24. LEARNING *WITH* TECHNOLOGY: USING COMPUTERS AS COGNITIVE TOOLS

David H. Jonassen
PENNSYLVANIA STATE UNIVERSITY

Thomas C. Reeves
UNIVERSITY OF GEORGIA

24.1 INTRODUCTION

Human progress can be investigated in many ways. One insightful approach is to study the nature and quality of the tools people have discovered, invented, and refined over the centuries. The most common understanding of tools focuses on them as external implements, i.e., the levers, pulleys, and simple machines that have enabled physically weak human beings to change the course of mighty rivers, build giant edifices, and create ever-more complicated machines. A more theoretical perspective of tools recognizes that some tools are powerful without having a tangible physical substance, in the sense that a hammer does. Pea (1985) refers to these tools as *cognitive technologies,* and Salomon, Perkins, and Globerson (1991) call them *technologies of the mind.* In this chapter, we prefer the term *cognitive tools* (Kommers, Jonassen & Mayes, 1992) and elsewhere *mindtools* (Jonassen, 1996). Cognitive tools refer to technologies, tangible or intangible, that enhance the cognitive powers of human beings during thinking, problem solving, and learning. Written language, mathematical notation, and, most recently, the universal computer are examples of cognitive tools. This chapter focuses on computer-based cognitive tools, including common software applications and interactive learning environments, and their effects in the context of human learning.

Our emphasis on the uses and effects of computers and related technologies as cognitive tools is distinctly different from that of most of the other chapters in this handbook, in which technologies are primarily considered as forms of "media." Despite efforts to change the focus of the debate (cf. Jonassen, Campbell & Davidson, 1994), long-standing arguments about the relative effectiveness of media continue (cf. Clark, 1994; Kozma, 1994). Whether one sides with those who believe that media have little or no effects on learning or with those who promote its unique instructional effectiveness, such arguments are limited by narrow definitions of media as conveyors of information, communicators of knowledge, or tutors of students. We regard the "technology as instructional communications" perspective (see Chapter 4), although admittedly widespread throughout education and training, to be inherently flawed because it fails to recognize learners as active constructors of knowledge (Duffy & Jonassen, 1992; see Chapters 7 and 23).

Grounded in this limited perspective, most research studies reported in the other chapters in this handbook treat students as perceivers or recipients of knowledge encoded in various forms of instructional media. In essence, these studies and the technology applications investigated in them are about "educational communications," i.e., the deliberate and intentional act of communicating content to students, with the assumption that they will learn something "from" these communications (see Chapter 4). In educational communications, information or knowledge is encoded visually or verbally in the symbol systems enabled by various technologies. During the "instructional" process, students perceive the messages encoded in the media, e.g., in video, and occasionally "interact" with the technology, e.g., in computer-based instruction. Interaction is normally operationalized in terms of student input to the technology, some form of answer judging, and a response in the form of some message previously encoded in the media. Technologies as conveyors of information have been used for centuries to "teach" students, whereas interactive technologies began to be introduced early in the 20th century to "engage" students in the learning process (Cuban, 1986).

Educational communications and the technologies in which they are encoded are conceived, analyzed, and designed by educational specialists (often referred to as *educational* or *instructional technologists*). Historically, educational media have been developed by teams of educational technologists, including instructional designers,

media producers, and media managers, in collaboration with other types of specialists, e.g., subject-matter experts and teachers. These teams often employ systematic instructional design models (cf. Dick & Carey, 1990; Gagné, Briggs & Wager, 1987) to guide their efforts to analyze, develop, produce, and evaluate instruction. Design decisions made by these teams are purported to be informed by the kinds of educational communications and media research represented throughout this handbook, and some theorists even claim to be on the verge of automating the instructional design process based on existing learning theory and research (cf. Merrill, Li & Jones, 1990; Spector, Polson & Muraida, 1993).

24.2 COMPUTERS AS COGNITIVE TOOLS

By contrast, this chapter represents a departure from the central emphasis in this handbook on media and technology as vehicles for educational communications. Instead, we focus on the applications of technologies, primarily computers, as cognitive tools. This chapter is about computer-based cognitive tools and learning environments that have been adapted or developed to function as intellectual partners to enable and facilitate critical thinking and higher-order learning. Examples of cognitive tools include (but are not necessarily limited to):

- Databases
- Spreadsheets
- Semantic networks
- Expert systems
- Multimedia/hypermedia construction software
- Computer-based conferencing
- Collaborative knowledge construction environments
- Computer programming languages
- Microworlds

The cognitive tools perspective is distinctly different from traditional conceptions of instructional technologies. In cognitive tools, information is not encoded in predefined educational communications that are then used to transmit knowledge to students. With cognitive tools, the instructional design processes referred to above are eliminated. Instead of specialists such as instructional designers using technology to constrain students' learning processes through prescribed communications and interactions, the technologies are taken away from the specialists and given to learners to use as media for representing and expressing what they know. Learners themselves function as designers using technologies as tools for analyzing the world, accessing information, interpreting and organizing their personal knowledge, and representing what they know to others.

As important as it is to distinguish the "cognitive tools" perspective from the traditional educational media approach, it is also important to highlight differences between this conception of technology and earlier perspectives of using computers to support learning that have not been successful. Ever since Taylor (1980) presented his classic model of the roles of computers in education as "tutor, tool, and tutee," many educators and commercial entrepreneurs have predicted that computers would revolutionize education through one or more of these roles. In reality, none of these approaches has lived up to its promise.

In recent years, advocates of computer-based instruction and intelligent tutoring systems (ITS) who represent the computer-as-"tutor" perspective have begun to acknowledge the lack of impact they have had on mainstream education and training (cf. Lajoie & Derry, 1993; Shlechter, 1991). At least part of this failure stems from the overly restrictive perspective of students as perceivers or recipients of educational communications that characterizes the research in this field. Another factor contributing to the lack of success of ITS is that the technical difficulties inherent in building student models and facilitating humanlike communications have been greatly underestimated by proponents of the "tutor" model (see 19.5).

The computer-as-"tool" approach has also disappointed many of its proponents, although there have been some successes when tools have been embedded within innovative pedagogy such as a whole-language approach to literacy development (cf. Bruce & Rubin, 1993). In many cases, software tools such as word-processing, spreadsheet, database, and computer-aided design (CAD) programs have failed to improve teaching and learning significantly because they have been largely relegated to the service of a traditional "instructivist" pedagogy. Goodlad (1984) and others have described the teacher-directed, text- and workbook–dominated curriculum that has characterized educational practice for decades. Instead of being employed as cognitive tools to solve challenging problems, pursue personal learning goals, or accomplish authentic tasks, computer tools have often been regarded as objects for study themselves and subjected to the same deadly instructivist pedagogy that has stymied intellectual growth by most students in more traditional areas such as science, mathematics, and social studies. Consider, for example, computer-aided design (CAD) software, which has revolutionized professional practices and dramatically increased productivity in engineering, architecture, and other design fields. Industrial arts teachers (now called *technology educators*) have enthusiastically adopted CAD software into their classrooms and computers labs, but instead of engaging students in authentic tasks, they usually "teach" students the command sets for the software outside of any meaningful contexts. Not surprisingly, students end up failing to perceive the relevance and value of such programs within the design professions or their own lives. As pointed out by Salomon et al. (1991), "No important impact can be expected when the same old activity is carried out with a technology that makes it a bit faster or easier; the activity itself has to change" (p. 8).

The results of the "tutee" role for computers in education, despite the almost religious fervor with which it has been embraced in some circles (cf. Papert, 1980), have also been much less spectacular than promised (see 12.3.2.1).

According to the computer-as-"tutee" approach, students develop higher-order thinking skills and creativity by teaching the computer to perform tasks, e.g., draw a picture, through the use of "friendly" programming languages such as Logo (Papert, 1980) and microworlds such as Karel the Robot (Popyack, 1989). Studies aimed at investigating the effects of Logo (cf. Pea & Kurland, 1987) have failed to demonstrate the cognitive advantages promised by Papert and others. Defenders of the "tutee" approach would maintain that the implementations of Logo investigated in most studies were too brief and unfocused. To be sure, many applications of Logo and other microworlds described in the literature seem to lack the "mindful engagement" that Salomon and Globerson (1987) argue is necessary for learning. As shown in greater detail below, more intensive applications of Logo, wherein students are engaged in meaningful tasks over longer periods of time, have demonstrated more impressive cognitive effects (cf. Harel, 1991; Papert, 1993).

24.3 WHY COGNITIVE TOOLS?

The history of educational communications and technology includes numerous examples of failed innovations and unfulfilled promises. Cognitive tools could become yet another casualty in the difficult struggle to improve teaching and learning unless it has a strong foundation of theory and practical principles to support it. In the following section, we briefly describe constructivist learning theory and related principles (see also Chapter 7) for its implementation. Considered together, constructivism and its attendant principles constitute a strong rationale for using technology as cognitive tools.

24.3.1 Knowledge Construction, Not Reproduction

Learning theory is in the midst of a revolution. Constructivist learning theory is gradually gaining the same respect and attention long accorded to instructivist learning theories such as behaviorism (Duffy & Jonassen, 1992), though not without a struggle (cf. Phillips, 1995). Constructivism is concerned with the process of how we construct meaning and knowledge in the world as well as with the results of the constructive process. How we construct knowledge depends on what we already know, our previous experiences, how we have organized those experiences into knowledge structures such as schemata and mental models, and the beliefs that we use to interpret the objects and events we encounter in the world. Cognitive tools can help us as learners organize, restructure, and represent what we know.

Constructivists (cf. von Glaserfeld, 1989) maintain that we construct our own reality through interpreting our experiences in the world. From the constructivist perspective, the ultimate nature of reality, or whether it even exists, does not matter as much as our unique and shared constructions of reality. According to constructivism, the teacher cannot map his or her own interpretations of the world onto the learner, because they do not share a set of common experiences and interpretations. Reality (or at least what we know and understand of it) resides in the mind of each knower who interprets the external world according to his or her own experiences, beliefs, and knowledge. Learners are able to comprehend a variety of interpretations and to use them in arriving at their own unique interpretations of the world. The mind filters input from the world in making its interpretations, and therefore we each conceive of the external world somewhat differently.

Whereas instructivists emphasize the transmission of standardized interpretations of the world by teachers and the educational communications they employ, as well as standardized assessments to test the degree to which students' understandings match the accepted interpretations, constructivists are more interested in creating learning environments wherein learners use cognitive tools to help themselves construct their own knowledge representations. Cognitive tools and the goals, tasks, culture, resources, and human collaboration integral to their use enable learners to engage in active, mindful, and purposeful interpretation and reflection. In traditional instruction, *active* refers to stimulus, response, feedback, and reinforcement conditions that help students mirror accepted views of reality, whereas in constructivist learning environments, *active* learners participate and interact with the surrounding environment to create their own interpretations of reality.

24.3.2 Designers as Learners

Ironically, the people who seem to learn the most from the systematic instructional design of instructional materials are the designers themselves. Jonassen, Wilson, Wang, and Grabinger (1993) reported this discovery while developing expert systems advisors that were intended to supplant the thinking required by instructional designers. The process of articulating their knowledge about the process of instructional design forced them to reflect upon their knowledge in a new and meaningful way. Following the old adage that the surest way to learn about subject matter is to have to teach it, the process of designing and producing instructional materials as performed by designers of educational communications enables instructional designers to understand content much more deeply than the students whose thinking will be constrained and controlled by the very materials they are developing. It follows that empowering learners to design and produce their own knowledge representations and educational communications is a powerful learning experience.

24.3.3 Learners as Designers

Langer (1989) and others (cf. Salomon & Globerson, 1987) have reminded us of the importance of mindfulness in learning. Students learn and retain the most from thinking

in meaningful (mindful) ways. Some of the best thinking results when students try to represent what they know. Representing knowledge as a mindful task can be enabled by cognitive tools such as hypermedia construction software or electronic spreadsheets. Such cognitive tools require students to think in meaningful ways to use the application's capabilities and features to represent what they know. Just as electronics troubleshooters cannot work effectively without the use of tools such as probes and oscilloscopes, students cannot learn deeply or mindfully without access to cognitive tools that help them assemble and represent knowledge. In short, the real power of computers to improve education will only be realized when students actively use them as cognitive tools rather than passively perceive them as tutors or repositories of information.

24.3.4 Experiential and Reflective Thinking

Norman (1983) distinguishes between two forms of thinking: experiential and reflective. Experiential thinking evolves from our experiences in the world; it is reflexive and occurs automatically. We experience something in the world and react to it; e.g., we see a red light and brake the car. Reflective thinking, on the other hand, requires more careful deliberation. We encounter a complex situation, think about it, reflect on stored knowledge, make inferences about it, determine implications, and reason about it. Reflective thought is the careful, deliberate kind of thinking that helps us make sense of what we have experienced and supports our construction of what we know. For example, consider the reflective thought required by major decisions in life concerning career, family, and health. Reflective thinking often requires external support, including books, computers, or other people. Norman contends that computers support reflective thinking when they enable users to compose new knowledge by adding new representations, modifying old ones, and comparing the two. Cognitive tools should be readily accessible to learners to support reflective thinking within the context of learning.

24.3.5 The Effects of Learning *with* and *of* Technology

Salomon, Perkins, and Globerson (1991) make an important distinction between the effects of learning *with* and *of* technology:

> First, we distinguish between two kinds of cognitive effects: Effects *with* technology obtained during intellectual partnership with it, and the effects *of* it in terms of the transferable cognitive residue that this partnership leaves behind in the form of better mastery of skills and strategies (p. 2).

Cognitive tools are important in both respects. With respect to the "with" effects, we agree with Salomon et al. (1991) that "the cognitive effects with computer tools greatly depend on the mindful engagement of learners in the tasks afforded by these tools" (p. 2). We think that educators should empower learners with cognitive tools and assess their abilities in conjunction with the use of these tools.

Such a development will entail a new conception of ability as an "intellectual partnership" between the learner's mind and various cognitive tools. Although some might worry that this partnership makes learners too dependent on the technology to perform without it, we must recognize that many contemporary performances are meaningless without the technologies that enable them. Allowing students to demonstrate their learning in collaboration with cognitive tools may be attacked by certain authorities with heavy investments in the existing system, but we should remember that such attacks have occurred in the face of every innovation. For example, Plato criticized written language as a technology that would weaken human memory. Just as we would not assess the ability of an artist without allowing the use of brushes, paint, and other media, we should not assess contemporary intellectual abilities without the tools of contemporary intellectual practices, including books and computers (Salomon et al., 1991). Indeed, our very conception of knowledge must change. For example, Simon (1987) maintains that we should move from a conception of knowledge as possession of facts and figures to one of knowledge as the ability to retrieve information from databases and use it to solve problems.

Postmodernists and other critical theorists (see Chapters 9 and 10) worry about our eventual evolution into cyborgs (Yeaman, 1994). Although we do not share this concern, we recognize that there are many important intellectual abilities that should be performed and accessed without the aid of cognitive tools. This is where Salomon et al.'s (1991) delineation of the learning effects *of* technology become so important:

> Until intelligent technologies become as ubiquitous as pencil and paper—and we are not there yet by a long shot—how a person functions away from intelligent technologies must be considered. Moreover, even if computer technology became as ubiquitous as the pencil, students will still face an infinite number of problems to solve, new kinds of knowledge to mentally construct, and decisions to make, for which no intelligent technology would be available or accessible (p. 5).

Salomon et al. (1991) argue that the existing research, largely experimental, "has demonstrated more what transferable effects the partnership with computer tools and programs *can be made* to have than the effects it actually does have under more natural conditions of daily employment" (pp. 6, 7). Whether cognitive tools leave the "cognitive residue" that they are predicted to leave in practical educational settings is the focus of some of the research reported in this chapter.

24.3.6 Meaningful versus Easy Learning

One of the false promises of many previous instructional innovations has been to make learning fun and easy

(Cuban, 1986). Cognitive tools make no such promise, either for learners or teachers. Instead, cognitive tools and interactive learning environments activate complex cognitive learning strategies and critical thinking. These computer-based tools not only extend the mind, they have the potential also to reorganize mental functioning (Pea, 1985) and engage learners in high-level generative processing of information (Wittrock, 1974). In generative processing, deeper information processing results from activating appropriate mental models, using them to interpret new information, assimilating new information back into those models, reorganizing the models in light of the newly interpreted information, and using the newly aggrandized mental models to explain, interpret, or infer new knowledge (Norman, 1983). Knowledge acquisition and integration, according to these perspectives, is a constructive process involving "mindful" cognitive effort (Langer, 1989; Salomon & Globerson, 1987). When using cognitive tools, learners engage in knowledge construction rather than knowledge reproduction.

Cognitive tools actively engage learners in creating knowledge that reflects their comprehension and conceptualization of information and ideas rather than absorbing predetermined presentations of objective knowledge. Cognitive tools are learner controlled, not teacher controlled or technology driven. For example, when students construct databases, they are constructing their own conceptualization of the organization of the content domain. Cognitive tools are not designed to reduce information processing, that is, make a task easier, as has been the goal of instructional design as a field and many previous instructional innovations. Nor are they "fingertip" tools (Perkins, 1993) that learners use naturally, effortlessly, and effectively. Rather, cognitive tools are essential components of a learning environment in which learners are required to think harder about the subject-matter domain being studied or the task being undertaken and to generate thoughts that would be impossible without these tools.

As noted above, cognitive tools are reflection tools that amplify, extend, and even reorganize human mental powers to help learners construct their own realities and complete challenging tasks. However, the enormous potential of cognitive tools can only be realized within a constructivist framework for learning. Moreover, the nature and source of the task becomes paramount in such an environment. Past failures of "tool" approaches to using computers in education can be largely attributed to the relegation of the tools to traditional academic tasks set by teachers or the curriculum within the context of outmoded instructivist pedagogy. Cognitive tools are best used by students to represent knowledge and solve problems within the context of pursuing investigations that are relevant to their own lives. Those investigations are ideally elicited or supported by a constructivist learning environment (Duffy, Lowyck & Jonassen, 1993). Cognitive tools are less likely to be effective when used to support only teacher-controlled or curriculum-driven tasks.

24.3.7 (Un)intelligent Tools

Education communications too often try to do the thinking for learners, to act like tutors and guide learning. During the last decade, the most exalted form of educational communications systems have been called *intelligent tutoring systems* (ITS) (Chapter 19; Polson & Richardson, 1988). ITS possess some degree of "intelligence," usually in the form of expert and student models that are used to make instructional decisions about how much and what kind of instruction learners need. In the face of the disappointing results of ITS, some experts suggest that "the appropriate role for a computer is not that of a teacher/expert but rather that of a mind-extension 'cognitive tool'" (Derry & Lajoie, 1993, p. 5). Cognitive tools, as we conceive them, are *un*intelligent tools, relying on the learner to provide the intelligence, not the computer. This means that planning, decision making, and self-regulation are the responsibility of the learner, not the technology. Cognitive tools can serve as powerful catalysts for facilitating these skills, assuming that they are used in ways that promote reflection, discussion, and collaborative problem solving.

24.3.8 Distributed Cognitive Processing

Cognitive technologies may be provided by any medium that helps learners transcend the limitations of their minds, such as limits on memory, thinking, or problem solving (Pea, 1985). The most pervasive cognitive technology is language. Imagine trying to learn to do something complex without the use of language. Language amplifies the thinking of the learner. Computers may also function as cognitive tools for amplifying and reorganizing the ways that learners think.

Computer-based cognitive tools can function as intellectual partners that share the cognitive burden of carrying out tasks (Salomon, 1993). When learners use computers as partners, they off-load some of the unproductive memorizing tasks to the computer, allowing themselves to think more productively. Perkins (1993) claims that learning does not result from solitary, unsupported thinking by learners. Cognitive tools enable us to allocate to ourselves the responsibility for the cognitive processing we do best, while we allocate to the technology the processing that it does best.

Rather than focusing on microlevel decisions about message design and media presentation features on a computer screen, researchers should seek to reveal the nature of interactions and collaborations between the learner and the computer. Unfortunately, most of the research in our field has investigated how we can use the limited capabilities of the computer to present information and judge learner input (neither of which computers do well) while asking learners to memorize information and later recall it on tests (which computers do with far greater speed and accuracy than humans; see Chapter 39). The cognitive tools approach assumes that we will assign cognitive responsibility to the part of the learning system that does it best. The learner

should be responsible for recognizing and judging patterns of information and then organizing it, while the computer should perform calculations, store information, and retrieve it on the learner's command. When cognitive tools function as intellectual partners, the performance of the learner is enhanced, leaving some "cognitive residue" in the learner that may transfer in situations where the learner encounters the tool again or even to situations where the tool is inaccessible (Salomon, 1993).

24.3.9 Summary of the Foundations for Cognitive Tools Research

The following principles sum up the foundations for the research findings reviewed in the rest of this chapter:

- Cognitive tools will have their greatest effectiveness when they are applied within constructivist learning environments.
- Cognitive tools empower learners to design their own representations of knowledge rather than absorbing knowledge representations preconceived by others.
- Cognitive tools can be used to support the deep reflective thinking that is necessary for meaningful learning.
- As a form of cognitive technology, cognitive tools have two kinds of important cognitive effects, those that are *with* the technology in terms of intellectual partnerships and those that are *of* the technology in terms of cognitive residue that remains after the cognitive tools are used.
- Cognitive tools enable mindful, challenging learning, rather than the effortless learning promised but rarely realized by other instructional innovations.
- The source of the tasks or problems to which cognitive tools are applied should be learners, guided by teachers and other resources in the learning environment.
- Ideally, tasks or problems for the application of cognitive tools should be situated in realistic contexts, with results that are personally meaningful for learners.
- Cognitive tools do not contain preconceived intelligence in the sense that intelligent tutoring systems are claimed to possess, but they do enable intellectual partnerships in the form of distributed cognitive processing.

24.4 OVERVIEW OF THE CHAPTER

In the remainder of this chapter, we describe the use of different cognitive tools for engaging learners in critical thinking, knowledge representation, and problem solving. We begin with computer programming languages as cognitive tools because these languages, especially Logo, were among the earliest applications of the "cognitive tools" perspective and because more research has been done with programming languages than with other types of cognitive tools. We then describe research conducted with hypermedia/multimedia authoring systems, semantic networking, and expert systems, three cognitive tools that have attracted much attention in recent years. Lastly, we review the limited research conducted with databases and spreadsheets, popular software applications that, although creative teachers have adopted them as cognitive tools for years, have been the focus of relatively little research.

24.5 COMPUTER PROGRAMMING LANGUAGES AS COGNITIVE TOOLS

24.5.1 What Are Computer Programming Languages?

Computers are essentially high-speed calculators that are able only to accept and move around electronic signals. Their power and any intelligence that can be ascribed to them are implicit in the programs that operate them. Many different kinds of programs can be entered into computers to control what they do. At the lowest level, many of a computer's electronic components (the ROM and EPROM chips) have "hard-wired" programs that are encoded into the logic of the physical connections themselves. The electronic programs that actually drive the computer describe memory locations to which combinations of high and low voltage charges (zeroes and ones) should be moved or ways to manipulate the charges that correspond to larger numbers that represent higher-level information. Programmers use higher-level assembly language programs to activate these machine language programs. Assembly language programs are comprised of low-level commands to move information around the machine.

Programming most often refers to the use of higher-level, procedural languages, like BASIC and Pascal. These languages are the ones that are most often taught to students in schools. Computer programming languages consist of sets of keywords and commands that are interpreted or compiled by other assembly language programs through the programming language editors into machine code that ultimately "runs" the computer. Combinations of commands define programming structures. Procedural languages have only three major types of programming structures: list, repetition, and decision (selection) structures. *List structures* describe linear sequences of operations that are performed by the computer every time the list routine is invoked. *Repetition structures* (loops) are sets of operations that are repeated by the computer. The same set of operations may be repeated a specific number of times or until a certain numerical state exists, or while a certain numerical state exists. Repetition structures are often embedded within each other, so that loops of operations run inside other loops, which run inside still other loops. *Selection structures* describe the causal, decision-making operations in a program. These statements are typically written in IF-THEN-ELSE format; that is, if a specific conditions exists, do one sequence of operations. If another condition exists, do another sequence of operations. If neither condition exists (ELSE), do something else. Decisions can be combined to provide complex options. There

are many ways these structures can be written and combined in order to solve computational problems.

Selecting and sequencing programming commands in order to solve some computational problem is a very complex process. Taylor (1991) defines the process of computer programming in five steps: problem definition, algorithm design, code writing, debugging, and documentation. According to Pea and Kurland (1984), programming consists of similar subtasks, such as understanding the problem, designing and planning the program, coding the program, and comprehending and debugging the program.

For many years, people have theorized that learning to program is an activity that develops higher-order thinking skills (Taylor, 1980). The languages that have most often been taught to learners in American schools in hopes of developing reasoning and thinking skills are BASIC, Pascal, and LOGO. A microcomputer version of the artificial intelligence (AI) language, Prolog, is often taught in European schools.

24.5.1.1. BASIC. The development of BASIC (Beginner's All-purpose Symbolic Instruction Code) accompanied the rapid growth of computing in the 1960s. It was developed at Dartmouth University as a standard, introductory procedural programming language. Although very popular in the "computer literacy" movement of the 1980s, BASIC has been phased out of many educational programs in recent years. The primary complaint about BASIC is its inherently unstructured nature. BASIC code is written in the numerical sequence in which it will be executed. The flow of any BASIC program is controlled by GOTO statements that refer to the line number of the statement to be executed. Unless a careful and very structured approach to writing BASIC code is used, the order of operations can become very confused through unrestricted use of flow control statements. Newer versions of BASIC are inherently more structured and powerful.

24.5.1.2. Pascal. Unlike BASIC, Pascal is an inherently structured programming language that requires that programmers identify all of the variables, procedures, and functions that they intend to use in the program. The purpose of this requirement is to produce more organized, better-structured programs. Since its introduction in 1971, Pascal, named for the French mathematician, has become the language that is often taught first to computer science students. Pascal, like LOGO (described below), enables programmers to define subprograms, including functions and procedures, and then to call them whenever they are needed. This feature avoids having to repeat sections of the code within the program. Another advantage of Pascal is the flexibility of its control structures, which are used to define subtasks or program modules. Repetition structures, such as REPEAT-UNTIL and WHILE-SO, and selections structures, such as IF-THEN-ELSE and CASE, are straightforward and easy to use.

24.5.1.3. LOGO. LOGO is a simplified language created at the Massachusetts Institute of Technology (MIT) by Seymour Papert (1980) to engage children in the construction of microworlds. The part of the LOGO language that is used most often consists of geometric commands that are sent to a turtlelike object on the computer screen to draw objects on the screen. (A cybernetic turtle that moves along the floor and draws onto a piece of paper under the turtle is also sometimes used in classrooms with very young children.) Children use LOGO to "teach" the turtle to draw objects by writing Pascal-like procedures. Children learn to combine those procedures into larger procedures that draw more complex scenes containing those objects. The syntax of the language is simplified enough to allow learners to explore and experiment with creating scenes (called *microworlds*). The main ideas that are fostered by LOGO are procedures, nesting procedures, and recursion (having a procedure call itself).

In addition to turtle graphics, LOGO contains a set of list-processing commands that are equivalent to the AI language, LISP. Learners use these commands and the procedures acquired through turtle graphics to create poems or conversations with each other. Although LOGO is syntactically much easier than BASIC or Pascal, it still requires the understanding of very abstract concepts, such as variables, procedures, and recursion. A more recent development is the integration of LOGO, the programming language, with Lego, the toy building blocks. With LOGO-Lego, children build physical structures (e.g., a windmill) with Lego pieces and activate them with LOGO commands.

24.5.1.4. Prolog. Prolog (PROgramming in LOGic) was developed as an artificial-intelligence language for solving problems that involve objects and their relationships stated in terms of declarative logic. That is why it is so often used to write programs representing human knowledge structures. It is interactive and conversational. Programming in Prolog consists of declaring facts about objects and their relationships, defining rules about those objects and relationships (like inheritance), and asking questions about those objects and relationships.

Micro-Prolog (Prolog for microcomputers) programs are made up of sentences that state objects and their relationships. These objects and relationships are added to the program individually. Prolog allows the user to ask true-false and search questions about the database that has been developed. The power of Prolog is afforded by its use of conditional, rule-based logic. Using simple building blocks, Prolog can be used to develop complex databases of information. What makes Prolog especially useful as a cognitive tool is its focus on knowledge representation.

24.5.1.5. Other Languages. Other procedural languages, like C and its derivatives, and AI languages like LISP, are also taught to students, though certainly not with the degree of regularity of those described above. The trends in programming include a diminished interest in procedural languages like BASIC and Pascal, and a translation of those and other procedural languages into "object-oriented programming systems" (OOPS). Rather than procedures and functions as sequences of actions, in OOPS, these are treated as reusable objects that can be combined like building blocks to construct a program. The program or the

user sends messages to these objects. These objects respond according to the message sent. The building block approach is especially important in defining screen objects like scroll bars, windows, buttons, icons, and menus in window-type environments that comprise the user interface to the program. So when the user points and clicks at an icon, the icon object responds depending on its location and program. Languages like Smalltalk were originally designed as OOPS environments; however, object-oriented versions of procedural languages like BASIC (Microsoft's Visual BASIC), Pascal (Borland's Delphi), and C (Borland's C++) are preferred by most programmers today. OOPS languages can help to promote critical thinking as well as programming efficiency because of their inherent structure and more clearly defined interfaces and usage declarations. OOPS may encourage more effective collaboration in defining interfaces, and the skills students learn may be more marketable since most businesses are using object-oriented versions of popular programming languages. Probably the most commonly used object-oriented language in schools is HyperTalk, the scripting language that is used with HyperCard, though object-oriented versions of most computer languages are now available.

24.5.2 How Are Computer Programming Languages Used as Cognitive Tools?

Proponents of programming have argued that learning to program requires that learners think in an organized, systematic way about the problems they are attempting to solve, and that the thinking skills acquired while learning to program transfer to other nonprogramming problem situations. Before considering the cognitive effects of programming, we briefly review research on the cognitive requirements of learning to program.

Programming is a complex task. It involves many aspects of problem solving, such as problem decomposition, selecting appropriate information, assigning variables, identifying plausible solutions, applying programming structures, debugging code, and so on. McCoy (1990) showed that learning to program requires five skills: general strategy, planning, logical thinking, variables, and debugging.

Complex skills like these require different forms of intelligence. Cafolla (1987–88) found that the strongest predictors of learning to program were verbal reasoning, level of cognitive development, and mathematical reasoning. Additionally, analogical reasoning ability is strongly related to the ability to write subprocedures among high school students learning to program in LOGO (Clement, Kurland, Mawby & Pea, 1986). Computer programming is also dependent on various cognitive controls and styles of learners (Jonassen & Grabowski, 1993). Students who are field independent perform better in computer programming classes than field-dependent learners; i.e., field-independent students are more analytical thinkers. In fact, field independence, logical reasoning, and direction-following skills were found to be highly correlated with programming skills among college students (Foreman, 1988).

Computer programming is also subject to developmental differences among learners. Fourth-graders were far less able than older students to understand and use variables because of the level of abstraction, the dynamic nature of the values of variables, the degree of complexity in using variables, and the level of reasoning required (Nachmias, 1986). This means that computer programming may be introduced at the upper elementary level, but that abstract concepts in programming must wait until later. Even college students experience difficulties in learning to program. Among undergraduates, Fischer (1986) found that programming skill in a course was highly correlated to formal operational reasoning (from Piaget), especially the classification of abstract concepts, control structures, and top-down programming. Formal operational reasoning ability is also essential for the use of some of the other cognitive tools, especially expert systems.

Computer programming has long been associated with mathematics. It has often been assumed that skilled mathematicians make competent programmers, because the same kinds of logical reasoning are required. This is partially true; however, programming computers is more directly related to analytical reasoning. Chin and Zecker (1985) found that programming ability was not, as expected, related to math ability, but rather internal locus of control was a much better predictor. On the other hand, Nowaczyk (1983) found that mathematics and English course performance, previous computer experience, and logic and algebraic word problem performance were significantly correlated to programming performance among college students. Being able to break down problems and search and select relevant information and solutions to the problem are most important for learners. Many people, including scientists, engineers, and mathematicians, are analytic, while many people are not. Most people can learn to function more analytically, but it is not easy. Programming is difficult because it requires thinking that learners are not often called on to perform.

24.5.3 What Learning Outcomes Result from Using Computer Programming Languages as Cognitive Tools?

Most of the educational research on computer programming has assessed how much the logical reasoning required to program computers generalizes or transfers to other nonprogramming problems. The assumptions of most of this research is that the analytic thinking required to program will naturally make learners better problem solvers in other settings. Learning to program has been shown to have a variety of effects on learners' thinking in different settings, although the research findings are inconsistent. Ahmed (1992) reviewed 21 studies and found that half of the studies showed some positive effects of learning to program;

half did not. Much of this research focuses on the cognitive residue from learning to program. Some studies found positive correlations. Liao and Bright (1991) conducted a meta-analysis of research on the effects of computer programming on cognitive outcomes and found that the large majority of studies concluded that students who learned to program scored higher on various cognitive tests than those who did not, although the differences were not large. For example, students who learned six weeks of BASIC performed better on mathematical-thinking skills, including programming ability, generalization, and understanding variables, than the control group (Oprea, 1988). This was especially true for shorter programming courses rather than longer ones and for learning LOGO rather than BASIC or Pascal.

Most of the cognitive outcomes research was conducted on learning to use LOGO for building microworlds. While LOGO was designed as an experimental tool for building microworlds, it also uses a procedural programming language. The cognitive residue from these experiences varies. Clements (1985) conducted a thorough review of LOGO research and concluded that almost all children can learn to program in LOGO. LOGO is especially effective in encouraging prosocial behavior, positive self-image, and positive attitudes toward learning. However, the cognitive gains from LOGO apply mainly to LOGO-related learning. Clements also found that programming does facilitate some problem-solving behaviors, and LOGO may facilitate the development of some cognitive skills such as classifying, seriating, and conserving. Clements and Gullo (1984) found that learning to program in LOGO improved 6-year-olds' reflectivity, divergent thinking, and metacognitive ability, when compared to a group receiving computer-assisted instruction. After one year of LOGO experience, fifth- and sixth-graders performed better on LOGO-related problem solving, general problem solving, and the mental rotation of geometric figures, a spatial reasoning ability (Miller, Kelly & Kelly, 1988). However, Pea and Kurland (1984) found that LOGO programmers were no better at planning skills than control group students, and no differences in the ability to visualize and draw designs resulted from learning to program among fourth- and fifth-graders (Williamson & Ginther, 1992). Kurland, Pea, Clement, and Mawby (1986) found that after studying programming for a year, LOGO programming experience did not transfer to other domains with similar properties. Jansson (1987) conducted three studies that showed no benefit of learning to program on conditional reasoning tasks.

The equivocal results of learning to use LOGO is partially a result of the teaching methods used with LOGO. Although it was intended to be used as an experiential, discovery-learning tool, too often LOGO was taught by direct teaching. Swan and Black (1990) found that explicitly teaching problem-solving strategies and applying them to solve problems in LOGO was more effective than providing only LOGO programming practice, teaching the strategies with concrete manipulables, or traditional problem-solving instruction. That is, learning to program is facilitated by practice in programming. But programming was not the primary purpose of LOGO. Harel (1991) maintains that one reason for the failure of LOGO skills to transfer to other domains is that teachers have often treated "LOGO as an object of knowledge in itself, rather than as a tool for acquiring other learning" (p. 37). Harel also complains that in most of the negative studies of the effects of learning LOGO, "children did not have time to learn LOGO in any depth" (p. 84). However, even if the sustained study of programming can be demonstrated to improve the ability of students to perform on some critical-thinking tests, the gains may not be worth the time and effort required to learn to program.

Harel's (1991) Instructional Software Design Project (ISDP) represents a unique effort to use programming as a cognitive tool within a software design context. Harel claims that the ISDP combines Papert's "constructionist" theory (1993) with Perkins "knowledge as design" pedagogy (1986). In the ISDP research study conducted by Harel for her doctoral dissertation, 17 fourth-grade students used LOGO for a semester to create software products that were intended to teach fractions to third-grade students. Her study combined quantitative, qualitative, and comparative research methods to investigate the effects of this "learners as designers" approach.

Harel (1991) reports that the fourth-grade students spent an average of 70 hours working on their software design projects. While the particular nature of the software projects was left open, there were two important requirements for students in the program: (1) writing in a "Designer's Notebook" every day and (2) attending periodic "Focus Sessions" about software design, LOGO programming, and fractions. A teacher and the researcher were available at all times to help the students with their design efforts. Although each of the students produced a separate software product, collaboration among the students was encouraged.

Harel compared the differences in LOGO skills and fractions knowledge between the 17 students in the ISDP and 34 other students in two classes who were studying LOGO and fractions via "a traditional teaching method" (p. 263). No significant differences were found in pretests among the three classes. Harel reports that "In general, the 17 children of the experimental class did better than the other 34 children on all posttests (fractions and LOGO)" (p. 272). Although not all differences were statistically significant, the general trend was quite positive in terms of specific learning outcomes as measured by multiple measures, including paper-and-pencil tests, computer exercises, videotaped observations, and interviews.

The major part of Harel's (1991) study is a detailed description of the activities and metacognition of one student, "Debbie," over the 4-month period of the project. Harel wrote that her detailed analysis of Debbie's work as well as her observations of other students indicated that "Throughout ISDP, the students were constantly involved in metacognitive acts: learning by explaining, creating, and discussing knowledge representations, finding design

strategies, and reflecting on all of the above" (p. 359). In addition to positive cognitive effects in terms of metacognition, Harel concluded that the ISDP students acquired enhanced cognitive flexibility, better control over their problem solving, and greater confidence in their thinking abilities. She notes, however, that the study did not include any direct measures of thinking skills, but her own interpretations of the students' metacognition and problem-solving processes were based on observations and analysis of documentation such as their Designer's Notebooks.

Thinking is always somewhat dependent on the nature of the problem or the content. Programming clearly requires learners to think deeply. However, this deep thinking does not necessarily transfer to other content or problems as much as educational experiences that are focused on content-dependent problem-solving skills. The transfer of programming logic could probably be enhanced by direct instruction that models how to apply programming skills to other problems rather than teaching the language and only later applying it to solving problems. Harel's (1991) study provides an exemplary approach to the integration of learning programming into a design and problem-solving environment. Kafai's (1995, 1996) studies of students using LOGO to create games for other students is a creative continuation of the research carried out by Harel. Interestingly, Kafai found that the students designing games using LOGO did not perform as well as a control group of students using LOGO to design instruction in the manner used in Harel's ISDP.

In light of the results described above, a question rises: Why have the results of learning to program been so inconsistent and generally disappointing to the advocates of this approach? The answer may be found in both the nature of the cognitive tools themselves and the different approaches to applying them. Some cognitive tools such as databases, spreadsheets, and even hypermedia/multimedia authoring systems described later in this chapter share a common set of attributes (Jonassen, 1996). They are readily available, generic applications; they are affordable; they are used to represent knowledge in content domains; they are applicable across different subject domains; they engage critical thinking in learners; they facilitate transfer of learning; they are simple, powerful formalisms; and they are reasonably easy to learn.

This latter criterion is the most problematic for computer programming. Programming in most computer languages requires learning as many as 100 different commands, knowing when and how to embed those commands into programming structures, and, most problematic and especially time consuming, a lot of syntax that constrains both of these. After a semester-long Pascal course in high school, students made errors using virtually every Pascal construction (Sleeman, Putnam, Baxter & Kuspa, 1986). Punctuation, spaces, order of operations, and a host of other syntactical requirements add hundreds and even thousands of rules that must be learned and faithfully used before programs run without error. Unlike traditional human languages, where the meaning of speech or writing can usually be determined regardless of misspellings, placement, c usage errors, computers are unforgiving. A single error ca prevent a program from running, despite the fact that man beginning programming students believe that computer have the reasoning power of human beings in comprehend ing language (Sleeman et al., 1986).

The primary conclusion of programming research is tha the cognitive overhead (the amount of mental effor required to use programming languages) mitigates the abil ity of the learner to use computer programming as an easy and effective means for solving problems or representing what the learner knows, which is the goal of using cognitive tools in the first place. After 2 years of programming instruction, many students have only rudimentary understanding of programming ideas (Kurland, Pea, Clement & Mawby, 1986). Until programming skills become automated (which requires years of experience to occur), more effort is required to program the computer than to represent the knowledge or solve the problem. There is little doubt, however, that computer programming is among the most flexible and powerful of the cognitive tools for those who are skilled programmers. It is the requirement of these complex skills that calls into question the utility of computer programming as a cognitive tool.

This argument against programming as a robust cognitive tool is being mitigated by newer computer-programming environments, like Think Pascal and Think C, that have syntax error detection and correction routines built in. These routines identify syntax errors when they are made, thereby reducing the cognitive load and responsibilities on the learner. They also automatically format the code by indenting where necessary. Newer programming languages, such as Visual BASIC, Delphi, and Visual C, provide even more sophisticated code-generating routines, so that the programmer needs only to identify the program structure and the variables, and the environment is able to generate the necessary programming code. These languages are especially effective for producing user interfaces by automating the creation dialog boxes, windows, and so on. These types of environments represent major improvements in simplifying the programming process, allowing the programmer to act as a designer who focuses on the problem-solving task more than on writing code. However, to what extent does this replacement affect the thinking that the programmer is required to do? These environments definitely enhance code generation, but do they support critical thinking? Additional research must be done before we can answer this question.

In addition to differences in the nature of various cognitive tools, the approach to using them obviously has a major impact on their effectiveness. When cognitive tools become objects of study in and of themselves, as seems to be the case in many studies of the effects of programming, they cannot be expected to have major effects on higher-order thinking skills. However, where these tools are applied to meaningful and personally rewarding tasks, they may have much more impressive results. Despite the cogni-

tive demands of learning programming, there may be nothing inherently wrong in the use of programming as cognitive tools if the context in which programming is learned captures the attention, intrinsic motivation, and commitment of learners.

24.6 HYPERMEDIA/MULTIMEDIA AUTHORING SYSTEMS AS COGNITIVE TOOLS

24.6.1 What Are Hypermedia and Multimedia?

The concept of hypermedia is founded on the earlier notion of hypertext, i.e., the nonsequential, nonlinear method for organizing and displaying text (Jonassen, 1989). It was designed to enable the reader to access information from a text in ways that are more meaningful for the reader, based on the assumption that the organization that the reader imposes on a text is more meaningful (to the reader) than that of the author (Nelson, 1980). Hypermedia is an extension of hypertext that integrates graphics, animation, audio, and video with text.

The most pervasive characteristic of hypermedia is the node that consists of chunks or fragments of text or other media. The most common metaphor for a node is a card, as in HyperCard (Apple Computer, 1984). Nodes are the basic unit of information storage, and may consist of a page of text, a graphic, a sound bite, a video clip, or even an entire document. While studying a hypermedia knowledge base, learners can access any node, depending on their interests or needs. In many hypermedia systems, nodes can be amended or modified by the learner. The learner may add to, delete, or change the information in a node or create his or her own unique nodes of information. In short, a hypertext can be a dynamic knowledge base that continues to grow, representing new and different points of view.

The organization of a hypermedia knowledge base, that is, the interrelationships between the nodes, is defined by the links that interconnect the nodes. Links in hypermedia systems are typically associative; that is, they describe associations between the nodes they connect. While looking at a node, the user's attention may be drawn to a link (usually identified by buttons or hot spots on the screen). If the user activates the link by clicking on a button or hot spot with a mouse or other pointing device or pressing an associated keyboard key, the user will be linked to another node of information. Having arrived at the new node related to the previous link, the user may wish to return to the node from which he or she came or to go to yet another node. The links in hypermedia transport the user through the information space to the nodes that are selected, enabling the user to move through the knowledge base. The node structure and the link structure form a network of ideas in the knowledge base—structures and networks that may be very rich.

Multimedia are just that: multiple media or the integration of more than one medium into some form of communication. Most often, multimedia refers to the integration of media such as text, sound, graphics, animation, video, imaging, and spatial modeling into a computer system (von Wodtke, 1993). Multimedia in other forms, such as slide-tape presentations and interactive video, have been available for a long time. The term has gained popularity recently with the advent of high-resolution monitors, sound and video compression cards, increased random-access memory, and larger storage media for personal computers. Employing relatively inexpensive desktop computers, users are now able to capture sounds and video, manipulate audio and images to achieve special effects, synthesize audio and video, create sophisticated graphics including animation, and integrate them all into a single multimedia presentation. Individuals with very little experience are becoming their own multimedia artists, producers, and publishers.

Multimedia presentations are engaging because they are multimodal. In other words, multimedia can stimulate more than one sense at a time, and in doing so may be more attention getting and attention holding. Futurists and industry leaders, as well as many educators, often promote the idea that multimodal access is essential when teaching today's video generation (cf. Perelman, 1992). Not surprisingly, others have begun to question the benefits of multimedia and related technologies (cf. Stoll, 1995).

Nielsen (1990) attempts to clarify the difference between multimedia and hypermedia:

> . . . even though many hypertext systems are in fact hypermedia systems and include many multimedia effects, the fact that a system is multimedia based does not make it hypertext. The mixture of text and graphics is not enough in itself. Many multimedia systems are based mostly on displaying various films clips to a passive user who does not get to navigate an information space. Only when users interactively take control of a set of dynamic links among units of information does a system get to be a hypertext (p. 10).

Although hypermedia affords many options to users, some very significant problems have plagued hypermedia users. The most commonly acknowledged problem is navigation. Hypermedia documents often contain thousands of nodes, each with multiple links to other nodes. It is easy for users to get lost in that morass of information. Users become disoriented, unaware of the route they took, or how to find their way out of the hypertext or to another topic of interest.

A related problem is how and where do users access information in the hypertext. Most hypertexts provide an array of options to the user but typically fail to provide suggestions about where the user should begin. The user's initial access to the hypertext may greatly affect the user's understanding of the information contained in it. Another problem is a lack of orientation as to how much of the hypertext the user has accessed and how much remains to be revealed.

Perhaps the greatest concern related to using multimedia and hypermedia to facilitate learning is the problem learners face in integrating the information acquired in a hypertext into their own knowledge structures (Jonassen

Beissner & Yacci, 1993). As learners navigate through the hypertext, how can they best relate new information to what they already know and what they have learned from the hypermedia? How do learners develop their own knowledge structures and use them to accommodate the new information? One aspect seems clear, i.e., the creation of new knowledge structures must be applied within a personally meaningful context to a relevant problem or task. Learners don't automatically create new knowledge structures simply by browsing hypermedia resources (Jonassen & Wang, 1993).

24.6.2 How Are Hypermedia/Multimedia Authoring Systems Used as Cognitive Tools?

A solution to the problems of navigating and integrating information in hypermedia is not to think of hypermedia as a form of instruction to learn *from*, but rather to look at hypermedia as a tool for constructing and learning *with* (Jonassen, Myers & McKillop, 1996). In other words, we advocate the use of hypermedia as a cognitive tool. Learners may create multimedia databases that reflect their own perspectives on, or understanding of, ideas. Or learners may collaborate with other learners to develop a classroom or school hypermedia knowledge base. We contend that students are likely to learn more by constructing hypermedia instructional materials than by studying hypermedia created by others. Of course, hypermedia created by others (such as World Wide Web sites) can provide excellent resources for students in the process of creating their own hypermedia.

Hypermedia and multimedia construction is predicated on the idea of knowledge as design (Perkins, 1986), which refocuses the educational process away from one of knowledge as information and the teacher as transmitter of that knowledge to one of teachers and students as collaborators in the knowledge construction process. One way to promote this design process is to place learners in the role of instructional software designers (Harel, 1991). Rather than reading textbooks and solving workbook problems, students must define and constantly refine the nature of a problem they have identified, reconstruct their knowledge to solve that problem, and represent their solution in hypermedia (Lehrer, 1993).

Designing multimedia presentations is a complex process that engages many skills in learners. Carver, Lehrer, Connell, and Ericksen (1992) suggest some of the major thinking skills that learners need to use as designers:

Project Management Skills
- Creating a timeline for the completion of the project
- Allocating resources and time to different parts of the project
- Assigning roles to team members

Research Skills
- Determining the nature of the problem and how research should be organized
- Posing thoughtful questions about structure, models, cases, values, and roles
- Searching for information using text, electronic, and pictorial information sources
- Developing new information with interviews, questionnaires, and other survey methods
- Analyzing and interpreting all the information collected to identify and interpret patterns

Organization and Representation Skills
- Deciding how to segment and sequence information to make it understandable
- Deciding how information will be represented (text, pictures, movies, audio, etc.)
- Deciding how the information will be organized (hierarchy, sequence) and how it will be linked

Presentation Skills
- Mapping the design onto the presentation and implementing the ideas in multimedia
- Attracting and maintaining the interests of the intended audiences

Reflection Skills
- Evaluating the program and the process used to create it
- Revising the design of the program using feedback

While the engagement of each of these skills has yet to be empirically validated in a series of studies, there is no doubt that constructing hypermedia and multimedia programs represents a complex combination of skills. Verifying these skills and their effects on thinking should become a major research agenda in the years to come.

24.6.3 What Research Supports the Use of Hypermedia/Multimedia Authoring Systems as Cognitive Tools?

One of the principles stated above for the implementation of cognitive tools is: "Ideally, tasks or problems for the application of cognitive tools should be situated in realistic contexts with results that are personally meaningful for learners." Beichner (1994) reports on a project where these conditions were met in a unique way. The subjects in this study were seventh- and eight-grade students enrolled in a middle school located on the grounds of a large, metropolitan zoo. The school is a magnet school emphasizing the study of science to which students are admitted based on a lottery. A primarily qualitative, observational investigation was conducted over a 2-year period while the students worked cooperatively to create interactive displays for a touch-sensitive multimedia kiosk for the zoo.

Several categories emerged out of the qualitative analysis of the data, which included extensive videotapes, interviews, observations, and student-created materials. The students' strong appreciation that they were preparing multimedia materials for a real audience emerged as the core category in the analysis. Additional positive findings were:

(1) Students demonstrated great concern for accuracy in their displays; (2) students quickly assumed the major responsibility for content and editing decisions, despite the fact that the original task of designing the displays had been structured for them by the teacher; (3) students accessed wide ranges of science materials and sources to find the content they desired; and (4) their commitment to and enthusiasm for the project remained very high. On the negative side, the project failed to integrate its activities into the larger curriculum in the school or to attract the participation of teachers other than the computer coordinator. The bottom line was that by establishing an environment where creative thinking about content is combined with real-world assignments, students learned the content, enjoyed the learning process, and recognized that they have created something worthwhile.

Lehrer (1993) describes the development, use, and results of a hypermedia construction tool called HyperAuthor that was used by eighth-graders to design their own lessons about the American Civil War. This study exemplifies the principle that: "Cognitive tools empower learners to design their own representations of knowledge rather than absorbing knowledge representations preconceived by others." As Perkins (1986) maintains, knowledge is a process of design and not something to be transmitted from teacher to student. Thus, students should be engaged in "HyperComposition" by designing their own hypermedia (Lehrer, 1993). The process requires learners to transform information into dimensional representations, determine what is important and what is not, segment information into nodes, link the information segments by semantic relationships, and decide how to represent ideas. This is a highly motivating process because authorship results in ownership of the ideas in the presentation.

Students in the Lehrer (1993) study were high- and low-ability eighth-graders who worked at the hypermedia construction tasks for one class period of 45 minutes each day over a period of several months. The students worked in a media center of the school's library where they had access to a color Macintosh computer, scanner, sound digitizer, HyperAuthor software, and numerous print and nonprint resources about the Civil War. An instructor was also available to coach students in the conceptualization, design, and production of the hypermedia programs. Students created programs reflecting their unique interests and individual differences. For example, they created hypermedia about the role of women in the Civil War, the perspectives of slaves toward the war, and "not-so-famous people" from that period.

According to Lehrer (1993): "The most striking finding was the degree of student involvement and engagement" (p. 209). Both high- and low-ability students became very task oriented, increasingly so as they gained more autonomy and confidence with the cognitive tools. At the end of the study, students in the hypermedia group and a control group of students who had studied the Civil War via traditional classroom methods during the same period of time were given an identical teacher-constructed test of knowledge. No significant test differences were found. Lehrer conjectured that "these measures were not valid indicators of the extent of learning in the hypermedia design groups, perhaps because much of what students developed in the design context was not anticipated by the classroom teacher" (p. 218). However, a year later, when students in the design and control groups were interviewed by an independent interviewer unconnected with the previous year's work, important differences were found. Students in the control group could recall almost nothing about the historical content, whereas students in the design group displayed elaborate concepts and ideas that they had extended to other areas of history. Most importantly, although students in the control group defined history as the record of the facts of the past, students in the design class defined history as a process of interpreting the past from different perspectives. In short, the hypermedia "design approach lead to knowledge that was richer, better connected, and more applicable to subsequent learning and events" (p. 221).

Lehrer, Erickson, Love, and Connell (1994) conducted another study with ninth-grade students who were using HyperAuthor to develop hypermedia about World War I, lifestyles between 1870 and 1920, immigration, and imperialism. They found similar results to the aforementioned Civil War project: (1) Students' on-task behavior increased over time; (2) students perceived the benefits of planning and transforming stages of development; and (3) they developed generalizable skills such as taking notes, finding information, coordinating their work with other team members, writing interpretations, and designing presentations.

The Highly Interactive Computing Environments (HI-CE) Group at the University of Michigan has developed a multimedia composition tool called MediaText (Hays, Weingard, Guzdial, Jackson, Boyle & Soloway, 1993). They believe that rather than using media to deliver instruction to learners, learners should use the media to generate their own instruction and, in so doing, learn more about the content. The HI-CE group has studied high school students creating MediaText stories, biographies, or instructional aids, as well as multimedia essays. Students have learned to use techniques such as mentioning, directives, titling, and juxtaposition to integrate their documents. They have found that as students' experiences with MediaText increase, their documents become more integrated rather than merely annotated text. Students have been very enthusiastic about being constructionists (Papert, 1993), believing that they are learning more because they understand the ideas better.

The ACCESS (American Culture in Context: Enrichment for Secondary Schools) Project (Spoehr, 1994; Spoehr & Shapiro, 1991) focuses on the subject matter commonly taught in high school, such as United States history, American literature, and American studies. The project began with teachers assembling a collection of textual, pictorial, audio, and video materials to supplement their courses. Initially, students simply used the materials for information retrieval. Students who made more extensive use of the

conceptual organization built into the system benefited more than the students who used the system like a linear electronic book. The researchers found that hypermedia's effectiveness depends on the extent to which students can internalize the important conceptual structures in a subject matter as they browse.

Eventually, the ACCESS project orientation shifted from teacher-created hypermedia materials to student-generated hypermedia documents. To make it easier for students to create hypermedia projects, the ACCESS user interface was improved. Students generally produce several small hypermedia documents of increasing size and complexity early in the school year to become familiar with the authoring process. Later, they generally take on one or more major research projects, the results of which are presented as hypermedia.

According to Spoehr (1994), the structures that students impose on their hypermedia knowledge vary. A few students (5% to 10%) typically underutilize the power of the hypermedia and use a linear format (i.e., one overview card followed by a linear series of screens). Most students produce more interesting organizational types, including the "star," in which the entry point is an overview containing buttons to two or more subtopics, each of which appears as a linear sequence, and the "tree," in which one or more main branches off the initial overview in the program are subdivided into further subtopics that are then organized as linear sequences or divided into sub-subtopics. Students utilizing the "tree" organization (about 25% of the students) generally show more sophisticated understanding of the topic than students using the "star" structure.

There are many ways that the ACCESS Project students appear to benefit from their experiences as hypermedia authors, most of which fall into the category of superior knowledge representation and higher-order thinking skills. Spoehr (1993) reports that students who build and use hypermedia apparently develop a proficiency in organizing knowledge about a subject in a more expertlike fashion. For example, they are able to represent multiple linkages between ideas and organize concepts into meaningful clusters. In turn, these superior knowledge representations support more complex arguments in written essays. Most importantly, the conceptual organization skills acquired through building hypermedia are robust enough to allow students to generalize these skills to content that they acquire from other sources.

The studies described above, especially the research of Lehrer (1993) and Spoehr (1994), illustrate the need to investigate the effects of using hypermedia and multimedia as cognitive tools on the development of the higher-order thinking skills of students. In turn, focusing on higher-order outcomes requires an emphasis on alternative approaches to assessment (Mitchell, 1992; Reeves & Okey, 1996). Higher-order learning outcomes such as the ability to frame and resolve ill-defined problems or the tendency to exhibit intellectual curiosity are rarely directly observable. These types of outcomes can only be inferred from students' performance on a range of alternative assessments (Neimeyer, 1993). Alternative cognitive assessments will most likely be quite different from traditional testing procedures that assess lower-level knowledge and skills. Research focused on the higher-order outcomes of cognitive tools such as hypermedia/multimedia construction software must proceed hand in hand with the development of reliable, valid, and feasible cognitive assessments (Worthen, 1993).

24.7 SEMANTIC NETWORKING AS COGNITIVE TOOLS

24.7.1 What Are Semantic Networks?

A new genre of cognitive tools, semantic networking tools, has appeared in recent years. Programs such as SemNet (Fisher, 1990, 1992), Learning Tool (Kozma, 1987, 1992), and TextVision (Kommers, 1989) are cognitive tools that provide visual and verbal screen tools for developing concept maps, otherwise known as *cognitive maps* (see Fig. 24-1). Cognitive maps are spatial representations of ideas and their interrelationships that are stored in memory. These tools enable learners to interrelate the ideas they are studying as multidimensional networks of concepts, to label the relationships between those concepts, and to describe the nature of the relationships between all of the ideas in the network.

Semantic networks are representations of human memory structures. The cognitive theory underlying semantic networks maintains that human memory is organized semantically, that is, according to meaningful relationships between ideas in memory. These ideas, known as *schemas,* are arranged in networks of interrelated ideas known as *semantic networks.* Semantic networking programs are computer-based, visualizing tools for representing semantic networks. Perhaps the best-known theory of semantic networks is active structural networks (Quillian, 1968). These are mental structures composed of nodes (representing schemas) and ordered relationships or links connecting them. The nodes are instances of concepts or propositions, and the links describe the prepositional relationship between the nodes. In computer-based semantic networks, nodes are represented as information blocks or cards and the links as labeled lines (see Fig. 24-1).

The purpose of semantic networks is to represent the organization of ideas that someone knows about some phenomenon (e.g., baseball) or the underlying organization of ideas in a content domain (e.g., sociology). Semantic networks function as cognitive tools by engaging learners in analyzing the structural relationships among the content being studied. They can also be used as evaluation tools for assessing changes in thinking by learners (Preece, 1976). If we agree that a meaningful representation of memory is a semantic network, then learning can be thought of as a reorganization of semantic memory. Producing semantic networks reflect those changes in semantic memory, since the networks describe what a learner knows. In this way, semantic networking programs can be used to reflect knowledge acquisition.

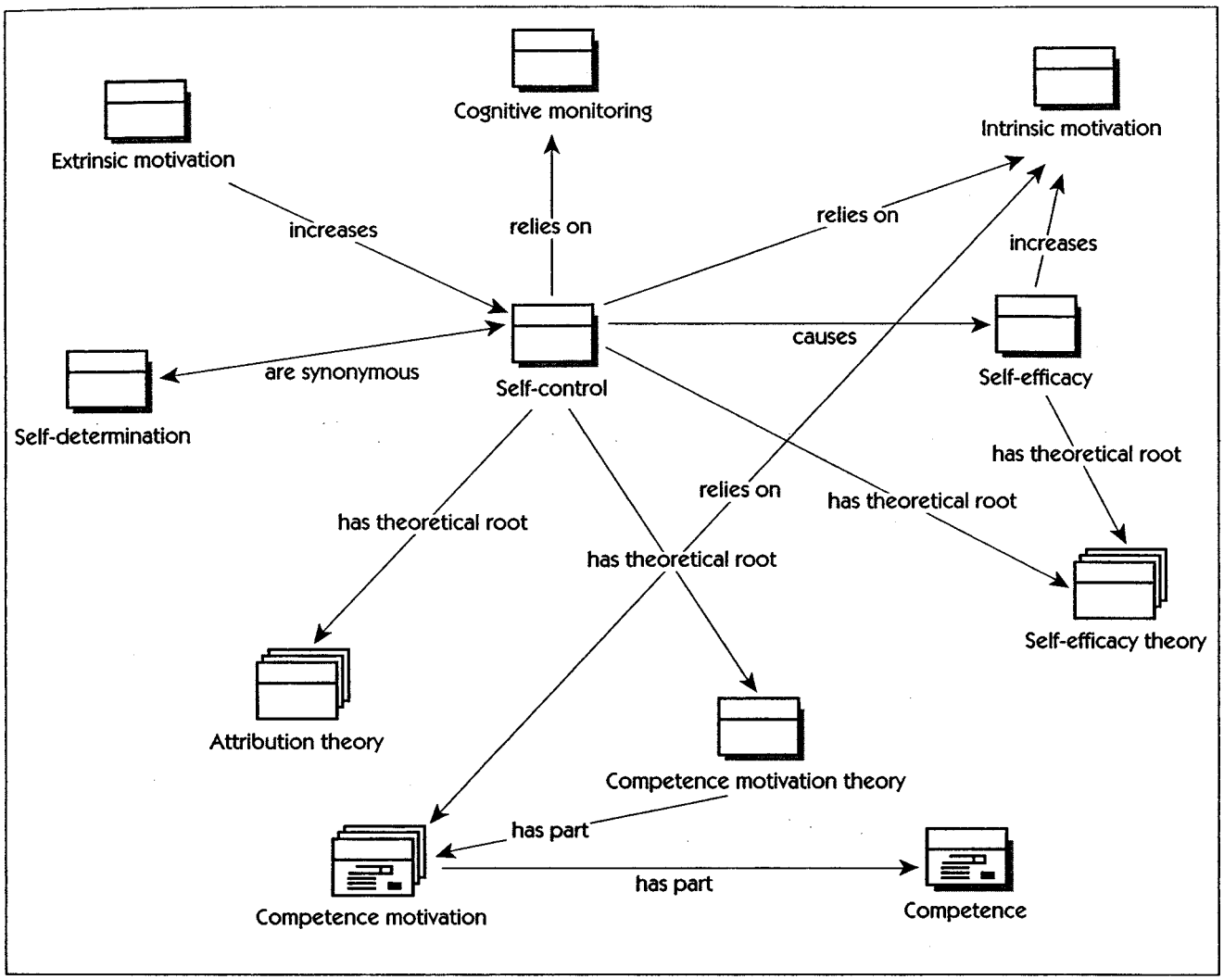

Figure 24-1. Semantic network of concepts in motivation theory.

24.7.2 How Are Semantic Networks Used as CognitiveTools?

Semantic networking aids learning by requiring learners to analyze the underlying structure of ideas they are studying. The process of creating semantic networks engages learners in an analysis of their own knowledge structures, which helps them integrate new knowledge into existing knowledge structures. The result is that the knowledge that is acquired can be used more effectively. Kozma (1987, 1992), one of the developers of the semantic networking tool Learning Tool, believes that semantic networks are cognitive tools that amplify, extend, and enhance human cognition. Constructing computer-based semantic nets engages learners in (1) the reorganization of knowledge through the explicit description of concepts and their interrelationships; (2) deep processing of knowledge, which promotes better remembering, retrieval, and the ability to apply knowledge in new situations; (3) relating new concepts to existing concepts and ideas which improves understanding (Davis, 1990); and (4) spatial learning through the spatial representation of concepts within an area of study (Fisher, Faletti, Patterson, Lipson, Thornton & Spring, 1990).

24.7.3 What Research Supports the Use of Semantic Networks as Cognitive Tools?

The usefulness of semantic nets and concepts maps is perhaps best indicated by their relationships to other forms of higher-order thinking. They have been significantly related to formal reasoning in chemistry (Schreiber & Abegg, 1991) and reasoning ability in biology (Briscoe & LeMaster, 1991; Mikulecky, 1988). Semantic networks can also provide a useful evaluation tool for measuring the acquisition of knowledge. In a geometry class, concept maps were used to evaluate teaching outcomes and to monitor student progress in the course (Mansfield & Happs, 1991).

An important research agenda in learning psychology focuses on the expert-novice distinction, comparing student knowledge representation with teacher or expert representations. Research has shown that during the process of

learning, the learner's knowledge structure begins to resemble the knowledge structures of the instructors, and the degree of similarity is a good predictor of classroom examination performance (Diekhoff, 1983; Shavelson, 1972, 1974; Thro, 1978). Instruction, then, may be conceived of as the mapping of subject-matter knowledge (usually that possessed by the teacher or expert) onto the learner's knowledge structure. Semantic nets are a way of measuring that convergence.

Using compared Pathfinder nets (Schvaneveldt, 1990), researchers have also shown that semantic nets are related to course examination performance (Goldsmith, Johnson & Acton, 1991). In a study examining the use of generating computer-based semantic networks in a computer programming course, Feghali (1991) found that students who built nets scored better in course tests; however, the differences were not statistically significant. More and better research of this type is needed to verify a consistent relationship between particular criteria for evaluating semantic nets and traditional measures of course performance, such as exams, research papers, or case studies. That research needs to relate semantic net construction with different cognitive outcomes, not just test performance. In fact, traditional test results may be the least effective variable to investigate. Tests of transfer to performance environments would be more useful dependent variables.

Another potentially important area of research with semantic networks involves changes in knowledge structures of learners as an outcome of learning. Constructing semantic networks and cognitive maps has been shown to be an accurate means for representing cognitive structure (Jonassen, 1987). That is, semantic networking helps learners to map their own cognitive structure. This affords researchers a powerful tool for verifying other learning effects. In a more recent study, Jonassen (1993) showed that building semantic networks in a course resulted in more consistent, hierarchical, and coherent knowledge structures than building expert systems in the same course. Semantic networks are a powerful knowledge analysis and integration tool that also provides means for assessing knowledge structures. Schema-based theories of learning (Rumelhart, 1980; Rumelhart & Ortony, 1977) suggest that learning is, at least in part, a reorganization of the learner's knowledge structure. Semantic network tools provide powerful assessment tools for evaluating those changes in knowledge structure.

24.8 EXPERT SYSTEMS AS COGNITIVE TOOLS

24.8.1 What Are Expert Systems?

Expert systems are computer-based tools that are designed to function as intelligent aids to decision making in all sorts of tasks. Early expert systems, such as MYCIN, were developed to help physicians diagnose bacterial infections with which they were unfamiliar. Prominent expert systems have also been developed to help geologists decide where to drill for oil, fire fighters decide how to extinguish different kinds of fires, computer sales technicians how to configure computer systems, and employees to decide among a large number of company benefits alternatives. Problems whose solution includes recommendations based on a variety of decisions are good candidates for expert systems.

Expert systems have evolved from research in the field of artificial intelligence (AI). AI is a field of computer science and cognitive science that focuses on the development of both hardware innovations and programming techniques that enable machines to perform tasks that are regarded as intelligent when done by people. *Intelligence* is the capacity to learn, reason, and understand. *Artificial* means simulated. So, in other words, AI researchers and expert system builders attempt to develop programs that simulate the human capability to reason and to learn. *Simulated* means only imitating a real object or event. AI programs, including expert systems, may perform functions that resemble human thinking, such as decision making. In reality though, AI programs are just computer programs; they only imitate a human activity within a narrowly defined situation.

An expert system, then, is a computer program that simulates the way human experts solve problems—an artificial decision maker. For example, when we consult an expert (e.g., doctor, lawyer, teacher) about a problem, the expert asks for current information about our condition, searches his or her knowledge base (memory) for existing knowledge that can be related to elements of the current situation, processes the information (thinks), arrives at a decision, and presents his or her solution. Like a human expert, an expert system (computer program) is approached by an individual with a problem. The system queries the individual about the current status of the problem, searches its own knowledge base (stored previously) for pertinent facts and rules that reflect the knowledge of an expert, processes the information, arrives at a decision, and reports the solution to the user.

Most expert systems consist of several components, including the knowledge base, inference engine, and user interface. The knowledge base consists of facts and rules that are programmed into the system by the designer. For example, an expert system designed to diagnose cars that will not start might include facts and rules such as:

Fact: Battery supplies voltage to ignition.
Fact: Ignition routes voltage to solenoid.
Rule: IF ignition is on,
AND solenoid is not engaged,
THEN battery is dead,
OR ignition switch is faulty.

The expert system inference engine is programmed into the system and acts on the knowledge base and current problem data to generate solutions. It sets a goal and then collects information from the knowledge base in order to yield a solution. When the knowledge base does not contain enough information, the inference engine asks the user to supply the missing information. The inference engine continues to seek information until it is able to reach a solution which the sys-

tem then presents to the user. The inference engine is the logic unit in the expert system. The part of the expert system that makes it a cognitive tool is the knowledge base. Building the knowledge base requires the designer to articulate the expertise that the system provides, not only in the form of facts but also rules. Identifying the causal relationships and procedural knowledge underlying a knowledge domain necessarily engages designers of expert systems in higher-order thinking.

24.8.2 How Are Expert Systems Used as Cognitive Tools?

When analyzing outcomes from building and using expert-systems, a distinction must be drawn between using an existing expert system rule base to support decision making and building an expert system. The former is the most common application. Although expert systems are primarily used in businesses as advisors that control production processes or in certain professions to assist practitioners in decision making, they also have many applications in education. Chandler (1994) describes the development of an expert-system design to help teachers plan science education lessons. Considerable research has focused on developing expert-system advisors to help teachers identify and classify learning disabled students (cf. Fuchs, 1992). Expert-system advisors have been developed to guide novices through the instructional development process (Tennyson & Christensen, 1991) or to assist students in selecting the correct statistical test (Karake, 1990; Saleem & Azad, 1992). With this type of application, professional knowledge engineers produce expert-system knowledge bases that are accessed by users when they need advice in making decisions (Bossinger & Milheim, 1993). However, simply using existing knowledge bases to get advice does not engage users as deeply as building a knowledge base to reflect their own thinking (Wideman & Owston, 1993). Querying a knowledge base to help solve a problem involves primarily comprehension of the problem and its factors; the application of some predetermined rules for solving the problem is often hidden from the user within the expert system itself.

Expert systems can also function as cognitive tools (Kommers, Jonassen & Mayes, 1992). Trollip, Lippert, Starfield, and Smith (1992) believe that the development of expert systems results in deeper understanding because they provide an intellectual environment that demands the refinement of domain knowledge, supports problem solving, and monitors the acquisition of knowledge. Building expert systems requires the developer to model explicitly the knowledge of the expert (Starfield, Smith & Bleloch, 1990). This entails identifying declarative knowledge (facts and concepts), structural knowledge (the knowledge of the interrelationships of ideas in memory), and procedural knowledge (how to apply the former). In fact, building expert systems is one of the few formalisms for depicting procedural knowledge. Psychologists usually represent procedural knowledge as a series of IF-THEN rules (Gagné, 1985); such a representation mode is obviously well suited to expert-system codification. As learners identify the IF-THEN structure of a domain, they will tend to understand the nature of decision- making tasks better, and this deeper understanding should make subsequent practice opportunities more meaningful. This is not to suggest that the mere development of an expert system necessarily leads learners to acquire the compiled procedural knowledge of a domain. Students could correctly identify many of the IF-THEN rules involved in flying an airplane, but actually acquiring the procedural expertise to fly would still require extended practice opportunities in realistic performance settings.

When expert systems are used as cognitive tools, the roles of teachers and students change dramatically. Students as knowledge engineers assume a more active role in acquiring prerequisite knowledge and focusing and directing interactions with the teacher, who assumes the role of expert (Morrelli, 1990). This frees the teacher from having to motivate students and allows them to respond as an expert to student probing concerning the more demanding and interesting aspects of various problems. Students must analyze the knowledge domain (identifying outcomes, factors, and values for those factors) and then synthesize rules and rule sequences. Morrelli argues that interaction between active, self-directed learners and a supportive, articulate teacher is an excellent model for learning science. We agree.

24.8.3 What Research Supports the Use of Expert Systems as Cognitive Tools?

Much of the research with the use of expert systems has focused on teachers and students as users of predefined rule bases. For instance, students who used an expert system to select the most appropriate statistical analysis procedure were more accurate in their selections and also retained the information better than students who used traditional computer-assisted instruction (Marcoulides, 1988). Grabinger and Pollock (1989) used expert systems to direct students to evaluate their own projects. Students who generated their own feedback with the help of expert systems produced a greater number of criteria in subsequent exercises and favored the method to teacher-only feedback. As described earlier, using expert systems supplants (provides or substitutes knowledge that is not known) thinking and therefore does not necessarily engage users in thinking critically about the content they are studying.

The use of expert systems as cognitive tools is relatively recent. Trollip and Lippert (1987) found that the analysis of subject matter required to develop expert systems is so deep and so incisive that learners develop a greater comprehension of their subject matter. They reported that building expert system rule bases engages learners in analytical reasoning, elaboration strategies such as synthesis, and metacognition. Lippert (1988, 1989), among the early advocates of expert systems as cognitive tools, argued that asking students to construct small rule bases is a valuable method for teaching problem solving

and knowledge structuring for students from sixth grade to adults. Not only do learners solve problems, they also engage in metacognitive reflection on their problem solving while constructing rule bases (Trollip & Lippert, 1988). Developing the knowledge base requires learners to isolate facts, variables, and rules about the relationships between content in a domain. Developing rule bases as a cognitive tool represents a constructivist application of expert systems (Jonassen, Wilson, Wang & Grabinger, 1993).

A small body of research has validated the use of expert systems as cognitive tools. Lai (1992) found that when nursing students developed medical expert systems, they developed enhanced reasoning skills and acquired a deeper understanding of the subject domain. Lippert (1988) described the development of rule bases to solve problems about forces by six freshmen physics students who used an expert-system shell to create questions, decisions, rules, and explanations pertaining to classical projectile motion. The students developed more refined, domain-specific knowledge due to greater degrees of elaboration during encoding and greater quantity of material processed in an explicit, coherent context, and therefore in greater semantic depth (Lippert & Finley, 1988). Students identified factors such as kind of force acting on an object (e.g., gravitational or centripetal), motion of the object (e.g., free fall, circular, or sliding), velocity of the object, and so on. The decisions that students defined were based on the laws that affect the motion and the formulas that should be applied. Students reported meaningful learning from evaluating their own thought processes, more enthusiasm for learning, and the learning of content that they were not expected to master.

Knox-Quinn (1992) reported that MBA (masters of business administration) students who developed knowledge bases on tax laws in an accounting course were consistently engaged in higher-order thinking, such as classifying information, breaking down content, organizing information, and integrating and elaborating information. All of the students who developed rule bases showed substantial gains in the quantity and quality of declarative and procedural knowledge and improved their problem-solving strategies. Students who built expert systems reasoned similarly to experts.

Like most cognitive tools, the research base on expert systems is very limited. However, with the increased interest in constructivist applications of expert systems and other computer tools, the research base should grow dramatically. We predict that future research will continue to verify the cognitive and metacognitive effects of learners functioning as knowledge engineers.

24.9 DATABASES AS COGNITIVE TOOLS

24.9.1 What Are Databases?

Databases are computerized record-keeping systems that were designed originally to replace paper-based information retrieval systems. A database consists of one or more files, each of which contains information in the form of a set of records (e.g., an individual's bank account information). Each record in a database is divided into fields, which describe the class or type of information contained therein. The same type of information is stored in each field in each record. An address database for a professional association might contain many records, each with information such as names and addresses of the members. These records are systematically broken down into fields (subunits of each record) that define a common pattern of information. For example, an address database might contain six fields, one each for the name, street address, city, state, zip code, and telephone number. The content and arrangement of each field is standardized within the records so that the computer will "know" which part of the record to search for to locate a particular kind of information. Database management systems (DBMS) provide the capability for managing, searching, and sorting information in a database as well as creating and defining new database files. Having defined the data structure, information can be entered into or deleted from the file. File management functions enable the user to make permanent copies of the information in the database.

The most important functions of DBMS are the organization tools that help us answer queries about information in the database. These tools include the search functions we can use to search through the database to find specific information. We can search the entire database or by specific fields, using Boolean combinations of search terms, such as AND, OR, and NOT. The other organizational tool that is used extensively is the sort function that enables us to rearrange the contents of the database, usually in ascending or descending order. Essentially, DBMS allow us to store information in an organized way and to locate or arrange the order of information to help us answer queries about the information in it. Most applications of databases support administrative purposes such as maintaining records of dues paid by members of a professional association. However, we can use the same functions to analyze and enter subject-matter content into databases, which can then be searched and sorted to answer specific questions about the content or to seek interrelationships and inferences among the content records. In short, databases may be used as cognitive tools.

24.9.2 How Are Databases Used as Cognitive Tools?

The organized and defined nature of a database facilitates the acquisition or collection of information and the analysis of content domains through the breaking down of information into its constituent parts. Therefore, knowledge databases can function as cognitive tools. McCurry and McCurry (1992) described the use of database software to classify types of seashells. Rooze (1988–89) emphasized the value of databases in social studies classes in terms of their placing students in active rather than passive roles. Rooze maintained that the creation of databases allows students to deter-

mine what information to collect and to organize seemingly unrelated bits of information into meaningful categories. Rooze reported that the teacher must guide the development of categories and search procedures if the students are going to be able to use the database effectively, and recommended a "Concept Development Strategy" and an "Interpretation of Data Strategy" in the development and uses of databases.

Knight and Timmins (1986) also recommended the use of databases to meet the objectives of history instruction. Pon (1984) described the use of database software as an inquiry tool to aid higher-order thinking in a fourth-grade American Indian studies course. Watson and Strudler (1988–89) described a lesson based on Taba's Inductive Thinking Model that teaches higher-order thinking using databases. Watson and Strudler concluded that building databases involves analyzing, synthesizing, and evaluating information, all clearly important critical-thinking skills.

24.9.3 What Research Supports the Use of Databases as Cognitive Tools?

The use of databases as cognitive tools has generated no formal research, and as a result, only anecdotal evidence exists to support their efficacy. However, the opportunities for assessing the effects of building and using databases is extensive. The following paragraphs describe some of the issues that can be researched.

There are several basic activities involved in developing and using knowledge databases in education, each of which engages a different combination of cognitive processes. The simplest application is filling in an existing database by searching for information that fits into the data structure. For instance, a database comparing the social and economic development of different countries might include fields such as gross national product (GNP), population, infant mortality rate, personal income, literacy rate, defense spending, and so forth. Students could consult reference sources to locate information to contribute to the database.

Querying the databases they have created also supports learning. Learners can use the database to answer or construct questions about the information in it, such as:

1. What is the relationship between the average income and literacy rate? Which country is different from others with a high literacy rate? How will recent events affect that country?
2. If you knew nothing about these countries except what is in the database, in which one would you want to live? Why?
3. How are infant mortality rate and literacy related to GNP?
4. Which are the most socially advanced countries? Based on which criteria?

Querying a database is a two-stage process: understanding the query and then following database-specific procedures for answering the query (Schlager, 1991).

Even more intellectual engagement may result from identifying a content domain, sensing an information need, and developing a data structure for accommodating the information to be included and the kinds of questions that need to be asked. A large number of critical-thinking skills are required to construct and use knowledge-oriented databases. The necessity for conducting research on databases as cognitive tools is considerable. Opportunities in this area are nearly unlimited at many levels of education and training.

We were unable to locate any formal empirical research to validate the use of databases as a cognitive tool. Many educators have practiced using databases as cognitive tools, but none that we are aware of have researched the effects. We would expect that constructing databases would certainly enhance recall and retention of information. We would also expect that building databases would improve students' ability to comprehend domain knowledge as well as draw inferences and implications from information. There is great research potential here for future researchers.

24.10 SPREADSHEETS AS COGNITIVE TOOLS

24.10.1 What Are Spreadsheets?

Spreadsheets are computerized, numerical record-keeping systems that were originally designed to replace paper-based accounting systems. Essentially, a spreadsheet is a grid, table, or matrix of empty cells with columns identified by letters and rows identified by numbers. Each cell may contain values, formulas, or functions. Numerical or textual data can be entered into each cell. Functions consist of mathematical or logical operations that also act on the values of the different cells, such as sum or average. Other functions automatically match values in cells with other cells, look-up values in a table of values, or create an index of values to be compared with other cells.

Spreadsheets have three primary functions: storing, calculating, and presenting information. Information, usually numerical, can be filed by a spreadsheet program into a particular location (the cell). This enables that information to be accessed and retrieved efficiently. Most importantly, spreadsheets support calculation functions. The numerical contents of any combination of cells can be mathematically related in just about any way the user wishes. Cells can be added, multiplied, and factored in any combinations of ways. Most spreadsheets provide mathematical functions such as logarithms and trigonometric functions. Contemporary spreadsheet software such as Microsoft Excel also includes sophisticated tools for generating tables and graphs.

Most spreadsheets support the entering of values with functions such as replication, whereby the program will fill in formulas in cells by replicating a formula in another cell. During spreadsheet construction, the author is not required to copy a similar formula over and over again in different cells. The spreadsheet can change the formula relative to

the position of the cell as well. Many spreadsheet programs also allow users to write "macros," i.e., procedures for automating a series of spreadsheet functions by using a single command.

Spreadsheets were originally developed to support business decision-making and accounting operations. They are especially useful for answering "what if" questions; e.g., what if interest rates increased by 1%? Changes need to be made in only one location, and the spreadsheet automatically recalculates all of the affected values. Spreadsheets are powerful problem-solving tools. However, the difficulty in using spreadsheets for problem solving depends on the amount of abstractness and information processing the problem contains (Leon-Argyla, 1988).

24.10.2 How Are Spreadsheets Used as Cognitive Tools?

Spreadsheets may be used as a cognitive tool for amplifying and reorganizing mental functioning. Spreadsheets completely restructured the work of budgeting for managers and business people around the globe, enabling planners to be hypothesis testers (playing "what if?" games) rather than calculators (Pea, 1985). The unique power of spreadsheets is sometimes credited with spurring the remarkable growth of microcomputers, starting with the development of the Visicalc spreadsheet in 1978 (Ditlea, 1984).

In the same way that spreadsheets have qualitatively changed the accounting process, they can change any educational process that involves working with quantitative information. The Working Group for Technology of the National Curriculum Commission (1990) charged with framing the national curriculum in Great Britain has recognized the role of spreadsheets as tools that enable students "to use information technology to explore patterns and relationships and to form and test sample hypotheses."

Spreadsheets are rule-using tools that require that users become rulemakers (Vockell & van Deusen, 1989). Calculating values in a spreadsheet requires that the user identify relationships and patterns among the data that he or she wants to represent in the spreadsheet. Next, those relationships must be modeled mathematically, using rules to describe the relationships in the model. Building spreadsheets requires abstract reasoning by the user, thereby matching one of the important goals of cognitive tools.

Spreadsheets support problem-solving activities. Given a problem situation with complex quantitative relationships, spreadsheets can be used to represent those relationships. The "what if?" thinking that is best supported by spreadsheets is essential to decision analysis (Sounderpandian, 1989). Such reasoning requires learners to consider implications of conditions or options, thereby engaging higher-order thinking.

Identifying values and developing formulas to interrelate them in spreadsheets enhance learners' understanding of the algorithms used to compare them and also the mathematical models used to describe content domains. Students understand calculations (both antecedents and consequents) because they are actively involved in identifying the interrelationships between the components of the calculation. Spreadsheet construction and use demonstrate all steps of problem solutions, showing the progression of calculations as they are performed. The spreadsheet process models the mathematical logic that is implied by calculations. Making the underlying logic obvious to learners should improve their understandings of the interrelationships and procedures.

Numerous educators have explored the use of spreadsheets as cognitive tools. Spreadsheets have frequently been used in mathematics classes for such purposes as a calculator to demonstrate multiplicative relationships in elementary mathematics (Edwards & Bitter, 1989); for root finding in precalculus using synthetic division, bisection methods, and Newton's method (Pinter-Lucke, 1992); for helping students to understand the meaning of large numbers (e.g., a million) by comparing quantities to everyday things (Parker & Widmer, 1991); for solving elementary mathematical story problems in math classes (Verderber, 1990); for implementing linear system algorithms for solving advanced mathematical formulas (Watkins & Taylor, 1989); and for implementing Polya's problem-solving plan with arithmetic problems (Sgroi, 1992).

Spreadsheets have often been used to manifest quantitative relationships in various chemistry and physics classes, such as calculating the dimensions of a scale model of the Milky Way to demonstrate its intensity (Whitmer, 1990); solving complex chemistry problems such as wet and dry analysis of flue gases, which may be expanded to include volumetric flow rate, pressure, humidity, dew point, temperature, and combustion temperature in a mass and balances course (Misovich & Biasca, 1990); modeling the stoichiometric relationships in chemical reactions and calculating how many bonds are broken, the energy required to break bonds, and the new masses and densities of the products and reagents in the reactions (Brosnan, 1990); calculating the force needed to lift various weights in various level positions (Schlenker & Yoshida, 1991); solving rate equation chemical kinetics problems in a physical chemistry course (Blickensderfer, 1990); calculating and graphing quantum mechanical functions such as atomic orbitals to simulate rotational and vibrational energy levels of atomic components in a physical chemistry class (Kari, 1990); solving challenging science problems, including incline plane simulations and converting protein into energy (Goodfellow, 1990); solving physics laboratory experiments such as time, displacement, velocity, and their interrelationships using a free-fall apparatus (Krieger & Stith, 1990); and estimating and comparing the relative velocities of different dinosaurs (Karlin, 1988).

Spreadsheets are also useful in supporting social studies instruction, such as representing Keynesian vs. classical macro-economical models including savings-investment and inflation-unemployment relationships (Adams & Kroch, 1989); supporting decision analysis by helping users

to find the best use of available information, as well as evaluating any additional information that can be obtained (Sounderpandian, 1989); interrelating demographic variables in population geography using population templates (Rudnicki, 1990); tracking portfolio performance in a stock-trading simulation (Crisci, 1992); and creating and manipulating economic models (e.g., balance of payments, investment appraisal, elasticity, and cost benefit analysis) in an economics course (Cashian, 1990).

Spreadsheets have been used in other disciplines as well. They have supported ecology education in the analysis of field data about tree species (Sigismondi & Calise, 1990) and the analysis of lunchroom trash to help students make projections of annual waste accumulation for an Earth Day project (Ramondetta, 1992). Spreadsheets have even been used to facilitate student grading of peer speech performances, providing a high level of motivation for students (Dribin, 1985).

It is sometimes useful to provide guided activities and problems to structure the use of spreadsheets. For example, to support higher-level thinking skills such as collecting, describing, and interpreting data, Niess (1992) provided students with a spreadsheet with wind data from various towns. Wind directions (NE, SW, WSW) described rows of data, with the percentage of days for each month of the year representing columns. She then asked students to use the spreadsheet to answer queries, such as:

- Are the winds more predominant from one direction during certain months? Why do you think this is the case?
- In which months is the wind the calmest?
- Which wind direction is the most stable during the year?

24.10.3 What Research Supports the Use of Spreadsheets as Cognitive Tools?

As with databases, there has been very little empirical study of the effects of using spreadsheets as cognitive tools. A few studies have examined the effects of different instructional treatments on learning to use spreadsheets (Charney, Reder & Kusbit, 1990; Kerr & Payne, 1994; Tiemann & Markle, 1990). These studies were not investigating the cognitive requirements or effects of using spreadsheets. Rather they were interested in the effects of different computer-based tutorial treatments, and spreadsheets happened to be the content or skill being learned. Baxter and Oatley (1991) compared the effectiveness of two different spreadsheet packages. Not surprisingly, the users' prior experience levels with spreadsheets was far more important to learning than the usability of the software package. These studies provide few insights about the effectiveness of spreadsheets as cognitive tools.

In one of the rare studies investigating spreadsheets as cognitive tools, Sutherland and Rojano (1993) were interested in how prealgebra students could use spreadsheets to represent and solve algebra problems. This study was conducted simultaneously in Britain and Mexico and took place over a 5-month period. During that time, students moved from a strict cause-effect local numerical notion of algebraic relationships to general rule-governed relationships that could be symbolized both in the spreadsheet and in algebraic notation. Another study used spreadsheets in community college math classes to help students solve linear and nonlinear equations problems (Hulse, 1992). Nonsignificant increases in mathematics achievement and decreases in numerical computation anxiety were reported; however, this study was so methodologically flawed by short treatment times and the use of inappropriate measures of achievement that it would be difficult to generalize the results.

All of the literature that we found provides accounts of how to use spreadsheets in various curricular application, along with some occasional anecdotal support for their use. For instance, Kari (1990) reported from several years of student use that students can learn both spreadsheet construction as well as physical chemistry concepts when provided with partially completed spreadsheets or spreadsheet templates. So the use of spreadsheets as cognitive tools remains speculative. Research on the cognitive outcomes from using spreadsheets is needed before we can conclude that they can function as generalizable cognitive tools.

24.11 CONCLUSIONS

We have presented a strong rationale for the application of cognitive tools in education, described a number of alternative approaches to using cognitive tools, and provided evidence that supports the uses of these cognitive tools in specific contexts. Support for some cognitive tools, such as programming languages, was found to be inconsistent. We also reported that very little research has been done to investigate the premise that cognitive tools based on common software such as databases and spreadsheets have beneficial effects on the development of higher-order thinking skills.

The overall finding that (1) learners develop critical-thinking skills as authors, designers, and constructors of knowledge and (2) learn more in the process than they do as the recipients of knowledge prepackaged in educational communications presents a major challenge for researchers in our field. For starters, this finding throws into question the value of much of the research reported elsewhere in this handbook. The findings of traditional educational communications research, already narrow in scope and limited in generalizability, may ultimately have little or no relevance in a world transformed by constructivist learning theory.

24.11.1 Future Research with Cognitive Tools

If nothing else, the research described in this chapter illustrates the need for sustained research agendas regarding the cognitive effects of cognitive tools, such as the studies carried out by Harel (1991), Kafai (1995), and other associates of Seymour Papert at MIT, rather than the isolated short-

term "pseudoscience" studies carried out in many educational communications and technology research areas such as learner control research (Reeves, 1993). Our hope is that the research studies reported above will inspire an improved research agenda for our field that will eventually reform educational practice.

There are hopeful signs of change within the educational research community, as evidenced in a recent theoretical paper by Ackermann (1994):

> An increasing number of software designers, cognitive scientists, and educators have come to the view that experience is actively constructed and reconstructed through direct interaction with the world, and that, indeed, knowledge is experience. According to this view, a learner is not an empty vessel to be filled, or a passive listener to be filled in. Knowledge is not a mere commodity to be transmitted from one person to another. It is not an entity to be emitted at one end, encoded, stored, retrieved, and reapplied at the other. The conduit metaphor is progressively fading away, and is being replaced by the more recent toolmaker paradigm. . . . Children are perceived as the active builders of their own cognitive tools, comprising both mental capacities and external mediations that prolong those mental capacities. Constructivism is in the air . . . (pp. 13, 14).

We share in Ackermann's enthusiasm for these changes, and also in the caution she issues later in her paper that "'Hands-on' won't do without 'heads-in'" (p. 15). We view cognitive tools as affording learners unparalleled opportunities for heads-in learning within constructivist learning environments. Cognitive tools, as described in this paper and elsewhere (cf. Jonassen, 1996), represent a vehicle for both educational restructuring and a research agenda that will have significant impact on the restructuring process. We invite others to join us in this quest. To further that quest, we would make some recommendations about methodologies that should be used the:

24.11.1.1. Multiple Assessments. The cognitive processes engaged by cognitive tools are complex and cannot be adequately assessed using a single type of measuring device. We recommend assessing the products of using cognitive tools as evidence of the thinking engaged by them. Criteria for evaluating those outcomes have not been verified, though many are suggested by Jonassen (1996). For instance, for evaluating semantic networks that are produced by students, a researcher might assess the:

- Number of nodes (breadth of the net)
- Number of instances (extent of the net)
- Ratio of instances to concepts (integratedness or embeddedness of concepts)
- Centrality of each node
- Depth (hierarchicalness) of the net
- Number of links (parsimony or economy of connections)
- Consistency in use of links
- Number of "dead-end" nodes (linked to only one other concept)
- Ratio of the number of links to the number of nodes

Other products of learning might include traditional tests (objective form), essays (look at the use of structural knowledge in the essays), speeches and presentations, and, most importantly, measures of transfer of learning. Cognitive tools should benefit learning transfer, so it is important to include assessments of problem solving in different domains or contexts, as well as in the domain being studied. These measures must assess the ability of learners to solve original problems, diagnose situations, draw implications or inferences from problem situations, or predict the result of changes in any problem situation.

In addition to assessing products of learning, it is important to assess the process as well. This can be accomplished by observing students as they work with cognitive tools and assessing variables such as effortful time-on-task, level of collaboration, or creativity. Research focused on how learners construct mental models may be especially fruitful (Jih & Reeves, 1992; Seel & Dinter, 1995). We recommend carefully analyzing the mix of methods employed by Lehrer (1993), Lehrer et al. (1994), and Spoehr (1992, 1993, 1994) in their investigation of learning from producing hypermedia knowledge bases. These are exemplary studies.

24.11.1.2. Qualitative Methodologies. In order to assess the complexities and subtleties of knowledge construction, it is essential to use qualitative as well as quantitative assessment strategies. It is impossible and even inappropriate to hypothesize all of the cognitive outcomes of using cognitive tools. The processes are too rich and unpredictable. Qualitative methodologies are elaborated in Chapters 40 and 41.

24.11.1.3. Meaningful Contexts and Assignments. Knowledge that is acquired in classrooms is "inert" in part because the purpose and context for learning such knowledge often has no relevance to learners whatsoever. The use of cognitive tools will likely result in greater learning if they are used in the context of solving some kind of problem that is meaningful to the learners. However, the meaningfulness of the context may also provide a researchable variable. Comparing the effects of cognitive tools used in a meaningful context with the use of cognitive tools as an adjunct memorization aid might provide some illuminating results.

24.11.1.4. Multiple Evaluations. In addition to using multiple-assessment methods, the evaluations made from those assessments should also vary. Comparing the learner's knowledge or interpretations with the expert's or teacher's may be reasonable in some contexts. After all, research has shown that during the process of learning, the learner's knowledge structure increasingly resembles the knowledge structures of the instructors, and the degree of similarity is a good predictor of classroom examination performance (Diekhoff, 1983; Shavelson, 1974; Thro, 1978). Given an instructivist context where the purpose of instruction is to get the learner to think like the teacher, researchers may wish to determine the extent of the learner's knowledge growth and compare it to that of the teacher or expert. In such a context, the products of using cognitive tools can provide evidence in a pretest-posttest fashion of how much the learner has learned.

However, from a constructivist perspective, it is always necessary to assess the learner's unique perspective in addition to making any external comparisons. Understanding the sense that learners make from studying any content domain may be far more informative than comparing the student's knowledge to that of the teacher or expert. With this in mind, it is often useful and illuminating to allow learners to create multiple products (perhaps using different tools) with reference to the same content or problem. Some learners may be better able to express themselves through multimedia, others through more abstract tools such as spreadsheets or databases.

24.12 A FINAL WORD

We perceive major differences between the types of studies we advocate for investigating computers as cognitive tools (cf. Harel, 1991; Lehrer, 1993) and the morass of pseudoscience research studies endemic in the field of educational communications and technology (Reeves, 1993). In the first examples, pedagogical models grounded in robust cognitive learning theories have been identified, and, subsequently, powerful technologies have been used to implement these models. In the latter, the power of various forms of technology to instruct has been assumed, and reductionist experiments were conducted to detect the effects on students. Further, in Harel's (1991) and Lehrer's (1993) studies, students with authentic needs experienced powerful learning opportunities over a period of weeks and months. By contrast, in most pseudoscience studies, undergraduates earn "extra credit" for less than an hour of their time spent using some form of mediated "treatment" that has little or no relevance for them. The ethics of conducting the latter types of reductionist experiments in education should be more closely examined.

In a landmark paper, Salomon (1991) describes the contrast between analytic and systemic approaches to educational research. Salomon claims that this analytic-systemic contrast transcends the "basic versus applied" or "quantitative versus qualitative" arguments that so often dominate debates about the relevancy of educational research. Salomon concludes that the analytic and systemic approaches are complementary, arguing that "the analytic approach capitalizes on precision, while the systemic approach capitalizes on authenticity" (p. 16).

While we agree with Salomon in theory, the dominance of pseudoscience in educational communications and technology research threatens to invalidate this complementarity in practice. Many of those who engage in analytic research approaches consistently violate the basic premises of the empirical paradigm they espouse, especially with respect to the testing of meaningful hypotheses derived from strong theory (Reeves, 1993). The question must be asked: Can society continue to afford the conduct of atheoretical analytical research in education on the scale reported elsewhere in this volume? We think not. Educational communications and technology require a much more valid body of systemic and analytical research grounded in sound theory than currently exists. Although limited in scope, we believe that the research focused on the effects of using computers as cognitive tools described in this chapter points the way toward a more valid and socially responsible research agenda for the 21st century.

REFERENCES

Ackermann, E. (1994). Direct and mediated experience: their role in learning. *In* R. Lewis & P. Mendelsohn, eds. *Lessons from learning.* Amsterdam: North-Holland.

Adams, F.G. & Kroch, E. (1989). The computer in the teaching of macroeconomics. *Journal of Economic Education 20* (3), 269–80.

Ahmed, A.M. (1992). *Learning to program and its transference to student's cognition.* (ERIC Document No. ED 352261.)

Apple Computer (1984). *HyperCard* [Computer software]. Cupertino, CA: Apple Computer.

Baxter, I. & Oatley, K. (1991). Measuring the learnability of spreadsheets in inexperienced users and those with previous spreadsheet experience. *Behaviour & Information Technology 10* (6), 475–90.

Beichner, R.J. (1994). Multimedia editing to promote science learning. *Journal of Educational Multimedia and Hypermedia 3* (1), 55–70.

Blickensderfer, R. (1990). Learning chemical kinetics with spreadsheets. *Journal of Computers in Mathematics and Science Teaching 9* (4), 35–43.

Bossinger, J. & Milheim, W.D. (1993). The development and application of expert systems: a national survey. *Educational Technology 33* (7), 7–17.

Briscoe, C. & LeMaster, S.U. (1991). Meaningful learning in college biology through concept mapping. *American Biology Teacher 53* (4), 214–19.

Brosnan, T. (1990). Using spreadsheets in the teaching of chemistry: two more ideas and some limitations. *School Science Review 71* (256), 53–59.

Bruce, B. & Rubin, A. (1993). *Electronic quills: a situated evaluation of using computers for writing in classrooms.* Hillsdale, NJ: Erlbaum.

Cafolla, R. (1987–88). Piagetian formal operations and other cognitive correlates of achievement in computer programming. *Journal of Educational Psychology 16* (1), 45–57.

Carver, S.M., Lehrer, R., Connell, T. & Ericksen, J. (1992). Learning by hypermedia design: issues of assessment and implementation. *Educational Psychologist 27* (3), 385–404.

Cashian, P. (1990). Spreadsheet investigations in economics teaching. *Economics 26* (110), 73–84.

Chandler, T.N. (1994). The science education advisor: applying a user-centered design approach to the development of an interactive case-based advising system. *Journal of Artificial Intelligence in Education 5* (3), 283–318.

Charney, D., Reder, L. & Kusbit, G.W. (1990). Goal setting and procedure selection in acquiring computer skills: a comparison of tutorial, problem solving, and learner exploration. *Cognition & Instruction 7* (4), 323–42.

Chin, J.P. & Zecker, S.G. (1985). Personality and cognitive factors influencing computer programming performance. Paper presented at the Annual Meeting of the Eastern Educational Research Association, Boston, MA (ERIC Document No. ED 261666).

Clark, R.E. (1994). Media will never influence learning. *Educational Technology Research and Development 42* (2), 21–29.

Clement, C.A., Kurland, D.M., Mawby, R. & Pea, R.D. (1986). Analogical reasoning and computer programming. *Journal of Educational Computing Research 2* (4), 473–85.

Clements, D.H. (1985). Research in Logo in education: is the turtle slow but steady or not even in the race? *Computers in the Schools 2* (2–3), 55–71.

— & Gullo, D.F. (1984). Effects of computer programming on young children's cognition. *Journal of Educational Psychology 76,* 1051–58.

Crisci, G. (1992, Jan.). Play the market! *Instructor 101* (5), 68–69.

Cuban, L. (1986). *Teachers and machines: the classroom use of technology since 1920.* New York: Teachers College Press.

Davis, N.T. (1990). Using concept mapping to assist prospective elementary teachers in making meaning. *Journal of Science Teacher Education 1* (4), 66–69.

Derry, S.J. & Lajoie, S.P. (1993). A middle camp for (un)intelligent instructional computing: an introduction. *In* S.P. Lajoie & S.J. Derry, eds. *Computers as cognitive tools,* 1–11. Hillsdale, NJ: Erlbaum.

Dick, W. & Carey, L. (1990). *The systematic design of instruction,* 3d ed. Glenview, IL: Scott, Foresman/Little, Brown.

Diekhoff, G.M. (1983). Relationship judgments in the evaluation of structural understanding. *Journal of Educational Psychology 75,* 227–33.

Ditlea, S., ed. (1984). *Digital deli.* New York: Workman.

Dribin, C.I. (1985, Jun.). Spreadsheets and performance: a guide for student-graded presentations. *The Computing Teacher 12* (9), 22–25.

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

—, Lowyck, J. & Jonassen, D.H., eds. (1993). *Designing environments for constructive learning.* Berlin: Springer.

Edwards, N.T. & Bitter, B.G. (1989, Oct.). Teaching mathematics with technology. *Arithmetic Teacher 37* (2), 40–44.

Feghali, A.A. (1991). A study of engineering college students' use of computer-based semantic networks in a computer programming language class (doctoral dissertation, Purdue University). *Dissertation Abstracts International 53* (3), 701.

Fischer, G.B. (1986). *Computer programming: a formal operational task.* Paper presented at the Annual Symposium of the Piaget Society, Philadelphia, PA (ERIC Document No. ED 275316).

Fischer, K.M. (1992). SemNet: a tool for personal knowledge construction. *In* P. Kommers, D. Jonassen & T. Mayes, eds. *Cognitive tools for learning.* Berlin: Springer.

— (1990). Semantic networking: new kid on the block. *Journal of Research in Science Teaching 27* (10), 1001–18.

—, Faletti, J., Patterson, H., Thornton, R., Lipson, J. & Spring, C. (1990). Computer assisted concept mapping. *Journal of College Science Teaching 19* (6), 347–52.

Foreman, K.H.D. (1988). Cognitive style, cognitive ability, and the acquisition of initial programming competence. *In* M.R. Simonson & J.K. Frederick, eds. *Proceedings of selected research papers presented at the Annual Meeting of the Association for Educational Communications and Technology,* New Orleans, LA.

Fuchs, L.S. (1992). Effects of expert system consultation within curriculum-based measurement using a reading maze task. *Exceptional Children 58* (5), 436–50.

Gagné, E. (1985). *The cognitive psychology of school learning.* Boston, MA: Little, Brown.

Gagné, R.M., Briggs, L.J. & Wager, W. (1987). *Principles of instructional design,* 3d ed. New York: Holt, Rinehart & Winston.

Goldsmith, T.E., Johnson, P.J. & Acton, W.H. (1991). Assessing structural knowledge. *Journal of Educational Psychology 83,* 88–96.

Goodfellow, T. (1990). Spreadsheets: powerful tools in science education. *School Science Review 71* (257), 47–57.

Goodlad, J.I. (1984). *A place called school: prospects for the future.* New York: McGraw-Hill.

Grabinger, R.S. & Pollock, J. (1989). The effectiveness of internally generated feedback with an instructional expert system. *Journal of Educational Computing Research 5* (3), 12–15.

Harel, I., ed. (1991). *Children designers: interdisciplinary constructions for learning and knowing mathematics in a computer-rich school.* Norwood, NJ: Ablex.

— & Papert, S. (1990). Software design as a learning environment. *Interactive Learning Environments 1* (1), 1–32.

Hays, K.E., Weingard, P., Guzdial, M., Jackson, S., Boyle, R.A. & Soloway, E. (1993, Jun.). *Students as multimedia authors.* Paper presented at ED-MEDIA '93, Orlando, FL.

Hulse, J.W. (1992). A comparison of the effects of spreadsheet use and traditional methods on student achievement and attitudes in selected collegiate mathematics topics (doctoral dissertation, University of South Dakota). *Dissertation Abstracts International 45* (8), 2723.

Jansson, L.C. (1987). Computer programming and logical reasoning. *School Science and Mathematics 87* (5), 371–79.

Jih, H.J. & Reeves, T.C. (1992). Mental models: a research focus for interactive learning systems. *Educational Technology Research and Development 40* (3), 39–53.

Jonassen, D.H. (1996). *Computers in the classroom: mindtools for critical thinking.* Columbus, OH: Prentice Hall.

— (1993). Changes in knowledge structures from building semantic net versus production rule representations of subject content. *Journal of Computer Based Instruction 20* (4), 99–106.

— (1989). *Hypertext/hypermedia.* Englewood Cliffs, NJ: Educational Technology.

— (1987). Assessing cognitive structure: verifying a method using pattern notes. *Journal of Research and Development in Education 20* (3), 1–14.

— (1984). The electronic notebook: integrating learning strategies into courseware to raise the level of processing. *In* G. Mills & B. Alloway, eds. *Aspects of educational technology.* London: Kogan Page.

—, Beissner, K. & Yacci, M.A. (1993). *Structural knowledge: techniques for representing, conveying, and acquiring structural knowledge.* Hillsdale, NJ: Erlbaum.

—, Campbell, J.P. & Davidson, M.E. (1994). Learning with media: restructuring the debate. *Educational Technology Research and Development 42* (2), 31–39.

— & Grabowski, B.L. (1993). *Handbook of individual differences, learning, and instruction.* Hillsdale, NJ: Erlbaum.

—, Myers, J.M. & McKillop, A.M. (1996). From constructivism to constructionism: learning with hypermedia/multimedia rather than from it. *In* B. Wilson, ed. *Constructivist learning environments,* 93–106. Englewood Cliffs, NJ: Educational Technology.

—, Wilson, B.G., Wang, S. & Grabinger, R.S. (1993). Constructivistic uses of expert systems to support learning. *Journal of Computer Based Instruction 20* (3), 86–94.

Kafai, Y.B. (1995). *Minds in play: computer game design as a context for children's learning.* Hillsdale, NJ: Erlbaum.

— (1996). Gender differences in children's constructions of video games. *In* Patricia M. Greenfield & Rodney R. Cocking, eds. *Interacting with video,* 39–66. Norwood, NJ: Ablex.

Karake, Z.A. (1990). Enhancing the learning process with expert systems. *Computers and Education 14* (6), 495–503.

Kari, R. (1990). Spreadsheets in advanced physical chemistry. *Journal of Computers in Mathematics and Science Teaching 10* (1), 39–48.

Karlin, M. (1988, Feb.). Beyond distance = rate * time. *The Computing Teacher 15* (5), 20–23.

Kerr, M.P. & Payne, S.J. (1994). Learning to use a spreadsheet by doing and by watching. *Interacting with Computers 6* (1), 3–22.

Knight, P. & Timmins, G. (1986). Using databases in history teaching. *Journal of Computer Assisted Learning 2* (2), 93–101.

Knox-Quinn, C. (1992, Apr.). *Student construction of expert systems in the classroom.* Paper presented at the Annual Meeting of the American Educational Research Association, San Francisco, CA.

Kommers, P.A.M. (1989). *TextVision* [computer software). Enschede, Netherlands: Twente University.

Kommers, P., Jonassen, D.H. & Mayes, T., eds. (1992). *Cognitive tools for learning.* Berlin: Springer.

Kozma, R.B. (1994). Will media influence learning? Reframing the debate. *Educational Technology Research and Development 42* (2), 7–19.

— (1992). Constructing knowledge with Learning Tool. *In* P. Kommers, D. Jonassen & T. Mayes, eds. *Cognitive tools for learning.* Berlin: Springer.

— (1987). The implications of cognitive psychology for computer-based learning tools. *Educational Technology 27* (11), 20–25.

Krieger, M.E. & Stith, J.H. (1990, Sep.). Spreadsheets in the physics laboratory. *The Physics Teacher 28* (6), 378–84.

Kurland, D.M., Pea, R.D., Clement, C. & Mawby, R. (1986). A study of the development of programming ability and thinking skills in high school students. *Journal of Educational Computing Research 2* (4), 429–55.

Lai, K.W. (1992, Mar.). *Acquiring expertise and cognitive skills in the process of constructing an expert system: a preliminary study.* Paper presented at the Annual Meeting of the American Educational Research Association, San Francisco, CA.

Lajoie, S.P. (1990, Apr.). *Computer environments as cognitive tools for enhancing mental models.* Paper presented at the Annual Meeting of the American Educational Research Association, Boston, MA.

— & Derry, S.J., eds. (1993). *Computers as cognitive tools.* Hillsdale, NJ: Erlbaum.

Langer, E.J. (1989). *Mindfulness.* Reading, MA: Addison-Wesley.

Lehrer, R. (1993). Authors of knowledge: patterns of hypermedia design. *In* S.P. Lajoie & S.J. Derry, eds. *Computers as cognitive tools,* 197–227. Hillsdale, NJ: Erlbaum.

Lehrer, R., Erickson, J. & Connell, T. (1994). Learning by designing hypermedia documents. *Computers in Schools 10* (1–2), 227–54.

Leon-Argyla, E.R. (1988). A study of spreadsheet problem solving and testing for problem solving ability (doctoral dissertation, Michigan State University). *Dissertation Abstracts International 49* (10), 3005.

Liao, Y.K.C. & Bright, G.W. (1991). Effects of computer programming on cognitive outcomes. *Journal of Educational Computing Research 7* (3), 251–68.

Lippert, R.C. (1989). Expert systems: tutors, tools, and tutees. *Journal of Computer-Based Instruction 16* (1), 11–19.

— (1988). An expert system shell to teach problem solving. *Tech Trends 33* (2), 22–26.

— & Finley, F. (1988, Apr.). *Student's refinement of knowledge during the development of knowledge bases for expert systems.* Paper presented at the Annual Meeting of the National Association for Research in Science Teaching, Lake of the Ozarks, MO (ERIC Document No. ED 293872).

Mansfield, H. & Happs, J. (1991). Concepts maps. *Australian Mathematics Teacher 47* (3), 30–33.

Marcoulides, G.A. (1988). An intelligent computer-based learning program. *Collegiate Microcomputer 6* (2), 123–26.

McCoy, L.P. (1990). Literature related to learning problem solving in mathematics and computer programming. *School Science and Mathematics 90* (1), 48–60.

McCurry, N. & McCurry, A. (1992, Apr.). Science: database programs and the study of seashells. *The Computing Teacher 19* (7), 32–34.

Merrill, M.D., Li, Z. & Jones, M.K. (1990). Second generation instructional design (ID2). *Educational Technology 30* (2), 7–14.

Mikulecky, L. (1988). *Development of interactive programs to help students transfer basic skills to college level science and behavioral sciences courses.* Bloomington, IN: Indiana University (ERIC Document No. ED 318469).

Miller, R.B., Kelly, G.N. & Kelly, J.T. (1988). Effects of Logo computer programming experience on problem solving and spatial relations ability. *Contemporary Educational Psychology 13,* 348–57.

Misovich, M. & Biasca, K. (1990). The power of spreadsheets in a mass and energy balance course. *Chemical Engineering Education 24,* 46–50.

Mitchell, R. (1992). *Testing for learning: how new approaches to evaluation can improve American schools.* New York: Free Press.

Morrelli, R. (1990). The student as knowledge engineer: a constructivist model for science education. *Journal of Computing in Higher Education 2* (1), 78–102.

Nachmias, R. (1986, Apr.). *Variables—an obstacle to children learning computer programming.* Paper presented at the Annual Meeting of the American Educational Research Association, San Francisco, CA (ERIC Document No. ED 290459).

National Curriculum Commission. (1990). *Technology in the national curriculum.* London: National Curriculum Commission.

Neimeyer, G.J. (1993). *Constructivist assessment: a casebook.* Newbury Park, CA: Sage.

Nelson, T. (1980). Replacing the printed word: a complete literary system. *In* S.H. Lavingston, ed. *Proceedings of the IFIP Congress,* 1013–23. Amsterdam: North-Holland.

Nielsen, J. (1990). *Hypertext & hypermedia.* Boston, MA: Academic Professional.

Niess, M.L. (1992, Mar.). Math: winds of change. *The Computing Teacher 19* (6), 32–35.

Norman, D.A. (1983). Some observations on mental models. *In* D. Gentner & A.L. Stevens, eds. *Mental models.* Hillsdale, NJ: Erlbaum.

Nowaczyk, R.H. (1983, Mar.). *Cognitive skills needed in computer programming.* Paper presented at the Annual Meeting of the Southeastern Psychological Association, Atlanta, GA (ERIC Document No. ED 236466).

Oprea, J.M. (1988). Computer programming and mathematical thinking. *Journal of Mathematical Behavior 7,* 15–190.

Papert, S. (1993). *The children's machine: rethinking school in the age of the computer.* New York: Basic Books.

Papert, S. (1980). *Mindstorms: children, computers and powerful ideas.* New York: Basic Books.

Parker, J. & Widmer, C.C. (1991, Sep.). Teaching mathematics with technology: how big is a million? *Arithmetic Teacher 39* (1), 38–41.

Pea, R.D. (1985). Beyond amplification: using the computer to reorganize mental functioning. *Educational Psychologist 20* (4), 167–82.

— & Kurland, D.M. (1987). On the cognitive effects of learning computer programming. *In* R.D. Pea & K. Sheingold, eds. *Mirrors of minds: patterns of experience in educational computing,* 147–77. Norwood, NJ: Ablex.

Perelman, L.J. (1992). *School's out: hyperlearning, the new technology, and the end of education.* New York: Morrow.

Perkins, D.N. (1993). Person-plus: a distributed view of thinking and learning. *In* G. Salomon, ed. *Distributed cognitions: psychological and educational considerations,* 88–110. Cambridge, UK: Cambridge University Press.

— (1986). *Knowledge as design.* Hillsdale, NJ: Erlbaum.

Phillips, D.C. (1995). The good, the bad, and the ugly: the many faces of constructivism. *Educational Researcher 24* (7), 5–12.

Pinter-Lucke, C. (1992). Rootfinding with a spreadsheet in pre-calculus. *Journal of Computers in Mathematics and Science Teaching 11* (1), 85–93.

Polson, M.C. & Richardson, J.J., eds. (1988). *Foundations of intelligent tutoring systems.* Hillsdale, NJ: Erlbaum.

Pon, K. (1984, Nov.). Databasing in the elementary (and secondary) classroom. *The Computing Teacher 12* (3), 28–30.

Popyack, J.L. (1989). *Karel* [computer software]. Santa Barbara, CA: Intellimation.

Preece, P.F.W. (1976). Mapping cognitive structure: a comparison of methods. *Journal of Educational Psychology 68* (1), 1–8.

Quillian, M.R. (1968). Semantic memory. *In* M. Minsky, ed. *Semantic information processing,* 43–67. Cambridge, MA: MIT Press.

Ramondetta, J. (1992, Apr./May). Using computers: learning from lunchroom trash. *Learning 20* (8), 59.

Reeves, T.C. (1993). Pseudoscience in computer-based instruction: the case of learner control research. *Journal of Computer-Based Instruction 20* (2), 39–46.

— & Okey, J. (1996). Alternative assessment for constructivist learning environments. *In* B. Wilson, ed. *Constructivist learning environments,* 191–202. Englewood Cliffs, NJ: Educational Technology.

Rooze, G.E. (1988-89). Developing thinking with databases: what's really involved? *Michigan Social Studies Journal 3* (1), 25–26.

Rudnicki, R. (1990). Using spreadsheets in population geography classes. *Journal of Geography 89* (3), 118–22.

Rumelhart, D.E. (1980). Schemata: the building blocks of cognition. *In* R.J. Spiro, B.C. Bruce & W.F. Brewer, eds. *Theoretical issues in reading comprehension: perspectives from cognitive psychology, linguistics, artificial intelligence, and education.* Hillsdale, NJ: Erlbaum.

— & Ortony, A. (1977). The representation of knowledge in memory. *In* R.C. Anderson, R.J. Spiro & W.E. Montague, eds. *Schooling and the acquisition of knowledge.* Hillsdale, NJ: Erlbaum.

Saleem, N. & Azad, A.N. (1992). Expert systems as a statistics tutor on call. *Journal of Computers in Mathematics and Science Teaching 11* (2), 179–91.

Salomon, G. (1993). On the nature of pedagogic computer tools: the case of the writing partner. *In* S.P. Lajoie & S.J. Derry, eds. *Computers as cognitive tools,* 179–96. Hillsdale, NJ: Erlbaum.

— (1991). Transcending the qualitative-quantitative debate: the analytic and systemic approaches to educational research. *Educational Researcher 20* (6), 10–18.

— & Globerson, T. (1987). Skill may not be enough: the role of mindfulness in learning and transfer. *International Journal of Educational Research 11,* 623–38.

— Perkins, D.N. & Globerson, T. (1991). Partners in cognition: extending human intelligence with intelligent technologies. *Educational Researcher 20* (3), 2–9.

Schlager, M.S. (1991). *An analysis of database querying as a cognitive skill.* Doctoral dissertation, University of Colorado at Boulder, 1991.

Schlenker, R.M. & Yoshida, S.J. (1991, Feb.). A clever lever endeavor: you can't beat the spreadsheet. *The Science Teacher 58* (2), 36–39.

Schreiber, D.A. & Abegg, G.L. (1991, Apr.). *Scoring student-generated concept maps in introductory college chemistry.* Paper presented at the Annual Meeting of the National Association for Research in Science Teaching, Lake Geneva, WI (ERIC Document No. 347055).

Schvaneveldt, R.W., ed. (1990). *Pathfinder associative networks: studies in knowledge organization.* Norwood, NJ: Ablex.

Seel, N.M. & Dinter, F.R. (1995). Instruction and mental model progression: learner-dependent effects of teaching strategies on knowledge acquisition and analogical transfer. *Educational Research and Evaluation 1* (1), 4–35.

Sgroi, R.J. (1992, Mar.). Systematizing trial and error using spreadsheets. *Arithmetic Teacher 39* (7), 8–12.

Shavelson, R.J. (1974). Methods for examining representations of subject matter structure in students' memory. *Journal of Research in Science Teaching 11* (3), 231–49.

— (1972). Some aspects of the correspondence between content structure and cognitive structure in physics instruction. *Journal of Educational Psychology 63* (3), 225–34.

Shlechter, T.M., ed. (1991). *Problems and promises of computer-based training.* Norwood, NJ: Ablex.

Sigismondi, L.A. & Calise, C. (1990). Integrating basic computer skills into science classes: analysis of ecological data. *The American Biology Teacher 52* (5), 296–301.

Simon, H.A. (1987). Computers and society. *In* S.B. Kiesler & L.S. Sproul, eds. *Computing and change on campus,* 4–15. New York: Cambridge University Press.

Sleeman, D., Putnam, R.T., Baxter, J. & Kuspa, L. (1986). Pas-

cal and high school students: a study of errors. *Journal of Educational Computing Research 2* (1), 6–10.

Sounderpandian, J. (1989). Decision analysis using spreadsheets. *Collegiate Microcomputer 7* (2), 157–63.

Spector, J.M., Polson, M.C. & Muraida, D.J., eds. (1993). *Automating instructional design: concepts and issues.* Englewood Cliffs, NJ: Educational Technology.

Spoehr, K.T. (1994). Enhancing the acquisition of conceptual structures through hypermedia. *In* K. McGilly, ed. *Classroom lessons: integrating cognitive theory and classroom practice.* Cambridge, MA: MIT Press.

— (1993, Apr.). *Profiles of hypermedia authors: how students learn by doing.* Paper presented at the Annual Meeting of the American Educational Research Association, Atlanta, GA.

— (1992, Apr.). *Using hypermedia to clarify conceptual structures: illustrations from history and literature.* Paper presented at the Annual Meeting of the American Educational Research Association, Atlanta, GA.

— & Shapiro, A. (1991). *Learning from hypermedia: making sense of a multiply-linked database.* (ERIC Document No. ED 333864.)

Starfield, A.M., Smith, K.A. & Bleloch, A.L. (1990). *How to model it: problem solving for the computer age.* New York: McGraw-Hill.

Stoll, C. (1995). *Silicon snake oil: second thoughts on the information highway.* New York: Doubleday.

Sutherland, R. & Rojano, T. (1993). A spreadsheet approach to solving algebra problems. *Journal of Mathematical Behavior 12* (4), 353–83.

Swan, K. & Black, J.B. (1990, Apr.). *Logo programming, problem solving, and knowledge-based instruction.* Paper presented at the Annual Meeting of the American Educational Research Association, Boston, MA (ERIC Document No. ED 349968).

Taylor, K.A. (1991). *An annotated bibliography of current literature dealing with the effective teaching of computer programming in high schools.* (ERIC Document No. 338217.)

Taylor, R., ed. (1980). *The computer in the school: tutor, tool, tutee.* New York: Teachers College Press.

Tennyson, R.D. & Christensen, D.L. (1991). Automating instructional systems development. *In* M.R. Simonson, ed. *Proceedings of selected research presentations at the Annual Conference of the Association for Educational Communications and Technology,* 885–915. Washington, DC: Association for Educational Communications & Technology.

Thro, M.P. (1978). Relationships between associative and content structure of physics concepts. *Journal of Educational Psychology 70* (6), 971–78.

Tiemann, P.W. & Markle, S.M. (1990). Effects of varying interactive strategies provided by computer-based tutorial for a software application program. *Performance Improvement Quarterly 3* (2), 48–64.

Trollip, S. & Lippert, R. (1988). Constructing knowledge bases: a promising instructional tool. *Journal of Computer-Based Instruction 14* (2), 44–48.

—, Lippert, R., Starfield, A. & Smith, K.A. (1992). Building knowledge bases: an environment for making cognitive connections. *In* P. Kommers, D. Jonassen & T. Mayes, eds. *Cognitive tools for learning.* Berlin: Springer.

Verderber, N.L. (1990). Spreadsheets and problem solving with AppleWorks in mathematics teaching. *Journal of Computers in Mathematics and Science Teaching 9* (3), 45–51.

Vockell, E. & van Deusen, R.M. (1989). *The computer and higher-order thinking skills.* Watsonville, CA: Mitchell.

von Glaserfeld, E. (1989). An exposition of constructivism: why some like it radical. *In* R.B. Davis, C.A. Maher & N. Noddings, eds. *Constructivist views on the teaching and learning of mathematics.* Athens, GA: JRME Monographs.

von Wodtke, M. (1993). *Mind over media: creative thinking skills for electronic media.* New York: McGraw-Hill.

Watkins, W. & Taylor, M. (1989). A spreadsheet in the mathematics classroom. *Collegiate Microcomputer 7* (3), 233–39.

Watson, J. & Strudler, N. (1988-89). Teaching higher-order thinking skills with databases. *The Computing Teacher 16* (4), 47–50, 55.

Whitmer, J. C. (1990, Oct.). Modeling the Milky Way: spreadsheet science. *The Science Teacher 57* (7), 19–21.

Wideman, H.H. & Owston, R.D. (1993). Knowledge base construction as a pedagogical activity. *Journal of Educational Computing Research 9* (2), 165–96.

Williamson, J.D. & Ginther, D.W. (1992). Knowledge representation and cognitive outcomes of teaching Logo to children. *Journal of Computing in Childhood Education 3* (3-4), 303–22.

Wittrock, M.C. (1974). Learning as a generative activity. *Educational Psychologist 11,* 87–95.

Worthen, B.R. (1993). Critical issues that will determine the future of alternative assessment. *Phi Delta Kappan 74* (6), 444–56.

Yeaman, A.R.J. (1994). Deconstructing modern educational technology. *Educational Technology 34* (2), 15–24.