* is an assessment of the appeal of the courseware to the intended users. *Impact on learning* indicates whether or not different learning styles are accommodated and whether or not feedback is provided to the learner. *User interface* evaluates consistency, clarity, and ease of use as well as the effectiveness of help features available to the learner. *User interaction* assesses to what extent the software involves the learner and whether the involvement is active or passive. *Multimedia design* considers the quality of the multimedia and whether the media effectively support the learning process or merely distract the learner, instead. *Instructional use* concerns how easily the software can be incorporated into a course by a teacher other than an author. *Performance* appraises how well the software runs on the specified computer platform. *Accessibility from NEEDS* deals with operational issues of finding the files in the NEEDS database and downloading them. Although it would be difficult to argue that these categories can be neglected by authors of useful software, the review process based on them proved unwieldy, in practice, to reviewers. NEEDS therefore simplified the review process to focus on ensuring that the content is error free, that the package is complete and includes descriptions and recommendations for use, that the software is appealing to users, and that it is potentially useful to teachers other than the author.

Projects for developing computer-aided instruction, as well as complete courses designed with computer-aided instructional components, require a broader approach to evaluation than considering existing computer-aided instructional material for possible use. Fortunately, some widely accepted approaches to project evaluation (62) have been adapted for projects in engineering and to the development of computer-aided instruction projects, as well (16,33,64). Project evaluation consists of three basic stages: (1) planning evaluation, (2) formative evaluation, and (3) summative evaluation. During development of a project, planning evaluation, although often neglected, helps focus attention on goals of the project as well as on strategies and schedules for achieving them. While the project is underway, formative evaluation identifies opportunities to improve the project. After the project is completed, summative evaluation assesses the success of the project. Forsyth, Jolliffe, and Stevens discuss application of a multilevel evaluation model that permits concentration on (1) the learner's feelings and about the course, (2) learning achievement during the course, (3) behavioral changes in the learners during the course, or (4) overall impact of an innovation in an organization (65). Worthen, Sanders, and Fitzpatrick describe and contrast several alternative approaches to evaluation that can be adapted for application to computer-aided instruction (62).

Careful project evaluation is important in achieving credibility necessary for widespread use of the results of any project. It is almost essential, however, for projects in an emerging area such as computer-aided instruction in engineering where the lack of familiarity and confidence among potential users may breed skepticism that prevents widespread interest in the results of the project.

Pragmatic Development of Computer-Aided Learning Environments

As the power and sophistication of hardware and software available for computer-aided instruction continue to increase, contemporary answers to the fundamental question "How can we use computers to improve education effectively and inexpensively?" amount to discoveries of strategies for developing environments in which learners can proceed effectively.

Strategies for Conceptual Design. The likelihood that teachers of engineering can discover carefully and coherently designed ready-to-use courses of study that match the needs of their students always has been low. At best, teachers can hope to find elements and components that they can incorporate into their own designs. Incorporating computer-aided instruction into the design of courses of study changes that picture but little. An analogy with engineering textbooks suggests that some teachers of engineering will play a dominant role in developing computer-aided instructional components just as they do in authoring engineering textbooks. The market for engineering instructional materials is just too small to attract full-time authors with the appropriate expertise. Clearly, the teachers of engineering who develop components for widespread use must understand principles of course design very well. If their students are to realize substantial benefits from the materials available, however, even teachers of engineering who mainly integrate computer-aided instructional elements produced by others into learning environments for their own students must understand and apply the principles of course design as well. If they do not, the risk of producing a flood of poor-quality courses that can damage the success of computer-aided instruction in engineering for a long time is great. The stakes are high.

From one perspective, finding a suitable strategy for teaching a particular topic or designing a course seems confusing and, even worse, unlikely. Despite substantial developments in theories of learning and instruction, no consistent approach to designing learning environments (computer-aided or not) is widely accepted. Candidates for an overall theory of design suffer from (1) poor understanding of their domain of applicability and (2) scarcity of empirical verification. Perhaps with the availability of powerful, inexpensive computer systems and networks, the development and verification of design approaches based on theories of learning and instruction eventually will be forthcoming, but not now.

That pessimistic view comes easily to engineers who are accustomed to working with experimentally verified theories, such as those of electromagnetics, thermodynamics or signal processing, whose power, marvelous accuracy, and domain of applicability are well understood and documented. But engineers have long worked successfully where theories are far less robust. Management is just one example. A manager responsible for some particular effort finds no powerful universally sanctioned theoretical approach to managing the activities of a particular project. Indeed, a manager easily can become bewildered by the multiplicity of diverse and inconsistent approaches advocated by hosts of management theorists. And yet, individuals find ways to manage complex projects successfully. How? The situation is not quite as complicated as first it seems. Certain fundamental principles of management have become widely accepted and understood (66). These principles, alone, do not give very specific guidance to a manager, but they are ignored at great peril. More detailed guidance is provided by more specialized theories, such as Theory Z or total quality management (TQM), whose domain of applicability usually is not clear. The manager must develop a specific management approach for a particular effort based on management fundamentals and on insights provided by the specialized theories. That the specialized theories seem to come and go complicates matters, of course.

A designer of computer learning environments faces a similar situation. Certain fundamental ideas about learning are becoming accepted and understood by people with diverse perspectives. A teacher must combine a knowledge of these principles with insights from more specialized theories that seem to fit the situation, topics, and learners at hand to develop successful learning environments.

What are some points (67) of consensus that seem to be emerging? First, concepts are best learned when students encounter them in a variety of contexts rather than from a single perspective. Even if a learner retains a concept experienced from a single perspective, that concept is likely to be isolated and unavailable for linking with others to build related or more complex concepts. Applying the concept in a different context is an important means of understanding it. Second, realistic experiences are extremely effective in helping learners learn, especially in grasping abstractions. An abstraction not linked to several real situations is unlikely to be accessible for building understanding of diverse contexts in which it might apply. Third, learners learn effectively when they take action and then something happens in turn from which they can learn. Giving and receiving feedback in a peer group is one example. Physical experiments are another. Interaction with simulations offers a third possibility. In short, an emerging consensus is that learning should be active and experiential.

Tools for Implementation. Although careful conceptual design of a computer-aided learning environment is essential, the realization of the environment in practical and robust software may require more effort and produce greater frustration. The PLATO system included TUTOR, perhaps the first widely used tool designed specifically for helping teachers to become authors of interactive learning environments (3). CyberProf provides a set of tools for the same purpose (25). More generally, almost any programming language can serve, in principle, as a tool for constructing interactive learning environments. In practice, the complexities of building an acceptable graphical user interface alone require powerful software tools if mere mortals are to succeed. Fortunately, the business market has stimulated the development of numerous powerful, easy-to-use, and relatively inexpensive tools for potential authors of computer-aided instruction. Unfortunately, the large number of authoring programs available and the variety of features included (and omitted) can make the choice bewildering.

Helpful perspective is provided by Schwier and Misanchuck, who describe a number of features that should be considered in selecting an authoring program appropriate for construction of multimedia learning environments (17). *Portability* determines what fraction of desktop machines can use the learning environment constructed. Although the WWW has simplified the problem of portability to some extent, the basic hypertext markup language (HTML) environment on the WWW is not as rich as many platform-specific environments. Two trends are ameliorating this problem. First, HTML itself is being continually upgraded to provide a richer environment. Second, WWW browsers now accommodate special file-type-specific plug-ins that permit browsers to display, or play, files of almost any type, provided only that an appropriate plug-in is available and it has been installed in the browser on a particular machine. Just as browsers are platform specific (Windows, Macintosh, UNIX), so too are plug-ins, and it might seem that plug-ins accomplish little. The number of computers on the WWW is so large, however, that competitive pressures have led many vendors of software, including vendors of authoring programs, to make available, free of charge and for all major platforms, plug-ins that accommodate file types special to their products. With this approach, multimedia environments can be created with an authoring program that runs only on a single platform, although the resulting files can be displayed by browsers on any platform. Which file types ultimately may prove popular enough that capability for handling them is built into browsers and which file types will continue to be supported by special plug-ins is decided by complex market processes the outcome of which is difficult to foresee. Nevertheless, the WWW clearly is becoming an increasingly rich environment for computer-aided instruction.

Licensing agreements must be purchased before software produced with some authoring programs can be distributed. WYSIWYG (what you see is what you get) is an almost essential, but not universal, feature for contemporary authoring programs. Not being able to see an environment that you are designing without interrupting the design process specifically to display the environment wastes time and precludes convenience. *Flexibility* means that the author, not the authoring tool, should determine the kind of learning environment to be constructed, although *advanced author support* that provides suggestions to help the author maintain instructional integrity according to some particular design paradigm may be beneficial. The ideal authoring tool should accommodate *animation, video,* and *audio,* as well as *text* and *graphics. User control* concerns the degree to which the authoring program supports, for example, a keyboard, mouse, graphics tablet, touch screen, light pen, speech recognition interface, barcode reader, or virtual reality interface. The authoring program also should be extensible through *programming features* that give ready access to a high-level programming language that, among other things, accommodates bridges to external software and permits authors to add custom features to the authoring program. *Performance tracking* includes features such as answer judging and activity reporting. *Networkability* is a measure of how well the authoring program itself, and the software it produces, works on networks. If the learning environment constructed is for deployment on the WWW, for example, the availability of suitable plug-ins is an important consideration.

Overall, the desirable features of an authoring program amount to the requirement mentioned earlier for a successful graphical user interface: transparency. An author should be able to concentrate on designing the learning environment without distraction or frustration by the authoring program.

SIMPLE, a Windows authoring program available on an archived CD-ROM and used to create ELECSIM, is especially designed for constructing interactive learning environments (13). It provides a WYSIWYG authoring environment and straightforward extensibility through Visual Basic and incorporation of external software, includes performance-tracking features and configuration management tools for network deployment of the learning environments, incorporates multimedia and simple animation, and carries no license fee for educational use.

Although learning environments for the WWW offer unprecedented portability, the available authoring tools (apart from those for CyberProf) at first offered little more than capability for constructing graphical user interfaces for the learning environments. Platform-specific authoring environments were unrivaled in power and flexibility. Emergence of the JAVA programming language, however, promises development of authoring environments for the WWW that provide, in addition to the boon of portability, the power and flexibility previously available only with platform-specific tools. JAVA, an updated and improved version of the powerful object-oriented C++ programming language, is designed specifically to achieve (1) seamless incorporation of the WWW into software and (2) cross-platform portability far greater than provided by C++. JAVA programs, like HTML documents, require only a machine with a suitable WWW browser for use. Like HTML documents, JAVA programs (or applets), can be written to function perfectly well even on non-networked machines, although hyperlinks to sources on the WWW and calls to network servers are not possible, of course. Thus, as the huge WWW market stimulates the development of powerful and sophisticated HTML authoring programs that embody JAVA, it seems likely that the WWW (or a WWW-like environment on non-networked machines) will become the environment of choice for most computer-aided instruction, even that intended for use mainly on non-networked machines. Early applications of JAVA in interactive learning environments began to appear in 1997 (68). Scripting languages such as JavaScript, supported by most WWW browsers, offer the possibility of including and executing simple program procedures in HTML documents, although they provide far less capability than JAVA (13).

Current Status. Engineering teachers only have begun the difficult task of sorting through a multiplicity of conflicting approaches to find useful ways of relying on computer-aided instruction (69–71). The current best practice in assessing the fundamental question of computer-aided instruction, “How can we use computers to improve education effectively and inexpensively?”, for a particular learning situation is to apply the familiar approach of engineering design to the development of interactive learning environments, together with tactics suggested by available theories of learning and instruction, principles of interface design, and intuition as guides. This approach avoids the limitations inevitable in commitment to a single approach and gives creativity reign within constantly changing constraints. Only time will tell how satisfactory the best answers to the fundamental question prove in practice.

BIBLIOGRAPHY

1. W. L. Everitt, Engineering education—circa 2012 A.D. *Proc. IRE,* **50**: 571–572, 1962.
2. R. F. Mager (ed.), Special issue on automated teaching, *IRE Trans. Educ.,* **E-4**: 1961.
3. D. L. Bitzer, The wide world of computer-based education. In M. Rubinoff and M. Yovits (eds.), *Advances in Computers,* New York: Academic Press, 1976, Vol. 15.
4. D. L. Bitzer, The million terminal system of 1985. In R. J. Seidel and M. L. Rubin (eds.), *Computers and Communications: Implications for Education,* New York: Academic Press, 1977.
5. P. Saettler, *The Evolution of American Educational Technology,* Englewood, CO: Libraries Unlimited, 1990, Chaps. 1, 3, 10–12, and 14–16.
6. T. O’Shea and J. Self, *Learning and Teaching with Computers,* Englewood Cliffs, NJ: Prentice-Hall, 1983, Chaps. 3, 4, 6, and 7.
7. A. Collins, Cognitive apprenticeship and instructional technology. In L. Idol and B. F. Jones (eds.), *Educational Values and Cognitive Instruction: Implications for Reform,* Hillsdale, NJ: Lawrence Erlbaum Assoc., 1991, pp. 121–138.
8. D. R. Woolley, PLATO: the emergence of on-line community, *Comput.-Mediated Commun. Mag.,* **1** (3): 5, 1994.
9. J. S. Daniel, Why universities need technology strategies, *Change,* **29** (4): 11–17, 1997.
10. J. S. Daniel, *Mega-Universities and Knowledge Media: Technology Strategies for Higher Education,* London: Kogan Page, 1996.
11. D. H. Jonassen (ed.), *Instructional Designs for Microcomputer Courseware,* Hillsdale, NJ: Lawrence Erlbaum Assoc., 1988.
12. B. Oakley II, A virtual classroom approach to teaching circuit analysis, *IEEE Trans. Educ.,* **39**: 287–297, 1996. See especially material on the accompanying CD-ROM in file \ieeezips\002\cd\tutorial.htm.
13. W. M. Marcy and M. O. Hagler, Implementation issues in SIMPLE learning environments, *IEEE Trans. Educ.,* **39**: 423–429, 1996. See especially material on the accompanying CD-ROM in file \ieeezips\033\cd\elecsim\qselecsm.htm.
14. P. J. Mosterman et al., Design and implementation of an electronics laboratory simulator, *IEEE Trans. Educ.,* **39**: 309–313, 1996. See also material on the accompanying CD-ROM in folder \ieeezips\007\cd\.
15. T. G. Engel, Splice: an analytical network analysis software, *IEEE Trans. Educ.,* **39**: 394–398, 1996. See also material on the accompanying CD-ROM in folder \ieeezips\023\cd\.
16. T. Boyle, *Design for Multimedia Learning,* London: Prentice-Hall, 1997.
17. R. A. Schwier and E. R. Misanchuk, *Interactive Multimedia Instruction,* Englewood Cliffs, NJ: Educational Technology Publications, 1993.
18. E. R. Doering, CircuitViz: a new method for visualizing the dynamic behavior of electric circuits, *IEEE Trans. Educ.,* **39**: 297–303, 1996. See especially material on the accompanying CD-ROM in folder \ieeezips\005\cd\.
19. R. O. Harger, Teaching in a computer classroom with a hyperlinked interactive book, *IEEE Trans. Educ.,* **39**: 327–335, 1996. See also material on the accompanying CD-ROM in folder \ieeezips\010\cd\.
20. S. L. Wood, A new approach for interactive tutorial software for engineering education, *IEEE Trans. Educ.,* **39**: 399–408, 1996. See also material on the accompanying CD-ROM in folder \ieeezips\026\cd\.
21. M. O. Hagler et al., Standards, the Virtual University and CD-ROM/WWW Technology, 1996 ASEE Int. Conf. Proc., CD-ROM, file /PAPERS/HAGLER.PDF.
22. J. B. Schodorf et al., Using multimedia to teach the theory of digital multimedia signals, *IEEE Trans. Educ.,* **39**: 336–341, 1996. See also material on the accompanying CD-ROM in folder \ieeezips\011\cd\.
23. J. McClellan, R. Schafer, and M. Yoder, *DSP First: a multimedia approach,* Upper Saddle River, NJ: Prentice-Hall, 1998.
24. A. J. Sears and S. E. Watkins, A multimedia manual on the World Wide Web for telecommunications equipment, *IEEE Trans. Educ.,* **39**: 342–348, 1996. See especially material on the accompanying CD-ROM in folder \ieeezips\012\cd\.
25. A. W. Hubler and A. M. Hassad, CyberProf: An intelligent human-computer interface for asynchronous widearea training and teaching. In Proc. 4th Int. WWW Conf., pp. 231–238, 1995. See also http://www.w3.org/pub/Conferences/WWW4/Papers/247/.
26. The URL of the NEEDS home page on the WWW is http://www.needs.org.
27. The URL for the World Lecture Hall is http://www.utexas.edu/world/lecture/index.html.
28. M. Hagler, Hardware homework for courses in circuits and electronics. In Lawrence P. Grayson (ed.), *Proc. 1994 Frontiers Educ. Conf.,* Piscataway, NJ: IEEE, 1994, pp. 557–561.
29. The URL of the Sloan Foundation home page on the WWW is http://www.sloan.org.
30. D. Minoli, *Distance Learning Technology and Applications,* Boston: Artech House, 1996.
31. R. A. Reiser and W. Dick, *Instructional Planning: A Guide for Teachers,* 2nd ed., Boston: Allyn and Bacon, 1996.
32. M. P. Driscoll, *Psychology of Learning for Instruction,* Boston: Allyn and Bacon, 1994.
33. E. R. Steinberg, *Computer-Assisted Instruction: A Synthesis of Theory, Practice, and Technology,* Hillsdale, NJ: Lawrence Erlbaum Assoc., 1991, Chaps. 3, 9, and 10.
34. R. M. Gagne, L. J. Briggs, and W. Wager, *Principles of Instructional Design,* 4th ed., New York: Rinehart and Winston, 1992.
35. W. Dick and L. M. Carey, *The Systematic Design of Instruction,* 4th ed. New York: Harper Collins, 1996.
36. M. D. Merrill, Constructivism and instructional design. In T. M. Duffy and D. H. Jonassen (eds.), *Constructivism and the Technology of Instruction: A Conversation,* Hillsdale, NJ: Lawrence Erlbaum Assoc., 1992, pp. 99–114.
37. M. D. Merrill, *Instructional Design Theory,* Englewood Cliffs, NJ: Educational Technology Publications, 1994.
38. S. A. Mengel and W. J. Adams, The need for a hypertext instructional design methodology, *IEEE Trans. Educ.,* **39**: 375–380, 1996.

39. J. M. Nyce and P. Kahn (eds.), *From Memex to Hypertext: Vannevar Bush and the Mind's Machine,* Boston, MA: Academic Press, 1991.
40. J. H. Larkin and R. W. Chabay (eds.), *Computer-Assisted Instruction and Intelligent Tutoring Systems: Shared Goals and Complementary Approaches,* Hillsdale, NJ: Lawrence Erlbaum Assoc., 1992.
41. C. Frasson and G. Gauthier (eds.), *Intelligent Tutoring Systems: At the Crossroads of Artificial Intelligence and Education,* Norwood, NJ: Ablex, 1990.
42. E. Wenger, *Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge,* Los Altos, CA: Morgan Kaufmann, 1987.
43. J. S. Brown and R. R. Burton, Multiple representations of knowledge for tutorial reasoning. In D. G. Bobrow and A. Collins, *Representation and Understanding: Studies in Cognitive Science,* New York: Academic, 1975, pp. 311–349.
44. J. S. Brown, R. R. Burton, and J. de Kleer, Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II, and III. In D. Sleeman and J. S. Brown (eds.), *Intelligent Tutoring Systems,* London: Academic, 1982, pp. 227–282.
45. A. Lesgold et al., SHERLOCK: a coached practice environment for an electronics troubleshooting job. In J. H. Larkin and R. W. Chabay (eds.), *Computer-Assisted Instruction and Intelligent Tutoring Systems: Shared Goals and Complementary Approaches,* Hillsdale, NJ: Lawrence Erlbaum Assoc., 1992, pp. 201–238.
46. D. A. Kolb, *Experiential Learning: Experience as the Source of Learning and Development,* Englewood Cliffs, NJ: Prentice-Hall, 1984.
47. B. G. Davis, *Tools for Teaching,* San Francisco: Jossey-Bass, 1993.
48. R. Felder, Matters of style, *ASEE Prism,* **5** (4): 18–23, 1996.
49. S. F. Chipman, Integrating three perspectives on learning. In S. L. Friedman, K. A. Klivington, and R. W. Peterson (eds.), *The Brain, Cognition and Education,* Orlando: Academic Press, 1986, pp. 203–229.
50. T. H. Duffy and D. H. Jonassen (eds.), *Constructivism and the Technology of Instruction,* Hillsdale, NJ: Lawrence Erlbaum Assoc., 1992.
51. R. A. Schwier, Issues in emerging interactive technologies. In G. J. Anglin (ed.), *Instructional Technology: Past Present and Future,* 2nd ed., Englewood, CO: Libraries Unlimited, 1995, pp. 119–130.
52. A. Collins, Design issues for learning environments. In S. Vosniaadou, E. De Corte, R. Glaser, and H. Mandl (eds.), *International Perspectives on the Design of Technology-Supported Learning Environments,* Mahwah, NJ: Lawrence Erlbaum Assoc., 1996, pp. 347–361.
53. B. Wilson, J. Teslow, and R. Osman-Jouchoux, The impact of constructivism (and postmodernism) on ID fundamentals. In B. B. Seels (ed.), *Instructional Design Fundamentals: A Review and Reconsideration,* Englewood Cliffs, NJ: Educational Technology Publications, 1995, pp. 137–157.
54. D. LeBow, Constructivist values for instructional systems design: Five principles toward a new mindset. In B. B. Seels (ed.), *Instructional Design Fundamentals: A Review and Reconsideration,* Englewood Cliffs, NJ: Educational Technology Publications, 1995, pp. 175–187.
55. C. T. Sun and C. Chou, Experiencing CORAL: design and implementation of distant cooperative learning, *IEEE Trans. Educ.,* **39**: 357–366, 1996.
56. S. R. Hiltz, *The Virtual Classroom: Learning without Limits via Computer Networks,* Norwood, NJ: Ablex, 1994.
57. M. O. Hagler et al., Publication of archival journals accompanied by CD-ROMs, *IEEE Trans. Educ.,* **40** (4): 1997, CD-ROM file \CDROM\HTML\JAVASCRP\JAVASCRP.HTM.
58. D. H. Jonassen, *Computers in the Classroom: Mindtools for Critical Thinking,* Englewood Cliffs, NJ: Merrill, 1996.
59. M. O. Hagler, Spreadsheet solution of partial differential equations, *IEEE Trans. Educ.,* **E-30**:130–134, 1977.
60. L. P. Huelsman, Electrical engineering applications of microcomputer spreadsheet analysis programs, *IEEE Trans. Educ.,* **E-27**:86–92, 1984.
61. A. Kharab and R. Kharab, Spreadsheet solution of hyperbolic partial differential equations, *IEEE Trans. Educ.,* **40**: 103–110, 1997.
62. B. R. Worthen, J. R. Sanders, and J. L. Fitzpatrick, *Program Evaluation: Alternative Approaches and Practical Guidelines,* New York: Longman, 1997.
63. P. Eibeck, Criteria for per-review of engineering courseware on the NEEDS database, *IEEE Trans. Educ.,* **39**: 381–387, 1996. See also material on the accompanying CD-ROM in folder \ieee-zips\021\cd\.
64. F. Stevens, F. Lawrenz, and L. Sharp, In J. Frechtling (ed.), *User-Friendly Handbook for Project Evaluation: Science, Mathematics, Engineering and Technology Education,* Washington, D. C., National Science Foundation, NSF 93-152, 1993.
65. I. Forsyth, A. Jolliffe, and D. Stevens, *Evaluating a Course: Practical Strategies for Teachers, Lecturers and Trainers,* London: Kogan Page, 1995.
66. P. F. Drucker, *Management: Tasks, Responsibilities, Practices,* New York: Harper and Row, 1974.
67. F. J. Rutherford and A. Ahlgren, *Science for All Americans,* New York: Oxford University Press, 1990, pp. 185–188.
68. M. Chiorico et al., The real experiment execution approach to networking courseware, *IEEE Trans. Educ.,* **40**: 1997 (CD-ROM folder 15).
69. M. O. Hagler (guest ed.), Special issue on the application of information technologies to engineering and science education, *IEEE Trans. Educ.,* **39**: 285–454, 1996. See also material on the accompanying CD-ROM.
70. T. E. Batchman (ed.), *IEEE Trans. Educ.,* **40**: 1997. See especially the CD-ROM Rapid Publication Supplement.
71. M. F. Iskander (ed.), *Computer Applications in Engineering Education,* past and current issues, New York: Wiley.

Marion O. Hagler
William M. Marcy
Texas Tech University