CRC Press - A Taylor & Francis Company
Posted with permission of the publisher.

The Human-Computer Interaction Handbook

Fundamentals, Evolving Technologies and Emerging Applications

Edited by
Julie A. Jacko
Andrew Sears

Human Factors and Ergonomics

·34·

COMPUTER-BASED TUTORING SYSTEMS: A BEHAVIORAL APPROACH

Henry H. Emurian
UMBC

Ashley G. Durham
Centers for Medicare and Medicaid Services

INTRODUCTION

Students are not equally advantaged to learn something new. They differ in such fundamental factors as preparation, motivation, maturity, study skills, and available energy, to name just a few. An optimal learning environment that takes such individual differences into consideration as factors that influence learning and the achievement of excellence may require a human teacher to interact with the individual student (Bloom, 1984). A teacher can continuously adjust a learning encounter based on the student's responses, and this feature is the essence of tutoring (Skinner, 1968, p. 37). The obvious assumption is that the teacher knows how to do that optimally, making such a human teacher a scarce resource, indeed. To the extent that the art and science of effective teaching can be codified as a set of enumerated principles and procedures that collectively determine an optimal learning environment for the individual student, it is reasonable to attempt to make those conditions available to all learners. This would enable learners to achieve a specific educational outcome without regard to the availability of a human teacher to monitor and manage the moment-by-moment process of learning by an individual student.

A computer-based tutoring system, as a generic teacher, has the objective of changing the behavior of a learner by structuring a series of interactive experiences with the system to achieve a criterion of mastery at the level of the individual student (Bostow, Kritch, & Tompkins, 1995). For the purposes of this chapter, a tutoring system is characterized by the requirement that a learner actively constructs responses and/or selects assessment options during an interactive dialog with the system. Those responses are evaluated for accuracy, and the outcome of that evaluation determines the sequence and content of the dialog events that follow. These events systematically change the behavior of the learner until the final educational objective has been achieved. A computer-based tutoring system is an information system that manages those interactive events.

This approach to operationalizing the essential features of a computer-based tutoring system allows the inclusion of programs that encompass a broad range of interdisciplinary applications and objectives, as long as those systems exhibit the above features. Such programs, then, might include drill and practice programs for acquiring domain knowledge and specific skills in such areas as mathematics, foreign language, and music (Alessi & Trollip, 1991; Merrill et al., 1996). Also included would be programs that foster generalized creative problem-solving skills (Ray, 1995) and those that teach introductory statistics (Shute, Gawlick, & Gluck, 1998). The potential range of programs is represented by the *looking at technology in context* framework for organizing learning theory and educational practice research from information transmission models (e.g., information presentation) studied in the research laboratory to constructivist models (e.g., information generation) applied to interconnected schools (Cognition and Technology Group at Vanderbilt, 1996).

The proposed operational definition of a computer-based tutoring system is to provide individual instruction to achieve specific task mastery by the user. As such, it would not include a consideration of the important stream of computer-based instructional delivery systems and authoring languages that commenced in the early 1960s with the use of the IBM 1500 at Penn State University and the University of Alberta (Buck & Hunka, 1992; Szabo & Montgomerie, 1992). It would also not include studies that show improvements in cognitive performance (e.g., Raven's Colored Matrices) that have been reported following a learner's use of such computer-delivered programs as Logo (Klein & Darom, 2000) or studies that show improvements in solving algebra word problems following a learner's use of a computer-based Word Problem-Solving tutor (Wheeler & Regian, 1999). These types of computer-based instructional delivery systems and tutorials, while demonstrably useful for groups of learners, may not always include the features that are critical to a focus on the individual learner's documented attainment of a criterion of mastery in a knowledge domain. These features are discussed later.

It is worthwhile, moreover, to acknowledge that since the invention of the computer and the diffusion of information technology into our lives, nothing has changed about the ways that people learn (Bransford, Brown, & Cocking, 2000). Learning is a process that includes the actions of study and practice (Swezey & Llaneras, 1997), sometimes for years (Ericsson & Lehmann, 1996), and the assessment of effectiveness as a change in the learner (Skinner, 1953, 1954), a change that might be observed and documented by other people or even by the learner as a self-evaluating authority. In the case of a computer-based tutoring system, a computer program documents that change in the learner's behavior, hence determining whether learning competency has occurred.

Theoretical and applied models of learning, as they relate to academic performance throughout the traditional school years from kindergarten through college, have been proposed and evaluated for centuries. Models of record include the pedagogy of Comenius (1657), Cicero's commentary on the importance of drill and practice (cited in Dale, 1967), cognitive science foundations of instruction (Rabinowitz, 1993), constructivism and the technology of instruction (Duffy & Jonassen, 1992), commentary on science and mathematics education (Bransford, Brown, & Rodney, 1999), and neurophysiological models of learning and memory (Rolls, 2000). To the extent that controversy continues to exist regarding pedagogy applied to formal educational settings, that controversy will not be resolved by the introduction of computer-based tutoring systems. It would be anticipated that the latter systems, however skillfully developed, would nonetheless reflect an incomplete understanding of basic issues in student learning (Tennyson & Elmore, 1997) and instructional design (Tennyson & Schott, 1997).

Nevertheless, providing opportunities to achieve a designated learning outcome at the level of the individual learner and across a wide range of applications has motivated the proliferation of computer-based tutoring systems as evidenced by the availability of commercial products and by the ever-increasing volume of research articles addressing the design, implementation, and effectiveness of these systems. The range

of available systems, from tutoring systems to teach the alphabet to children[1] to simulations to support flying an aircraft,[2] makes it a daunting task to categorize computer-based tutoring systems based on common features of learning theory and instructional design.

Recent reviewers of this developing literature, however, identify historical landmarks in this stream of work (Brock, 1997; LeLouche, 1998; Shute & Psotka, 1996; Szabo & Montgomerie, 1992). Such organizing perspectives can be valuable to illuminate the sources and consequences of progress in this field of investigation and applications.

One class of landmarks is attributable to naming the available computing hardware power and software complexity to handle a particular knowledge domain in terms of recording learner input, providing corrective interventions for input errors, managing knowledge about the learner, and implementing alternative instructional strategies. The focal points of such landmarks are often identified as (1) programmed instruction, (2) computer-assisted instruction, (3) and intelligent computer-assisted instruction. The boundaries are arbitrary, however, and there is a continuum of programmatic complexity whereby improvements in the quality and quantity of interactive events are sometimes anthropomorphized as possessing human-like qualities (e.g., intelligence).

A second class of landmarks is attributable to tutoring system designs that craft a set of interactions based on theoretical models of cognitive functioning. The latter approach is exemplified by the Advanced Computer Tutoring Project at Carnegie Mellon University (Anderson, Corbett, Koedinger, & Pelletier, 1995). The interactive events programmed in the *cognitive tutors* are based on a production-rule theory of cognitive skill development, the ACT-R theory (Anderson, 1993), whereby the learner acquires cumulative units of goal-related knowledge. In this stream of work is included the research by Sohn and Doane (1997) that adopted a cognitive theoretical approach to understand how knowledge and processing demands influenced a user's acquisition of UNIX™ commands in a computer-based system. For the most part, however, the evolution of such computer-based systems to manage increasingly complex knowledge domains and instructional strategies reflects improvements in computer processing speed and memory, rather than a change in our understanding of the principles of learning (Szabo & Montgomerie, 1992).

Against that background, this chapter will focus upon an *atheoretical* approach to tutoring system design. This approach is based on the behavioral tradition of crafting a series of learner-environment interactions to support the progressive development of a complex repertoire. It is *atheoretical* in that it will focus on the interactions themselves as the explanation of the antecedents to knowledge and skill acquisition, and it will rely on those antecedents, rather than external explanatory metaphors, such as cognitive models, to explain the process and outcome of learning.

[1]JumpStart Toddlers by Knowledge Adventure, Inc.

[2]F-22 Air Dominance Fighter by Infogames Entertainment, Inc.

PROGRAMMED INSTRUCTION

Programmed instruction presents information to a learner in a planned sequence of steps that are interspersed with direct questions or other prompts (e.g., blanks) requiring a student's constructed response to demonstrate competency in the material presented. A *frame* consists of a unit of information together with the assessments to document learning. The essential features of programmed instruction are as follows: (1) comprehensibility of each unit; (2) tested effectiveness of a set of frames; (3) skip-proof frames; (4) self-correcting tests; (5) automatic encouragement for learning; (6) diagnosis of misunderstandings; (7) adaptations to errors by hints, prompts, and suggestions; (8) learner-constructed responses based on recall; (9) immediate feedback; (10) successive approximations to a terminal objective; and (11) student-paced progress (Scriven, 1969; Skinner, 1958; Vargas & Vargas, 1991). A program of study consists of many frames designed to promote the achievement of a demonstrable set of competencies, for the individual learner, at the conclusion of study.

A frame-based program of study is intended to promote generalized understanding, rather than rote memorization. Table 34.1 presents an example of a series of frames that leads a learner to understand the concept of a conductor (Deterline, 1962, p. 17). Several frames in this example, to include the final frame, do not lend themselves to automated assessment, and the importance of a personal interaction between a student and a teacher, at some point in a frame-based instructional system, is addressed in the personalized system of instruction discussed later.

There is nothing novel about this instructional technology. In the *Meno*, Socrates taught a slave boy the proof of the Pythagorean theorem by using simple diagrams to lead the learner, by small incremental steps, to significant generalizations (Corey, 1967). The anticipated use of computers in implementing this individualized instructional technology was stated over 30 years ago (Scriven, 1969, p. 15):

> The 'far-out' entry in the teaching stakes (from the viewpoint of the 1960s) is the super teaching machine, the teaching computer. The idea is straightforward. The computer begins with certain background data about the student, and feeds him a few appropriate . . . frames of instruction. On the basis of his responses to these frames, plus the background data (previous courses and grades, I.Q., etc.), the computer chooses the most appropriate instructional sequences for the next section of the program—and so on.

This latter description is prescient of adaptive systems (Benyon & Murray, 1993) and intelligent tutoring systems (Shute & Psotka, 1996).

The intellectual force behind the development of programmed instruction is attributable to Harvard psychologist B. F. Skinner. In an early landmark paper entitled *The Science of Learning and the Art of Teaching* (Skinner, 1954), Skinner

TABLE 34.1. A Series of Frames

1. A conductor will carry electric current. A wire or any substance that will carry or conduct an electric current is called a _____.	conductor
2. A copper wire will conduct or carry an electric current because copper wire is a good _____.	conductor
3. A conductor is a substance that will carry or _____ an electric current. Rubber is not a conductor, so rubber will not _____ an _____.	conduct conduct electric current
4. An insulator will not conduct an electric _____. Rubber is a good _____ because it will _____. (complete)	current insulator not conduct an electric current (or) not conduct current
5. Electric current can flow or travel along a _____, but cannot flow along an _____.	conductor insulator
6. You could receive a "shock" from a copper wire unless the copper wire is surrounded by an _____.	insulator
7. An insulator is a substance or material that will _____. (complete)	not conduct electric current (or) not let current flow (or) stop current
8. A conductor will _____. (complete)	conduct an electric current (or) carry current

Note. From W. A. Deterline, *An Introduction to Programed Instruction*, by W. A. Deterline, 1962, New York: Allyn & Bacon. Copyright 1962 by Allyn & Bacon. Reprinted with permission.

argued how principles derived from the laboratory experimental analysis of behavior could be directly applied to education. The contribution was the argument that the intended changes in a student's behavior—the goal of education—are a function of lawful behavioral processes that follow directly from the skillful presentation of contingencies of reinforcement that promote such changes and their maintenance over repetition and time at the level of the individual student. The focus was the application of experimentally derived principles that would foster the development of increasingly complex repertoires of behaviors, at the level of the individual learner, that are otherwise taken to be evidence of *understanding*, *thinking*, *problem solving*, *cognitive functioning*, and so on. The challenge, of course, was to design a course of study whose outcome would provide convincing evidence that such objectives had been achieved. But the objective of this teaching-oriented strategy was not to demonstrate that one teaching method was superior to another teaching method, as evidenced by comparisons between group averages. Rather, the objective was to determine ways to design an educational environment so that each learner can attain a specific level of competency in a subject matter (Bijou, 1970).

Skinner referred to the need for *mechanical help* (Skinner, 1954, p. 29) in the management of the many thousands of contingencies of reinforcement that would be required to apply these principles to the individual learner, without regard to the knowledge domain. In that latter regard, the history of published attempts to provide automated and mechanized support for student learning can be traced at least as far back as the work of Pressey (1927), who reported the design of a machine to administer and score tests, and most importantly, to teach. The purpose of the machine was to free the teacher from overseeing and managing a student's moment-by-moment progress of essential drill, thereby freeing the teacher to meet the needs of the individual learner as required. The continuation of this approach to individualized instruction is exemplified by Skinner (1958). In this latter paper, the application of behavior principles to the design of frames of material was elaborated as an extension of the early contribution of Pressey (1927).

The emphasis on a teaching technology that focused on the individual learner generated a stream of research and scholarly books on programmed instruction (e.g., Calvin, 1969; Green, 1967; Holland, 1960; Lange, 1967; Margulies & Eigen, 1962). One of the first presentations of a programmed instruction text was authored by Holland and Skinner (1961). The book itself was a frame-based presentation of introductory psychology material. This text was followed by journal articles that began to focus on the elements of programmed instruction, such as the frequency of overt responses (Kemp & Holland, 1966) and techniques for instructional design of material (Holland, 1967). The early use of a linear progression of frames was extended to branching approaches in which branching paths to a learning

objective were determined by a history of prior interactions with the material, not just an immediately prior frame (Crowder, 1962). This latter approach also adopted multiple-choice tests of competency to determine succeeding paths in learning.

Meta-analyses of the effectiveness of programmed instruction in higher education were generally supportive (Kulik, Cohen, & Ebeling, 1980), but research and corresponding journal activity declined during the 1970s. Moreover, some scholarly sources broadened their focus. For example, *Programmed Learning and Educational Technology*, a journal published by the Association for Educational and Training Technology since 1967, changed its title to *Education and Training Technology International* in 1989 and again to *Innovations in Education and Training International* in 1995. There were several factors influencing this trend.

Although the design of programmed instruction material to foster such intellectual skills as "thinking" had been discussed (Peel, 1967), it became increasingly apparent that it was difficult to operationalize many important objectives of education, such as mastering concepts, principles, rules, and cognitive strategies, as well as mastering the intellectual skills of reading, writing, and mathematical problem solving (Case & Bereiter, 1984). The inability of the behavioral approach to provide guidance on formulating the size of incremental steps leading to a terminal performance motivated learning theorists such as Gagne (1968) to propose an instructional technology based on hierarchical task analysis for identifying and sequencing intellectual skill development. Furthermore, the readiness of a learner to acquire new skills was recognized to require a consideration of developmental factors, not just learning technology factors (Case, 1978). These important contributions to an ontology of learning reflect an attempt to organize the stages in learning where specification of the exact learner–environment interactions taking place within those stages has yet to be realized. This is evidenced by the use of labels applied to learners, such as *intelligent*, *exceptional*, *average*, *novice*, and *expert*, to explain the outcome of a process of learning rather than the instructional events that account for the outcome itself. Operationalizing the instructional events, in terms of learner–media interactions, is a requirement for programmed instruction that is often problematic.

These latter developmental orientations are not necessarily incompatible with programmed instruction approaches, and the two differ primarily in terms of the methods used to group and classify the many functional components and interrelationships of a developing complex repertoire. In the history of educational technology, it is often the case that "educated person" is a term applied to a complex repertoire that is a by-product of pedagogical customs in which the specification of all the steps leading to an intended educational outcome is problematic. The importance of this is to be understood in terms of its impact on the expectations of the development of computer-based programmed instruction tutoring systems, which require an algorithm of interaction and knowledge acquisition that may not exist for very advanced areas of intellectual functioning. It is this combinatoric problem related to student–system interactions that motivated many early proponents of automated instructional systems—programmed instruction—to embrace a contextual orientation that included developmental readiness for learning (Case, 1978) and the achievement of superordinate objectives whereby the learner is taught to apply self-regulation strategies to his or her own knowledge acquisition processes (Schunk, 2000).

PERSONALIZED SYSTEM OF INSTRUCTION

Programmed instruction material, by itself, provides an incomplete set of pedagogical tools. Some reports question its assumptions about the effectiveness of stepwise progress as sufficient motivation to engage a learner's attention (Tennyson & Elmore, 1997), and other studies suggest that students are not enthusiastic about the material's format (Day & Payne, 1987). Much earlier, however, Keller (1968) had proposed a comprehensive learning environment that included programmed instruction as a potential, but not necessary, component. This learning environment came to be known as the personalized system of instruction (PSI), and its design may overcome at least some of the objections about programmed instruction material.

The essential features of PSI are similar to programmed instruction, and they are as follows: (1) self-paced progression, (2) unit perfection across success stages in learning, (3) lectures as vehicles of motivation rather than as sources of critical information, (4) the emphasis on text to transmit information, and (5) the use of proctors in an interpersonal interaction for testing. It is this last feature that best distinguishes PSI from programmed instruction, although none of the components of PSI is necessarily frame-based after the design of a programmed instruction system. The social interaction between student and proctor takes place after study of material by the student. The interaction is intended to broaden the student's repertoire into the "... understanding of a principle, a formula, or a concept, or the ability to use an experimental technique" (Keller, 1968, p. 84). It is this assessment of *understanding* as a shared history between a speaker and a listener (Skinner, 1957, p. 280) that has led to a consideration of the importance of synchronous and asynchronous collaborative learning environments in which learners seek an equilibrium of understanding where the latter is based on overlapping agreement by members of a verbal community (e.g., Jehng & Chan, 1998).

Because the combinatoric challenges of programmed instruction approaches may falter in this later regard, an informed dialog between a student and a teacher is a test of mutual understanding and a method of remediation of higher order intellectual skills whose component parts are difficult to operationalize at a level of rigor necessary for automated instructional systems (Emurian, 2001). Ferster and Perrott (1968) adopted the PSI in a comprehensive learning environment, which included frequent assessment of a learner's competency by a proctor. Finally, the effectiveness of the PSI was demonstrated in a series of meta-analyses (Kulik, Kulik, & Bangert-Drowns, 1990; Kulik, Kulik, & Cohen, 1979) and integrative commentary (Buskist, Cush, & DeGrandpre, 1991).

A modified PSI approach is used in conjunction with a Java-programmed instruction tutoring system that is discussed later. The approach is modified in the sense that the interpersonal

interactions among students and among instructors and students at the conclusion of the tutoring experience are more collaborative than proctoring. A modified approach to PSI was also reported by Crosbie and Kelly (1993) in a course in which computers were used primarily to administer and score objective tests automatically, and proctors were used in that course to confirm the test outcomes and to discuss incorrect student responses that were challenged by the student.

COMPUTER-BASED SYSTEMS

As early as 1961, researchers began to focus on the use of computers to automate programmed instruction systems (Coulson, 1962a). The CLASS system was a Philco 2000-based programmed instruction system supporting up to 20 learners (Coulson, 1962b). It provided a branching approach to instructional design, and it was intended to be one component in the entire complex of educational support, such as films, television, lectures, and textbooks. The PLATO II system was an ILLIAC-based programmed instruction system initially supporting two concurrent learners (Bitzer, Braunfelf, & Lichtenberger, 1962). Both the CLASS and PLATO II systems were based on the principles of programmed instruction, and both systems were able to support the multimedia information delivery that was available at that time. An example of a frame from the PLATO II system is presented here.

In the PLATO II system, textual information for both learning and testing was presented to the student by a television display, and a student's responses were input by a keyset. Figure 34.1 presents an example "text" slide displaying domain knowledge, and Fig. 34.2 presents an example of an "answer" slide. Correct input occasioned progression in the sequence, and incorrect input provided the opportunity to review the previous text slide and to branch to a series of smaller step remedial text slides. The authors anticipated the *student model* by suggesting

3 –

EACH POSITIVE INTEGER IS REPRESENTED IN DECIMAL NOTATION BY COMBINING THE TEN DIGITS:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

THUS THE SYMBOL '3,549' IS INTERPRETED TO MEAN:

$3 \times 10^3 + 5 \times 10^2 + 4 \times 10 + 9$

i.e., $3 \times 1000 + 5 \times 100 + 4 \times 10 + 9$.

FIGURE 34.1. An example "text" slide displaying domain knowledge from the PLATO II system. PLATO II: A multiple-student, computer-controlled, automatic teaching device. From *Programmed Learning and Computer-Based Instruction* (p. 210), by J. E. Coulson (Ed.), 1962, Santa Monica, CA: System Development Corporation. Copyright 1962 by System Development Corporation. Reprinted with permission of John Wiley & Sons, Inc.

QUESTION: GIVE THE POSITIVE, NONTRIVIAL DIVISORS OF 51 IN INCREASING ORDER.

$d_1 =$ [ ]

$d_2 =$ [ ]

FIGURE 34.2. An example "answer" slide from the PLATO II system. PLATO II: A multiple-student, computer-controlled, automatic teaching device. From *Programmed Learning and Computer-Based Instruction* (p. 211), by J. E. Coulson (Ed.), 1962, Santa Monica, CA: System Development Corporation. Copyright 1962 by System Development Corporation. Reprinted with permission of John Wiley & Sons, Inc.

the presentation of information based on a diagnosis of student errors, together with the history of successful prior interactions with the system.

The decline in journal activity related to programmed instruction was offset by the emergence of computer-assisted instruction, as evidenced by the CLASS and PLATO II systems. One of the first reports of a computer-based instructional program to appear in the general scientific literature was published in the journal *Science* in 1969 (Suppes & Morningstar, 1969). The fact that such a program was published in this prestigious scientific journal was evidence of the recognition of the potential impact of these pedagogical approaches to formal education.

Suppes and Morningstar (1969) reported the results of the use of computer-assisted instruction that provided drill and practice in the skills of arithmetic computation for students in grades one through six. The results of a multiyear project over several states generally showed greater improvement, determined by changes in scores on a test administered before and after the 189 days of the study, by students who had used the computer-assisted instruction in comparison with control students who received conventional classroom instruction, which often included drill and practice at a group level. The computer was a PDP-1, and the student interacted with the system by way of a model-33 teletype.

The teletype printed each individual problem and then positioned itself in readiness to accept the answer in the appropriate place. The student typed in the answer. If his answer was correct, he proceeded to the next problem. If he gave the wrong answer, the teletype printed 'No, try again,' and presented the problem once more. If the student gave the wrong answer for the third time, he was given the correct answer and the teletype automatically proceeded to the next problem (Suppes & Morningstar, 1969, p. 344).

The system exhibited *branching intelligence* and *adaptivity*, in that successive daily drill sessions were adjusted in difficulty depending on the percentage of problems that were solved

correctly in a given session. The system also selected a review drill based on the student's performance over several prior successive sessions. Computer-assisted drill and practice followed an introduction of concepts by a teacher in the classroom.

Although the presentation by Suppes and Morningstar (1969) did not cite any behavioral principles or literature, procedurally the computer-based tutoring system exhibited many of the features of programmed instruction to include self-paced progression, immediate feedback for performance, and the *successive approximation* of skill in arithmetic computation by the presentation of increasingly more difficult problems across successive practice sessions. The one notable difference was that a learner was advanced in the program after three attempts at entering the correct answer, even if the third attempt was incorrect. A strict behavioral approach might have required a correct solution prior to the program's advancement to another problem. However, the software, even though sophisticated for 1965, did not have the capability to diagnose the source of learner errors and to put the student on a remedial path. That capability was to appear later in the development of computer-assisted instruction.

Despite these promising background developments in computer-delivered programmed instructional material, the trend in research and applications over the next two decades or more gave witness to the broadening interdisciplinary scope of computer-based instructional systems as evidenced by the reviews cited earlier (Brock, 1997; LeLouche, 1998; Shute & Psotka, 1996; Szabo & Montgomerie, 1992). Studies that focused on programmed instruction, per se, and computer-based technology began to re-emerge in the behavioral literature only in the early 1990s. One of the first such studies was conducted by Tudor and Bostow (1991) who investigated the presentation of a 315-frame instructional program, controlled by an IBM PC Junior® microcomputer, to teach college students how to construct effective programmed instruction. Comparisons on a competency test administered before and after the students' use of the tutoring system showed gains in knowledge for an experimental group whose members were required to recall and type each tested item for a frame of information. Importantly, an integrative posttutor assessment that required the construction of a sample frame, based on all principles taught, showed the superiority of the instructional condition that required overt constructed responses during knowledge acquisition that occurred over the completion of the 315 separate frames in the tutoring system.

These results were confirmed in a subsequent study (Kritch & Bostow, 1998) that showed improved posttutor performance when learners emitted a relatively high density of constructed responses across 176 successive frames, in comparison with a low-density condition. Figure 34.3 presents a summary of the experimental conditions and an example of three frames, three posttutor test items, and three questionnaire items. The first frame shows the use of *prompting* and *matching*, because the correct response is embedded in the frame and because the answer is partially presented. The third frame shows the requirement for recall under conditions of *prompting*. The first sample test item shows the requirement for *response generality*, which occurs as a function of completion of all frames. These features together reflect the method of *successive approximations* to the objectives of learning (Sulzer-Azaroff & Mayer, 1991, p. 324). Finally, the importance of constructed responses, as they enhance the accuracy of posttutor assessments, was also demonstrated when the frames were prepared on videodisc that presented textual, graphics, and audio information (Kritch, Bostow, & Dedrick, 1995).

Another stream of research in the behavioral literature studied effects of delay intervals between constructed responses and progression in a computer-based tutoring system on response accuracy. Crosbie and Kelly (1994) presented 1,711 programmed instruction frames of the textbook by Holland and Skinner (1961) on an IBM-compatible PC to learners over 15 consecutive daily sessions. It was found that the presentation of a 10-sec postresponse delay interval, during which the frame material, the learner's constructed response, and the correct responses were simultaneously displayed, improved response accuracy overall. The researchers speculated that this effect was attributable to a diminution of *racing*, which was associated with a learner's insufficient study of the presented frames and with careless errors that might be easily correctable when incorrectly answered frames were repeated. These effects were confirmed in a systematic replication (Kelly & Crosbie, 1997) in which test performance of learning was superior for the imposed delay condition even after 1 month following completion of the computer-based programmed instruction tutoring system.

These background studies show, for the most part, that applications of information technology to the implementation of programmed instruction approaches have emphasized the student's learning of textual information in relatively simple knowledge domains. Recent studies have manipulated the parameters of the human–computer interaction, with the intention to improve instructional design.

THE LEARN UNIT

The contingencies of reinforcement underlying the steps in a programmed instruction approach have been conceptualized as a *learn unit* (Greer & McDonough, 1999). This unit consists of (1) discriminative stimulus for a response, the response itself, and a reinforcer consequence presented for response accuracy; and (2) branching or corrective remediation presented for response inaccuracy. A learn unit is specified with sufficient operational rigor to be countable. The size of the required response may change as a learner progresses through a programmed instruction tutor. Several elementary units, then, may combine into a higher order unit in which reinforcement is provided for the production of several prior and smaller units. This approach provides a mechanism for documenting and quantifying the total "learning force" required to progress from simple to complex units of performance. It is this approach that was adopted in the case studies to follow.

CASE STUDIES

Over the past several years, our work has focused on the investigation of factors related to the acquisition and retention of UNIX™ and Java™ by undergraduate and graduate students

High Density (HD) Overt responses to 176 frames	Low Density (LD) Overt responses to every other frame	Zero Density (ZD) Passive reading, key tapping to advance	Control for Time (CT) Passive reading, advance when HD advanced
1. A player piano is told what notes to play from a long scroll of paper with tiny holes punched through it. The paper scroll is like a script of commands that tells the piano what n---s to play.	1. A player piano is told what notes to play from a long scroll of paper with tiny holes punched through it. The paper scroll is like a script of commands that tells the piano what n---s to play.	1. A player piano is told what notes to play from a long scroll of paper with tiny holes punched through it. The paper scroll is like a script of commands that tells the piano what notes to play.	1. A player piano is told what notes to play from a long scroll of paper with tiny holes punched through it. The paper scroll is like a script of commands that tells the piano what notes to play.
2. With a player piano, the music is programmed. The scroll of paper is a script of commands that tells the player ---- what notes to play.	2. With a player piano, the music is programmed. The scroll of paper is a script of commands that tells the player piano what notes to play.	2. With a player piano, the music is programmed. The scroll of paper is a script of commands that tells the player piano what notes to play.	2. With a player piano, the music is programmed. The scroll of paper is a script of commands that tells the player piano what notes to play.
3. Like a player piano, a computer can be programmed. A computer program is like a sc---pt of commands that tells the c------r what to do.	3. Like a player piano, a computer can be programmed. A computer program is like a sc---pt of commands that tells the c------r what to do.	3. Like a player piano, a computer can be programmed. A computer program is like a script of commands that tells the computer what to do.	3. Like a player piano, a computer can be programmed. A computer program is like a script of commands that tells the computer what to do.

Yoked

Sample Test Items
1. The statements that cause a computer program to take actions are called -----.
2. The command that erases any previous material from the screen is the ---- command.
3. The command that tells the program to start a new frame is the ---- ---- command.

Questionnaire Items (1=very much dislike; 2=dislike; 3=neutral; 4=like; 5=very much like)
How would you describe your attitude about the instructional program that you experienced today?
How would you describe your attitude about computer assisted instructional programs in general?
How would you describe your attitude about computer assisted instructional programs that specifically teach program commands like those taught in the instructional program you just experienced?

FIGURE 34.3. A summary of the experimental conditions and an example of three frames, three posttutor test items, and three questionnaire items. The first row gives the experimental density conditions, which reflect how much interaction was required by the learner to progress from one frame to the next. From "Degree of Constructed-Response Interaction in Computer-Based Programmed Instruction," by K. M. Kritch and D. E. Barstow, 1998, *Journal of Applied Behavior Analysis*, 31, 387–398. Copyright 1998 by the Society for the Experimental Analysis of Behavior. Reprinted with permission.

who are information systems majors. The design of computer-based tutoring systems to accomplish the objectives follows the principles of programmed instruction. What follows is a presentation of a series of studies showing the transition of this instructional technology from the research laboratory to the classroom.

Unix Tutor

The first study compared the effectiveness of a learner's acquisition and retention of sequences of UNIX commands under conditions of recognition, recall, and a combination of recognition and recall. The details of the procedure and the theoretical

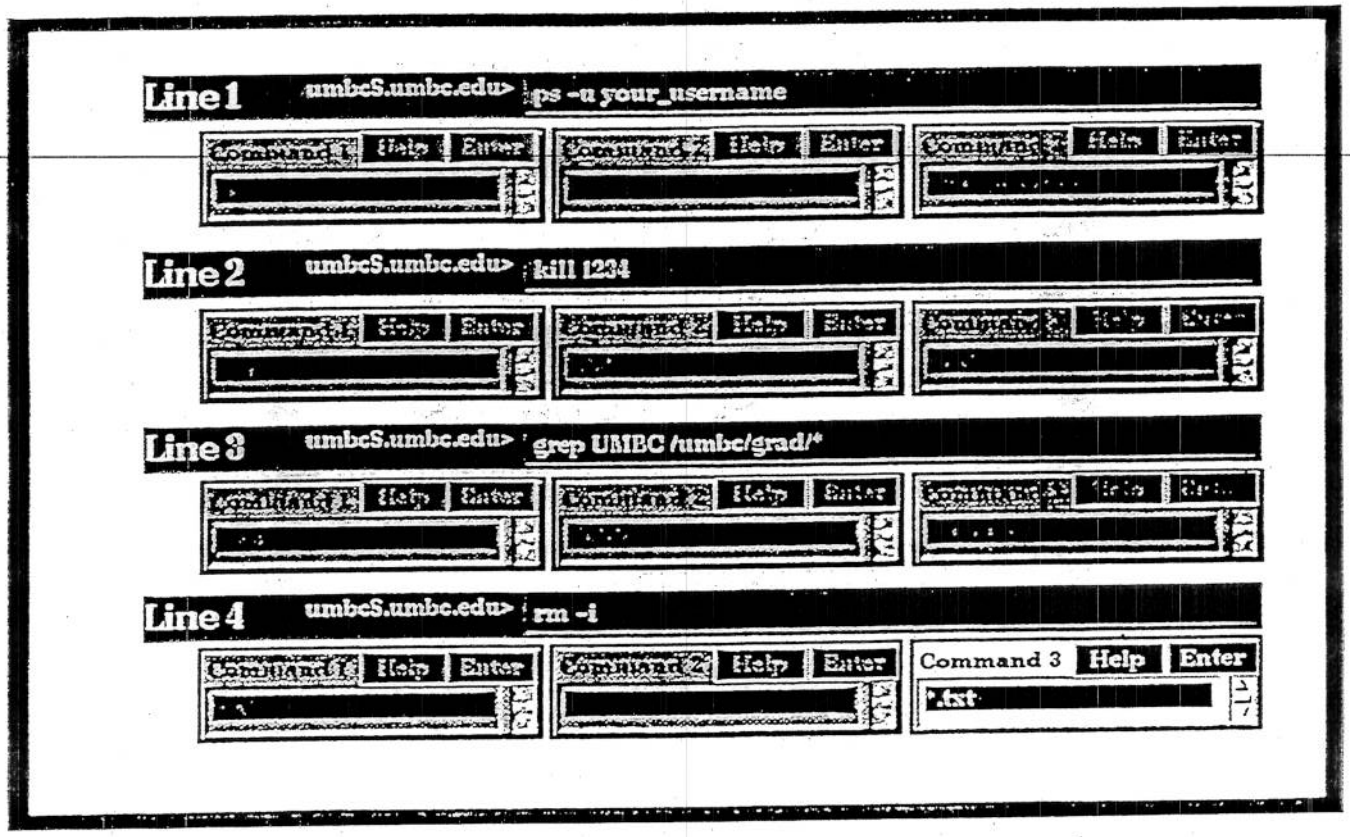

FIGURE 34.4. An example of the menu interface. umbc = UMBC. From "Learning and Retention With a Menu and a Command Line Interface," by A. G. Durham and H. H. Emurian, 1998, *Computers in Human Behavior,* 14, 597–620. Copyright 1998 by Elsevier Science. Reprinted with permission.

rationale for investigating these conditions are presented in Durham and Emurian (1998a). Within the context of a computer-based tutoring system, 36 student participants initially learned a series of up to 12 UNIX commands that were required to satisfy a common file, directory, or process manipulation objective. Initial learning occurred with a menu interface (recognition), a command line interface (recall), and a hybrid menu-command line interface (recognition-plus-recall). After a 4-week interval, participants repeated the task.

Figure 34.4 presents an example of the menu interface. Four three-item lines of UNIX commands and arguments were selected to achieve an objective involving common file, directory, and process manipulations. Within each Command Box was a scroll window with all the possible 51 commands and arguments used in the study. Each Command Box contained the identical list of possible commands and arguments, and each box was a separate *knowledge unit*, designed to teach the input for that particular field.

Each line of acceptable input contained a maximum of three fields, but no fewer than two fields: three columns of field values within a row of items that could be entered as one continuous line, including spaces, at the UNIX system prompt. The participant was presented with a description of the objectives of each series of commands, e.g., remove the read/write protection from a-file.txt; print a-file.txt; delete a-file.txt; return to home directory. The correct value for each field in a row was obtained by clicking the "Help . . ." button within the Command Box. When help was selected, a pop-up panel appeared describing and displaying the UNIX command or argument to be selected.

The second interface condition required the acquisition to occur under conditions of item recall by substituting a command line interface for the selection list interface. An error was recorded whenever the subject entered an incorrect UNIX command or left the field blank. Spelling errors or strings not within the list of acceptable commands in the tutor cleared the input field, but they did not count as an error. The features of the command line interface best reflect the programmed instruction approach.

The third interface condition replicated the previous two procedures by synthesizing the menu and command line interfaces. The task consisted of successive iterations through the selection-based menu interface and the keyin command line interface, presented sequentially.

For the three interface conditions, trials for each successive line of input were repeated until the subject entered all three fields correctly without committing an error or requesting help. A criterion of task mastery was reached when three successive error-free and help-free passes for each of the three objectives

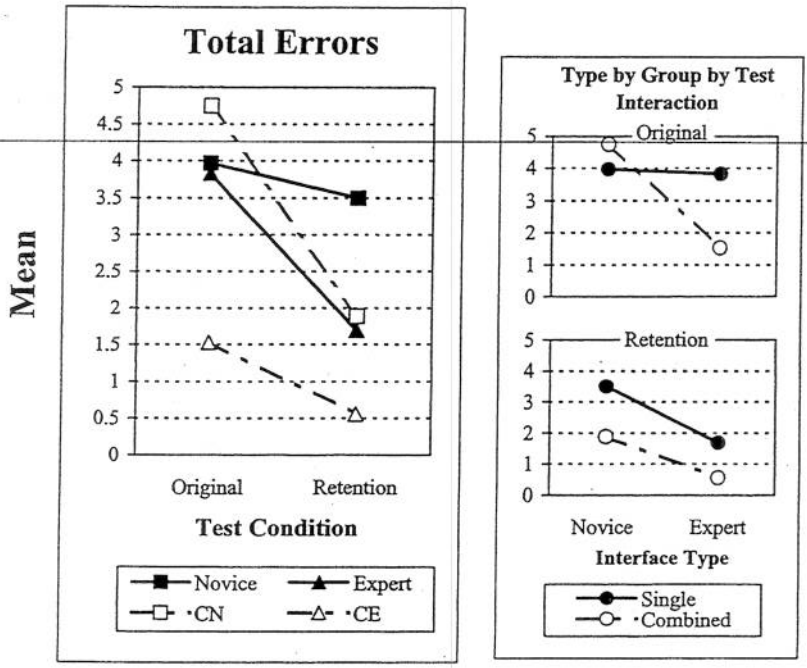

FIGURE 34.5. Mean total errors for all participants for the menu and command line interfaces within the single and combined presentation groups across original and retention testing. From "Learning and Retention With a Menu and a Command Line Interface," by A. G. Durham and H. H. Emurian, 1998, *Computers in Human Behavior*, 14, 597–620. Copyright 1998 by Elsevier Science. Reprinted with permission.

were achieved. After a 4-week interval, the sequence of events was replicated, with exactly the same tasks and mastery criteria.

The terminal performance was the learner's correct production of the 12 UNIX commands, as a serial stream, with no error and no help selection. Each command was first mastered as a single item in context, and each command box was a single learn unit. The learner repeated the interaction with a single command box until the command had been entered accurately. The next level of learn unit was the line consisting of the three successive items in that row. The learner repeated the line until it was entered with no error and no help selection on any Command Box in the row. The highest level of learn unit was the production of all 12 commands with no error and no help selection for three successive iterations through the interface. Thus, there was a maximum of 17 learn units in this programmed instruction tutoring system.

Figure 34.5 presents mean total errors for all participants for the menu and command line interfaces within the single and combined presentation groups across original learning and retention testing. Errors declined between original learning and retention testing, and a three-way interaction was observed. Without regard to the presence of a *transfer-appropriate processing effect* (Roediger & Guynn, 1996), which predicts facilitation from recognition to recall, the importance of acquisition under conditions of recall was revealed during retention testing. Recall learning was robust, and this outcome, which was observed in the research laboratory, influenced the design of our computer-based tutoring system for Java.

The importance of this background study is to be understood in terms of providing baseline knowledge for the instructional designer. As a programmed instruction tutoring system, the outcome was achieved by symbol constructions whose accurate productions led to progress through the tutor itself. Thus, the contingencies were artificial, and the reinforcers were not programmed for the learners to produce consequences that would otherwise be anticipated to occur when one uses these commands in a real-life setting to achieve important objectives (Ferster, 1967). Thus, the meaning of the symbols to the learner was related to a synthetic context, constraining the generality of the outcomes (Sidman, 1960). Although the artificial symbolic manipulations supported the value of active recall, it was essential to broaden the scope of these research findings to a realistic task environment. That was the motivation for the next development in this work.

Java Tutoring System

The next series of studies extended this background approach to the design and implementation of a programmed instruction tutoring system for a Java Applet.[3] This system was intended to

[3]The tutoring system is freely accessible on the World Wide Web: http://webct.umbc.edu/public/JavaTutor/index.html

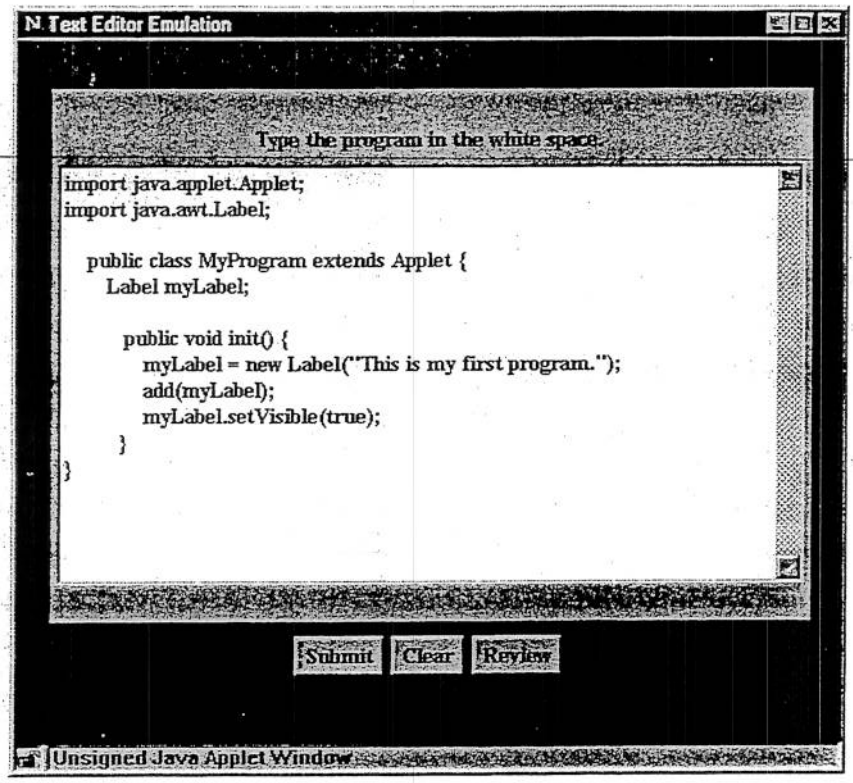

FIGURE 34.6. The code for the Java Applet that displays a Label object in a browser window. The code is displayed in an interface explained below. From *Managing Information Technology in a Global Environment* (pp. 155–160), by H. H. Emurian and A. G. Durham, 2001, Hershey, PA: Idea Group Publishing. Copyright 2001 by Idea Group Publishing. Reprinted with permission.

be used in the classroom as a component of a laboratory course in graphical user interface design. The rationale for using this tutoring system was to ensure a common and documented history in symbol manipulation in a group of students. This background set of experiences provided a competency reference point from which to determine the readiness of students to benefit from the instructional approaches that came later in the course. Details regarding this tutoring system have been published elsewhere (Emurian, Hu, Wang, & Durham, 2000).

Figure 34.6 presents the code for the Java Applet that displays a Label object in a browser window. The code is displayed in one of the learning interfaces described here. The production of the displayed stream of symbols is the terminal performance, the objective of learning. The program consists of 24 atomic items of code concatenated in a stream of 36 total items. The tutoring system design was intended to foster the learning of the individual items, the meaning of each item, the relationship of one item to another, the relationship of one item to the entire program, and the errorless production of the final stream of items.

The several stages in the tutoring system were based on a functional classification of verbal behavior that is assumed to underlie the acquisition of the form and meaning of textual information in context (Skinner, 1957), and the acquired functional interrelationships among the information items are fostered after the techniques in verbal memory studies (Li & Lewandowsky, 1995). The progression from general context through details and synthesis follows the elaboration theory of instruction (Reigeluth & Darwexeh, 1982). Repetitions of components of the interfaces to be described are grounded in the power function of learning (Lane, 1987). Classroom applications are based on the personalized system of instruction (Keller, 1968). Although initial explorations of this tutoring system were undertaken with undergraduate and graduate students in information systems courses, the learner was assumed to have no prior knowledge of Java and no differentiated motivation for undertaking mastery of the program (Coffin & MacIntyre, 1999; Ryan, 1999).

Figure 34.7 presents the first two interfaces. The left view shows the symbol familiarity interface that requires the learner to copy the displayed symbol into the keyin field. The purpose of this interface is to generate a minimal transcription repertoire (Skinner, 1957) with the symbol set prior to the acquisition of a symbol's meaning. The successive symbols are the atomic units in the program, and this sequence affords prior nondifferential exposure to the code, which itself generates associative learning by contiguous temporal and sequential pairing. The

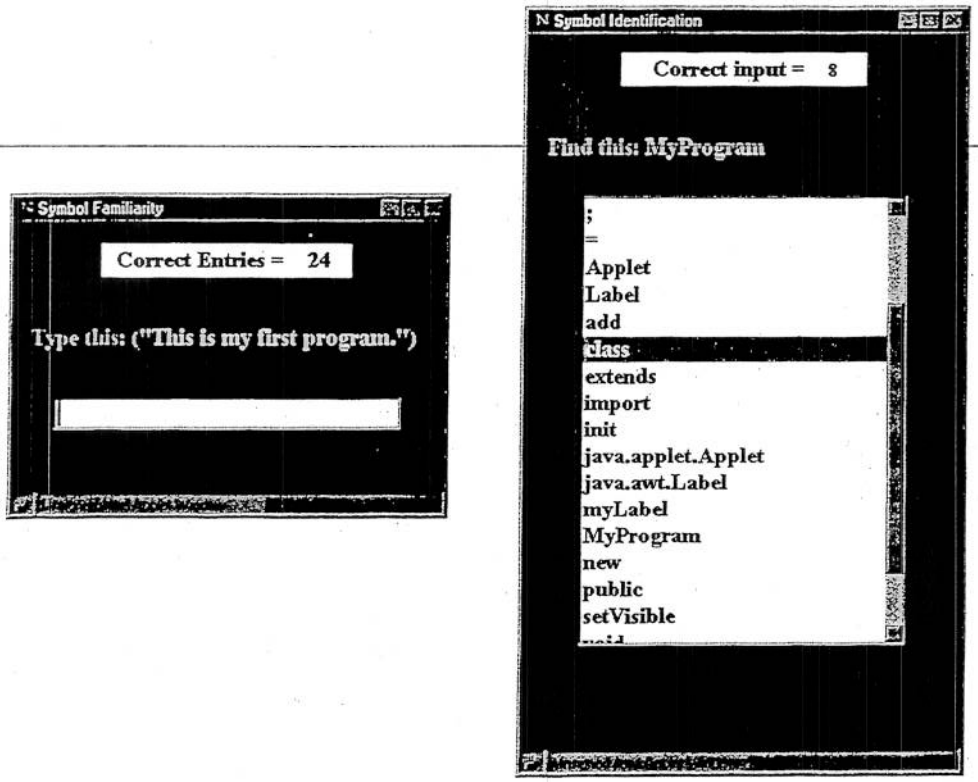

FIGURE 34.7. The first two interfaces. The left view shows the symbol familiarity interface that requires the learner to copy the displayed symbol into the keyin field. The right view shows the symbol identification interface that requires the learner to select the displayed symbol from a list of all symbols. From "Learning Java: A Programmed Instruction Approach using Applets," by H. H. Emurian, X. Hu, J. Wang, and A. G. Durham, 2000, *Computers in Human Behavior*, 16, 395–422. Copyright 2000 by Elsevier Science. Reprinted with permission.

right view shows the symbol identification interface that requires the learner to select the displayed symbol from among the 24 items presented in a list. The purpose of this interface is to sharpen a learner's discrimination of the formal properties among the symbols (Catania, 1998a). The sequence of selections followed the atomic units in the program, and this experience also contributes to associative development of the stream of symbols in the program.

Figure 34.8 presents the item interface (top view) and the serial stream interface (bottom view). The item interface teaches up to three items of code, and each of the three keyin fields is a separate knowledge unit. The purpose of this interface is to require accurate contextual response constructions by recall and to assess the understanding of an item's meaning by multiple-choice assessment. Each knowledge unit provides the occasion to observe the correct Java item prior to typing the item into the keyin field by recall. The explanation or meaning of the item is observed, and a multiple-choice test on the item's meaning must be passed to progress from one knowledge unit to the next. All entries for each unit must be completed correctly to progress to a succeeding unit. As a learner enters a unit correctly in the keyin field, the unit is displayed in formatted and cumulative sequence in the white text area. Upon completion of the three individual items, the serial stream interface requires entering those items as a single unit. This increases the size of the response required in the serial stream learn unit. If the learner is not able to enter the stream of three items correctly, the tutor branches back to the item interface, and that cycle continues until the serial stream is entered correctly. Then the next item interface is presented.

In this version of the tutor, there were 14 item interfaces, and the 36 total items learned were distributed across 10 lines of code. The item and serial stream interfaces reflected 86 learn units (36 items, 36 multiple-choice tests, and 14 serial streams).

The next interface advanced the response requirement to a row of items, and there were 10 rows. The progression in the complexity of the response and in the format for its entry reflects another application of *successive approximations* (Sulzer-Azaroff & Mayer, 1991, p. 394) to the final performance. Figure 34.9 presents a representation of the first row across the six iterations (i.e., passes) that were required through this interface. On each successive iteration, the graphic label prompts

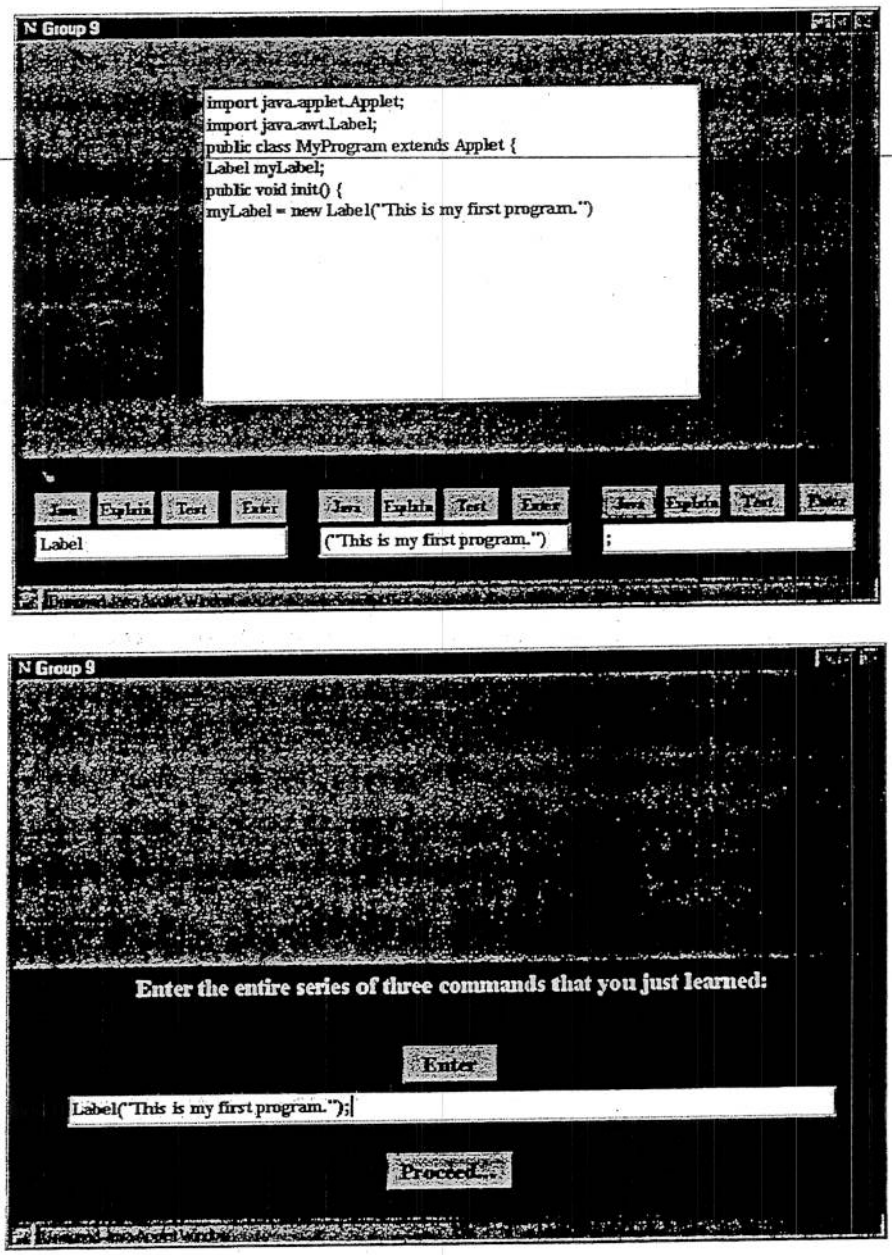

FIGURE 34.8. The item interface (top view) and the serial stream interface (bottom view). From "Learning Java: A Programmed Instruction Approach using Applets," by H. H. Emurian, X. Hu, J. Wang, and A. G. Durham, 2000, *Computers in Human Behavior*, 16, 395–422. Copyright 2000 by Elsevier Science. Reprinted with permission.

and background colors are gradually withdrawn as the stimulus control is transferred to the open area available in a text editor window. This shift in stimulus control reflects the process of *fading* (Sulzer-Azaroff & Mayer, 1991, p. 311). During the first iteration, the learner is required to pass a multiple-choice test on the objectives of each row immediately after entering the row correctly. If the learner is not able to enter the row, the tutor recycles through the item and serial stream interfaces that contain the code for that row. In this version of the tutor, there were 70 learn units in this interface, 20 on the first iteration (10 rows and 10 multiple-choice tests) and 10 on each of the following five iterations.

Figure 34.10 presents total number of learn units encountered by a test learner across the six successive iterations of this interface. The figure shows the minimum number of learn units available on each iteration, together with the number of units

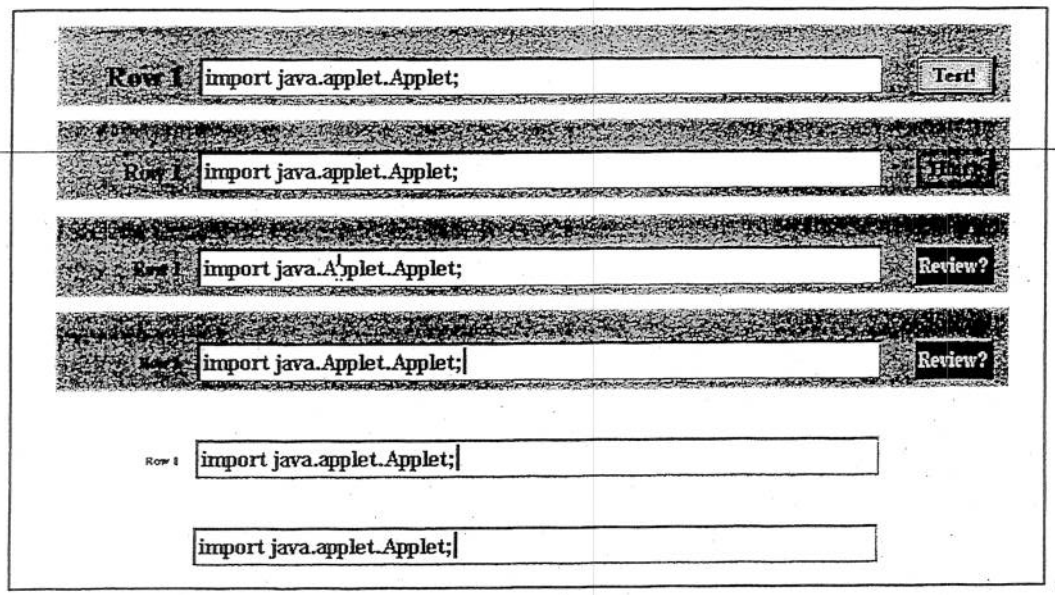

FIGURE 34.9. A representation of the first row across the six iterations (i.e., passes) that were required through this interface. From "Learning Java: A Programmed Instruction Approach using Applets," by H. H. Emurian, X. Hu, J. Wang, and A. G. Durham, 2000, *Computers in Human Behavior*, 16, 395–422. Copyright 2000 by Elsevier Science. Reprinted with permission.

encountered by the learner to complete each iteration. The total observed units reflect the additional learn units that were encountered whenever the learner selected a review of the item interfaces that supported the code in a given row. The figure shows a marked reduction in observed learn units between iterations 1 and 2, and this was followed by a gradual reduction in observed learn units across iterations 2 through 6. Even after six iterations, however, this learner required learn unit support in excess of the minimum presented during the sixth iteration.

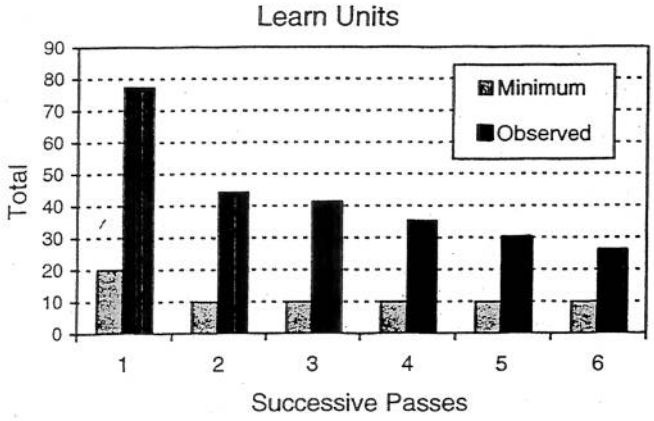

FIGURE 34.10. Total number of learn units encountered by a test learner across the six successive iterations of the row interface. From "Learning Java: A Programmed Instruction Approach using Applets," by H. H. Emurian, X. Hu, J. Wang, and A. G. Durham, 2000, *Computers in Human Behavior*, 16, 395–422. Copyright 2000 by Elsevier Science. Reprinted with permission.

Figure 34.11 presents two versions of accurate code displayed in the final interface, which emulated a text editor window. This interface advances the requirement for a correct response to be the stream of 36 Java items. These features reflect *successive approximation* and *fading*. The format for writing the code was relaxed for this interface, and the code was evaluated as a stream of characters. If the learner is not able to enter the code correctly, a review is available that recycles the learner back to the sixth iteration of the row interface. From there, the learner could cycle back to the item interfaces as needed. Once the code is entered correctly, the learner gains access to additional information that is required to compile the code and to run the Applet on the World Wide Web.

Classroom Applications

The programmed instruction tutoring system is used in the classroom as one component in the personalized system of instruction. During the first class period of a 14-period course in graphical user interfaces, based on the Java Abstract Windowing Toolkit, students are given a 2.5-hr interval in which to work on the tutor. Before engaging the tutor, each student completes a series of likert-type questionnaires (Critchfield, Tucker, & Vuchinich, 1998) that reflect the student's current confidence in being able to use each of the 24 atomic units to construct a Java program. The scale anchors are *1 = No confidence, a novice*

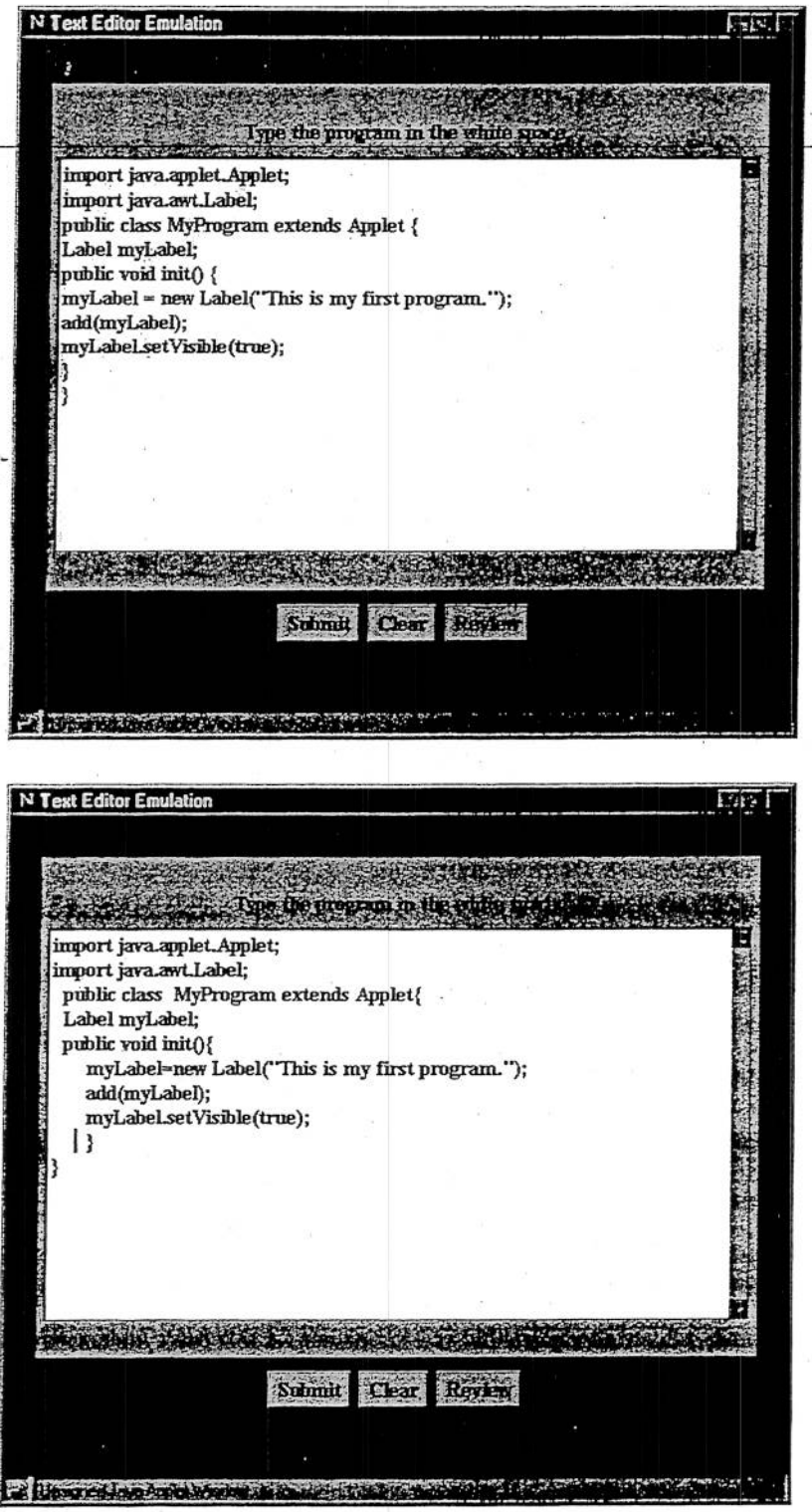

FIGURE 34.11. Two versions of accurate code displayed in the final interface, which emulated a text editor window. From "Learning Java: A Programmed Instruction Approach using Applets," by H. H. Emurian, X. Hu, J. Wang, and A. G. Durham, 2000, *Computers in Human Behavior*, 16, 395–422. Copyright 2000 by Elsevier Science. Reprinted with permission.

to *5 = High confidence, an expert.* Each student also attempts to write the code to display a text string as a Label object in the browser window. Self-report rating data are also collected on the following four scales: (1) experience with Java, where *1 = No experience, a novice* to *5 = High experience, an expert*; (2) overall impression of the tutor, where *1 = Negative*, to *5 = Positive*; (3) effectiveness of the tutor in learning Java, where *1 = Not effective* to *5 = Highly effective*; and (4) usability of the interfaces, where *1 = Difficult to use* to *5 = Easy to use*. These data are collected online with the assessment tools available in the WebCT™ instructional delivery platform. After the 2.5-hr interval or whenever the learner completes the tutor, the assessments for confidence and for writing the code are repeated.

During the second class period, the authors use a lecture format to review the code presented in the tutor. During this time, the students write the code in a text editor as it is being discussed. The compilation of the Java source code is then discussed, along with the HTML file that is used to start the Applet. The students are then encouraged to work further in a collaborative context with each other and with the course instructors and assistants. After all students have successfully run the Applet in the browser on the World Wide Web, the assessments for confidence and writing the code are repeated. These latter assessments are also administered again during the very last class of the semester. The combination of the programmed instruction tutoring system, the lectures and discussions, and the collaborative learning environment completes the personalized system of instruction for this introductory exercise.

Figure 34.12 presents self-report data on the four scales by a class of 12 graduate students (Emurian & Durham, 2001). Data are partitioned into two groups. Six students completed all interfaces in the tutor, and six students were working on the last interface when the 2.5-hr period expired. These self-report data show that inexperienced learners generally were positive in their work with the programmed instruction tutor interfaces. Although the measurements for the six students who did not complete the tutor are graphically less than the other students, significant differences were not supported between the two groups in their self-report ratings.

Figure 34.13 presents median self-report ratings of confidence for all 12 learners across the four assessment occasions. These data show that the group of learners showed a pronounced increase in confidence immediately after using the tutor. Confidence ratings thereafter were observed to increase over the remaining assessment occasions. The importance of these data are to be understood as a descriptive autoclitic verbal performance (Catania, 1998b, p. 407; Skinner, 1957, p. 313), which is based on a learner's ability to anticipate future behavior to use a symbol effectively. The performance indicates that a learner is able to describe his or her own competency. The impact of a supportive affective context in automated instructional systems has been increasingly recognized by educational scholars (e.g., Tennyson, 1999), and these descriptive autoclitic responses show the positive impact on learners of the programmed instruction tutoring system and the remaining course delivery as a personalized system of instruction.

Figure 34.14 presents cumulative total correct Java programs written by these 12 learners on the last three assessment

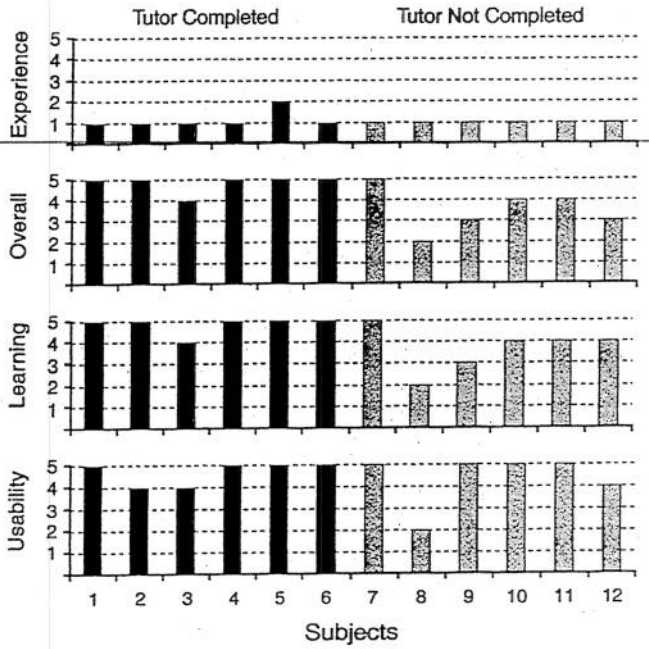

FIGURE 34.12. Self-report data on the four scales by a class of 12 graduate students. From *Managing Information Technology in a Global Environment* (pp. 155–160), by H. H. Emurian and A. G. Durham, 2001, Hershey, PA: Idea Group Publishing. Copyright 2001 by Idea Group Publishing. Reprinted with permission.

occasions. The instructions were to write the Java code that was taught by the tutor to display a text string, as a Label object, in a browser window. The code was written into a window on the WebCT assessment, and the accuracy of the code was determined by an error-free compilation. These data show that immediately after using the tutor, only two learners were able to write the code correctly in the assessment window. This outcome was observed even though six learners had entered the code correctly in the final interface of the tutor. After the students had run the Applet on the web, however, eight learners wrote the code correctly in the assessment window. But, during the final class period, only 1 of the 12 students was able to write the code correctly. This outcome presents a challenge for interpretation, especially in light of the reported beneficial effects of the tutoring system on learner confidence. An explanation for this latter finding requires consideration of several factors.

First, the overall course project objectives included only a single Applet subclass (... **My Program extends Applet** ...) along with many other custom and built-in Java classes that controlled a user's interactions with a web-based information system. After the first two sessions of instruction, the properties of the Applet class were not discussed in detail further. The strength of the originally learned response, then, may have declined over time, and such changes have been attributable to *passive decay with disuse* (Cowan, Saults, & Nugent, 2001) and to inadequate *overlearning* (Postman, 1962) in the verbal memory literature. Moreover, competing sources of influence

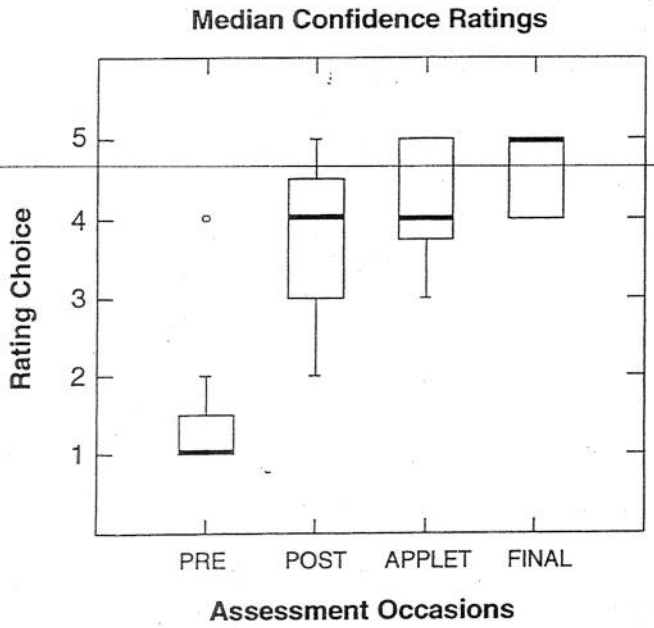

FIGURE 34.13. Box plots showing median confidence ratings for all 12 learners for the 24 distinct items that were used to compose the program. PRE = Pre-Tutor, POST = Post-Tutor, APPLET = Post-Applet, and FINAL = Final Class. From *Managing Information Technology in a Global Environment* (pp. 155–160), by H. H. Emurian and A. G. Durham, 2001, Hershey, PA: Idea Group Publishing. Copyright 2001 by Idea Group Publishing. Reprinted with permission.

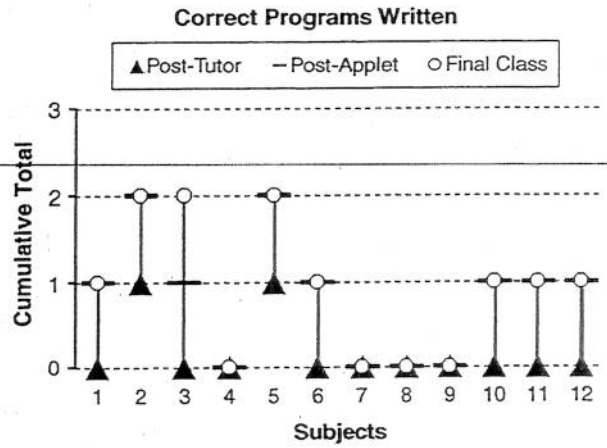

FIGURE 34.14. Cumulative total correct Java programs written by the 12 learners on the last three assessment occasions. From *Managing Information Technology in a Global Environment* (pp. 155–160), by H. H. Emurian and A. G. Durham, 2001, Hershey, PA: Idea Group Publishing. Copyright 2001 by Idea Group Publishing. Reprinted with permission.

from other subsequently learned Java items and class files suggest the involvement of symbol *interference* (Rehder, 2001; Underwood, 1957). The disuse interpretation is supported by the retention results with the UNIX tutor in which errors were observed on the second occasion of using the tutor, despite the more efficient learning that was observed. In the present situation, then, a more sensitive index of retention might have included performance data during reacquisition of the program taught by the Java-programmed instruction tutoring system.

Second, examination of the students' Java code that was submitted during the final assessment showed many instances of the students' attempts to use advanced, perhaps interfering, techniques to accomplish the requested outcome. Students attempted to write more sophisticated Java code at the end of the course than immediately after using the tutor, simply because they knew more object-oriented techniques by the end of the course. Still, they often produced programs that would not compile. Given the semester's experience in interpreting error messages, it is conceivable that a student might have discovered the source of an error and corrected it, if that opportunity had been a component of the final assessment.

These observations show that the effectiveness of repetition of a knowledge domain, presented within the context of a programmed instruction series of successive approximations to a terminal performance, requires documentation under more than a single occasion of rehearsal. Previous research, not directly related to programmed instruction, also emphasizes the importance of assessing retention, as an index of learning strength, of a newly acquired skill after a delay interval (Davis & Bostrom, 1993; Healy et al., 1995; Shute & Gawlick, 1995; Simon, Grover, Teng, & Whitcomb, 1996). Finally, studies using computer-based programmed instruction show retention effectiveness after a 5-week delay (Kritch et al., 1995) and after a 1-month delay (Kelly & Crosbie, 1997) following completion of a tutoring system.

Despite these observations, the personalized system of instruction (Keller, 1968), with the Java-programmed instruction tutor as a component, has been adopted by the authors to good advantage in the classroom because it generates a history of symbol use and confidence in each individual student. It combines both teaching and testing within a single conceptual framework: programmed instruction. It allows the needs of the individual student to be met because it frees the teacher from relying exclusively on traditional approaches, such as lecturing and writing on the board, to deliver technical information to a group of students. Most important, perhaps, the present tutoring system combines knowledge delivery with learning, assessment, and documentation of competence.

CONCLUSIONS

Learning occurs when there is a documented change in behavior that results from interactions with one's environment. To be informed, an instructional history is required. When a knowledge domain lends itself to enumeration of its components at atomic levels, the instructional history may be formulated by its representation within the context of programmed instruction tutoring systems, which emerged from the scholarly discipline of the experimental analysis of behavior. Such approaches to the design of an instructional system have the advantage of overseeing and managing the moment-by-moment process of

learning at the level of the individual student. Although the initial promise of the multimedia CLASS and PLATO systems has yet to be realized, perhaps, the application of computer-based information systems in this area offers the full range of computer hardware and software innovations to the instructional designer. It is understood that information technology applied in this area is but one tool to assist the development of human cognition (Mayer, 2000).

Identifying the conditions under which learning occurs continues to be the source of discourse and inquiry (Koubek, Benysh, & Tang, 1997), as does a consideration of how learning mechanisms should be implemented in training technologies (Swezey & Llaneras, 1997). How the parameters of these interactions should be structured to achieve specific educational objectives, in relationship to the status of the learner and the knowledge domain, has been the subject of inquiry for centuries, and the power function of learning is an effective model when applied to account for the acquisition and retention of a broad range of learning outcomes, to include intellectual skills (e.g., Anderson, 1987; Carlson & Yaure, 1990; Lane, 1987). The process of learning, then, is independent of a supportive instructional technology, which may be offered by computer-based systems or achieved by a learner who is skilled in self-regulation.

In that latter regard, Young (1996) proposed an instance of a student exhibiting a self-regulating style of learning that followed a self-directing and self-monitoring rehearsal strategy: "When I study for a test, I practice saying the important facts over and over to myself" (p. 18). It is, perhaps, a truism that effective learners already know how to regulate their learning environments and motivational status to achieve a criterion of mastery that is self-governed (Schunk, 2000; Skinner, 1968; Zimmerman, 1994). Effective self-management is itself a skill requiring a history whose components can be identified and made public in a scientific account of learning (Skinner, 1953). Programmed instruction approaches may be best suited for students who have not mastered the art of studying, and one important benefit of completing a programmed instruction tutoring system is that it teaches a learner how to acquire knowledge independent of the domain. The ultimate objective, then, of a programmed instruction tutoring system is to free the learner for advanced study undertaken with traditional forms of knowledge codification, such as a book (Brock, 1997).

Analyses of computer programming have been approached in the literature as a complex problem-solving activity (e.g., Campbell, Brown, & DiBello, 1992) and the acquisition of programming skills (e.g., Soloway, 1985; Van Merrienboer & Paas, 1990) For the most part, however, studies in the behavior of computer programming and program comprehension, ranging from early evaluations of conditional constructions (Sime, Green, & Guest, 1973) to recent simulations of memory representations by expert programmers (Altmann, 2001), provide observations and explanations of pre-existing human–computer interactions. Although identifying and classifying the strength of prior learning may cast light on the design effectiveness of alternative programming languages (Pane, Ratanamahatana, & Myers, 2001), educators seek to understand the user's history that leads to group membership as a *novice* or *expert* (Durham & Emurian, 1998b). Understanding that history suggests an instructional technology, and if the component steps can be identified, that summative history lends itself to implementation with computer-based programmed instruction.

A programmed instruction approach to learning a Java Applet was presented as an exemplar of this latter instructional technology that was implemented as a web-based tutoring system. The Applet program was operationalized as a serial stream, and a succession of learn units was programmed to concatenate a series of atomic units into a unitary serial stream of textual items. In the most elementary form, the learn unit required an item of Java code as the constructed response. In its terminal form, the learn unit required the serial stream of Java code as the constructed response. The utility of this approach for novice students to acquire competency in symbol manipulation was documented in the classroom where the programmed instruction tutoring system was applied within the context of a modified personalized system of instruction. The resulting repertoire, together with the positive affective responses by the learners, set the occasion for the continued mastery of advanced details and concepts of the Java programming language and the general properties of software design with the object-oriented model.

References

Alessi, S. M., & Trollip, S. R. (1991). *Computer-based instruction: Methods and development* (pp. 91–118). Englewood Cliffs, NJ: Prentice-Hall, Inc.

Altmann, E. M. (2001). Near-term memory in programming: A simulation-based analysis. *International Journal of Human-Computer Studies, 54*, 189–210.

Anderson, J. R. (1987). Skill acquisition: Compilation of weak-method problem solutions. *Psychological Review, 94*, 192–210.

Anderson, J. R. (1993). *Rules of the mind.* Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *Journal of Learning Science, 4*, 167–207.

Benyon, D., & Murray, D. (1993). Adaptive systems: From intelligent tutoring to autonomous agents. *Knowledge-Based Systems, 6*, 197–219.

Bijou, S. W. (1970). What psychology has to offer education—Now. *Journal of Applied Behavior Analysis, 3*, 65–71.

Bitzer, D. L., Braunfield, W. W., & Lichtenberger, W. W. (1962). PLATO II: A multiple-student, computer-controlled, automatic teaching device. In J. E. Coulson (Ed.), *Programmed learning and computer-based instruction* (pp. 205–216). New York: Wiley & Sons.

Bloom, B. S. (1984). The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher, 13*, 4–16.

Bostow, D. E., Kritch, K. M., & Tompkins, B. F. (1995).

Computers and pedagogy: Replacing telling with interactive computer-programmed instruction. *Behavior Research Methods, Instruments, & Computers, 27*, 297–300.

Bransford, J. D., Brown, A. L., & Cocking, R. R. (2000). *How people learn: Brain, mind, experience, and school* (Expanded Edition). Washington, DC: National Academy Press.

Bransford, J. D., Brown, A. L., & Rodney, R. C. (1999). *How people learn: Brain, mind, experience, and school.* Washington, DC: National Academy Press.

Brock, J. F. (1997). Computer-based instruction. In G. Salvendy (Ed.), *Handbook of human factors and ergonomics* (pp. 578–593). New York: Wiley.

Buck, G. H., & Hunka, S. M. (1992). Development of the IBM 1500 Computer-Assisted Instruction System. *IEEE Annals of the History of Computing, 17*, 19–31.

Buskist, W. B., Cush, D., & DeGrandpre, R. J. (1991). The life and times of PSI. *Journal of Behavioral Education, 1*, 215–234.

Calvin, A. D. (1969). *Programmed instruction: Bold new adventure.* Bloomington, IN: Indiana University Press.

Campbell, K. C., Brown, N. R., & DiBello, L. A. (1992). The programmer's burden: Developing expertise in programming. In R. R. Hoffman (Ed.), *The psychology of expertise* (pp. 269–294). New York: Springer-Verlag.

Carlson, R. A., & Yaure, R. G. (1990). Practice schedules and the use of component skills in problem solving. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 16*, 484–496.

Case, R. (1978). A developmentally based theory and technology of instruction. *Review of Educational Research, 48*, 439–463.

Case, R., & Bereiter, C. (1984). From behaviourism to cognitive behaviorism to cognitive development: Steps in the evolution of instructional design. *Instructional Science, 13*, 141–158.

Catania, A. C. (1998a). *Learning.* Upper Saddle River, NJ: Prentice-Hall, Inc.

Catania, A. C. (1998b). The taxonomy of verbal behavior. In K. A. Lattal & M. Perone (Eds.), *Handbook of research methods in human operant behavior* (pp. 405–433). New York: Plenum Press.

Coffin, R. J., & MacIntyre, P. D. (1999). Motivational influence on computer-related affective states. *Computers in Human Behavior, 15*, 549–569.

Cognition and Technology Group at Vanderbilt. (1996). Looking at technology in context: A framework for understanding technology and education research. In D. C. Berliner & R. C. Calfee (Eds.), *Handbook of educational psychology* (pp. 807–840). New York: Macmillan Publishing.

Comenius (1657). *The great didactic.* "education, history of" Encyclopædia Britannica [On-line]. Available: http://search.eb.com/bol/topic?eu=108334&sctn=5&pm=1 [Accessed April 5, 2001].

Corey, S. M. (1967). The nature of instruction. In P. C. Lange (Ed.), *Programmed instruction: The sixty-sixth yearbook of the National Society of the Study of Education* (pp. 5–27). Chicago, IL: The University of Chicago Press.

Coulson, J. E. (1962a). *Programmed learning and computer-based instruction.* New York: Wiley & Sons.

Coulson, J. E. (1962b). A computer-based laboratory for research and development in education. In J. E. Coulson (Ed.), *Programmed learning and computer-based instruction* (pp. 191–204). New York: Wiley & Sons.

Cowan, N., Saults, S., & Nugent L. (2001). The ravages of absolute and relative amounts of time on memory. In H. L. Roediger III, J. S. Nairne, I. Neath, & A. M. Surprenant (Eds.), *The nature of remembering* (pp. 315–330). Washington, DC: American Psychological Association.

Critchfield, T. S., Tucker, J. A., & Vuchinich, R. E. (1998). Self-report methods. In K. A. Lattal & M. Perone (Eds.), *Handbook of research methods in human operant behavior* (pp. 435–470). New York: Plenum Press.

Crosbie, J., & Kelly, G. (1993). A computer-based personalized system of instruction course in applied behavior analysis. *Behavior Research Methods, Instruments, & Computers, 25*, 366–370.

Crosbie, J., & Kelly, G. (1994). Effects of imposed postfeedback delays in programmed instruction. *Journal of Applied Behavior Analysis, 27*, 483–491.

Crowder, N. A. (1962). Intrinsic and extrinsic programming. In J. E. Coulson (Ed.), *Programmed learning and computer-based instruction* (pp. 58–66). New York: Wiley & Sons.

Dale, E. (1967). Historical setting of programmed instruction. In P. C. Lange (Ed.), *Programmed instruction: The Sixty-sixth yearbook of the National Society of the Study of Education* (pp. 28–54). Chicago, IL: The University of Chicago Press.

Davis, S. I., & Bostrom, R. P. (1993). Training end users: An experimental investigation of the roles of the computer interface and training methods. *Management Information Systems Quarterly, 17*, 61–85.

Day, R., & Payne, L. (1987, January). Computer managed instruction: An alternative teaching strategy. *Journal of Nursing Education, 26*, 30–36.

Deterline, W. A. (1962). *An introduction to programmed instruction.* Englewood Cliffs, NJ: Prentice-Hall, Inc.

Duffy, T. M., & Jonassen, D. H. (1992). *Constructivism and the technology of instruction: A conversation.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Durham, A. G., & Emurian, H. H. (1998a). Learning and retention with a menu and a command line interface. *Computers in Human Behavior, 14*, 597–620.

Durham, A. G., & Emurian, H. H. (1998b). Factors affecting skill acquisition and retention with a computer-based tutorial system. *Proceedings of the 4th World Congress on Expert Systems* (pp. 738–744). New York: Cognizant Communication Corporation.

Emurian, H. H. (2001, April–June). The consequences of e-Learning. *Information Resources Management Journal, 14*, 3–5.

Emurian, H. H., & Durham, A. G. (2001). A personalized system of instruction for teaching Java. In M. Khosrowpour (Ed.), *Managing information technology in a global environment* (pp. 155–160). Hershey, PA: Idea Group Publishing.

Emurian, H. H., Hu, X., Wang, J., & Durham, A. G. (2000). Learning Java: A programmed instruction approach using Applets. *Computers in Human Behavior, 16*, 395–422.

Ericsson, K. A., & Lehmann, A. C. (1996). Expert and exceptional performance: Evidence of maximal adaptation to task constraints. *Annual Review of Psychology, 37*, 273–305.

Ferster, C. B. (1967). Arbitrary and natural reinforcement. *The Psychological Record, 17*, 341–347.

Ferster, C. B., & Perrott, M. C. (1968). *Behavior principles.* New York: Appleton-Century-Crofts.

Gagne, R. M. (1968). *The conditions of learning.* New York: Holt, Rinehart, & Winston.

Green, E. J. (1967). *The learning process and programmed instruction.* New York: Holt, Rinehart, & Winston, Inc.

Greer, R. D., & McDonough, S. H. (1999). Is the learn unit a fundamental measure of pedagogy? *The Behavior Analyst, 22*, 5–16.

Healy, A. F., King, C. L., Clawson, D. M., Sinclair, G. P., Rickard, T. C., Crutcher, R. J., Ericsson, K. A., & Bourne, L. E. (1995). Optimizing the long-term retention of skills. In A. F. Healy & L. E. Bourne (Eds.), *Learning and memory of knowledge and skills: Durability and specificity* (pp. 1–65). Thousand Oaks, CA: SAGE Publications.

Holland, J. G. (1960). Teaching machines: An application of principles from the laboratory. *Journal of the Experimental Analysis of Behavior, 3*, 275–287.

Holland, J. G. (1967). A quantitative measure for programmed instruction. *American Educational Research Journal, 4*, 87–101.

Holland, J. G., & Skinner, B. F. (1961). *The analysis of behavior: A program for self-instruction*. New York: McGraw-Hill Co.

Jehng, J. J., & Chan, T. (1998). Designing computer support for collaborative visual learning in the domain of computer programming. *Computers in Human Behavior, 14*, 429–448.

Keller, F. S. (1968). Goodbye teacher. *Journal of Applied Behavior Analysis, 1*, 79–89.

Kelly, G., & Crosbie, J. (1997). Immediate and delayed effects of imposed postfeedback delays in computerized programmed instruction. *The Psychological Record, 47*, 687–698.

Kemp, F. D., & Holland, J. G. (1966). Blackout ratio and overt responses in programmed instruction: Resolution of disparate results. *Journal of Educational Psychology, 57*, 109–114.

Klein, P. S., & Darom, O. N. E. (2000). The use of computers in kindergarten, with or without adult mediation: Effects on children's cognitive performance and behavior. *Computers in Human Behavior, 16*, 591–608.

Koubek, R. J., Benysh, S. A. H., & Tang, E. (1997). Learning. In G. Salvendy (Ed.), *Handbook of human factors and ergonomics* (pp. 130–149). New York: Wiley.

Kritch, K. M., & Bostow, D. E. (1998). Degree of constructed-response interaction in computer-based programmed instruction. *Journal of Applied Behavior Analysis, 31*, 387–398.

Kritch, K. M., Bostow, D. E., & Dedrick, R. F. (1995). Level of interactivity of videodisc instruction on college students' recall of AIDS information. *Journal of Applied Behavior Analysis, 28*, 85–86.

Kulik, J. A., Cohen, P. A., & Ebeling, B. J. (1980). Effectiveness of programmed instruction in higher education: A meta-analysis of findings. *Educational Evaluation and Policy Analysis, 2*, 51–64.

Kulik, C. C., Kulik, J. A., & Bangert-Drowns, R. L. (1990). Effectiveness of mastery learning programs: A meta-analysis. *Review of Educational Research, 60*, 265–299.

Kulik, J. A., Kulik, C. C., & Cohen, P. A. (1979). A meta-analysis of outcome studies of Keller's personalized system of instruction. *American Psychologist, 34*, 307–318.

Lane, N. E. (1987). *Skill acquisition rates and patterns: Issues and training implications*. New York: Springer-Verlag.

Lange, P. C. (1967). *Programed instruction: The Sixty-sixth yearbook of the National Society for the Study of Education*. Chicago: The University of Chicago Press.

LeLouche, R. (1998). The successive contributions of computers to education: A survey. *European Journal of Engineering Education, 23*, 297–308.

Li, S., & Lewandowsky, S. (1995). Forward and backward recall: Different retrieval processes. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 21*, 837–847.

Margulies, S., & Eigen, L. D. (1962). *Applied programmed instruction*. New York: John Wiley & Sons, Inc.

Mayer, R. E. (2000). An interview with Richard E. Mayer: About technology. *Educational Psychology Review, 12*, 477–483.

Merrill, P. F., Hammons, K., Vincent, B. R., Reynolds, P. L., Christensen, L., & Tolman, M. N. (1996). *Computers in education* (pp. 65–86). Boston: Allyn & Bacon.

Pane, J. F., Ratanamahatana, C. A., & Myers, B. A. (2001). Studying the language and structure in non-programmers' solutions to programming problems. *International Journal of Human-Computer Studies, 54*, 237–264.

Peel, E. A. (1967). Programmed thinking. *Programmed Learning & Educational Technology, 4*, 151–157.

Postman, L. (1962). Retention as a function of degree of overlearning. *Science, 135*, 666–667.

Pressey, S. L. (1927). A machine for automatic teaching of drill material. *School and Society, 23*, 549–552.

Rabinowitz, M. (1993). *Cognitive science foundations of instruction*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Ray, R. D. (1995). A behavioral systems approach to adaptive computerized instructional design. *Behavior Research Methods, Instruments, & Computers, 27*, 293–296.

Rehder, B. (2001). Interference between cognitive skills. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 27*, 451–469.

Reigeluth, C. M., & Darwexeh, A. N. (1982). The elaboration theory's procedures for designing instruction: A conceptual approach. *Journal of Instructional Development, 5*, 22–32.

Roediger, H. L., & Guynn, M. J. (1996). Retrieval processes. In E. L. Bjork & R. A. Bjork (Eds.), *Memory* (pp. 197–236). New York: Academic Press.

Rolls, E. T. (2000). Memory systems in the brain. *Annual Review of Psychology, 41*, 599–631.

Ryan, S. D. (1999). A model for the motivation for IT retraining. *Information Resources Management Journal, 12*, 24–32.

Schunk, D. H. (2000). *Learning theories: An educational perspective* (pp. 355–401). Upper Saddle River, NJ: Prentice-Hall, Inc.

Scriven, M. (1969). The case for and use of programmed texts. In A. D. Calvin (Ed.), *Programmed Instruction: Bold New Adventure* (pp. 3–36). Bloomington: Indiana University Press.

Shute, V. J., & Gawlick, L. A. (1995). Practice effects of skill acquisition, learning outcome, retention, and sensitivity to relearning. *Human Factors, 37*, 781–803.

Shute, V. J., Gawlick, L. A., & Gluck, K. A. (1998). Effects of practice and learner control on short- and long-term gain and efficiency. *Human Factors, 40*, 296–310.

Shute, V. J., & Psotka, J. (1996). Intelligent tutoring systems: Past, present, and future. In D. Jonassen (Ed.), *Handbook of research on educational communications and technology* (pp. 570–600). New York: Macmillan Publishing.

Sidman, M. (1960). *Tactics of scientific research*. New York: Basic Books.

Sime, M. E., Green, T. R. G., & Guest, D. J. (1973). Psychological evaluation of two conditional constructions used in computer languages. *International Journal of Man-Machine Studies, 5*, 105–113.

Simon, J. S., Grover, V., Teng, J. T. C., & Whitcomb, K. (1996). The relationship of information system training methods and cognitive ability to end-user satisfaction, comprehension, and skill transfer: A longitudinal field study. *Information Systems Research, 7*, 466–490.

Skinner, B. F. (1968). *The technology of teaching*. Englewood Cliffs, NJ: Prentice-Hall, Inc.

Skinner, B. F. (1953). *Science and human behavior*. New York: The Free Press.

Skinner, B. F. (1954). The science of learning and the art of teaching. *Harvard Educational Review, 24*, 86–97.

Skinner, B. F. (1957). *Verbal behavior*. New York: Appleton-Century-Crofts.

Skinner, B. F. (1958). Teaching machines. *Science, 128*, 969–977.

Sohn, Y. W., & Doane, S. M. (1997). Cognitive constraints on computer problem-solving skills. *Journal of Experimental Psychology: Applied, 3*, 288–312.

Soloway, E. (1985). From problems to programs via plans: The content and structure of knowledge for introductory LISP programming. *Journal of Educational Computing Research, 1*, 157–172.

Sulzer-Azaroff, B., & Mayer, G. R. (1991). *Behavior analysis for lasting change*. Orlando, FL: Holt, Rinehart, & Winston, Inc.

Suppes, P., & Morningstar, M. (1969). Computer-assisted instruction. *Science, 166*, 343–350.

Swezey, R. W., & Llaneras, R. E. (1997). Models in training and instruction. In G. Salvendy (Ed.), *Handbook of human factors and ergonomics* (pp. 514–577). New York: Wiley.

Szabo, M., & Montgomerie, T. C. (1992). Two decades of research on computer-managed instruction. *Journal of Research on Computing in Education, 25,* 113–134.

Tennyson, R. D. (1999). Goals for automated instructional systems. *Journal of Structural Learning and Intelligent Systems, 13,* 215–226.

Tennyson, R. D., & Elmore, R. L. (1997). Learning theory foundations for instructional design. In R. D. Tennyson, F. Schott, N. M. Seel, & S. Dijkstra (Eds.), *Instructional design: International perspectives* (pp. 55–78). Mahwah, NJ: Lawrence Erlbaum Associates.

Tennyson, R. D., & Schott, F. (1997). Instructional design theory, research, and models. In R. D. Tennyson, F. Schott, N. M. Seel, & S. Dijkstra (Eds.), *Instructional design: International perspectives* (pp. 1–16). Mahwah, NJ: Lawrence Erlbaum Associates.

Tudor, R. M, & Bostow, D. E. (1991). Computer-programmed instruction: The relation of required interaction to practical application. *Journal of Applied Behavior Analysis, 24,* 361–368.

Underwood, B. J. (1957). Interference and forgetting. *Psychological Review, 64,* 49–60.

Van Merrienboer, J. J. G., & Paas, F. G. W. C. (1990). Automation and schema acquisition in learning elementary computer programming: Implications for the design of practice. *Computers in Human Behavior, 6,* 273–298.

Vargas, E. A., & Vargas, J. S. (1991). Programmed instruction: What it is and how to do it. *Journal of Behavioral Education, 1,* 235–251.

Wheeler, J. L., & Regian, J. W. (1999). The use of a cognitive tutoring system in the improvement of the abstract reasoning component of word problem solving. *Computers in Human Behavior, 15,* 243–254.

Young, J. D. (1996). The effect of self-regulated learning strategies on performance in learner controlled computer-based instruction. *Educational Technology Research and Development, 44,* 17–27.

Zimmerman, B. (1994). Dimensions of academic self-regulation: A conceptual framework for education. In D. H. Schunk & B. J. Zimmerman (Eds.), *Self-regulation of learning and performance* (pp. 3–21). Hillsdale, NJ: Erlbaum.