Mainstreaming Computers

Effectively integrating computers into the curriculum depends on three key variables: resource allocation, quality software, and implementation methods.

Douglas Carnine

The roots of the University of Oregon's Mainstreaming Computers Project lie in the Direct Instruction Follow Through Model (Becker and Carnine, 1980). We began in 1968 by asking the question that is often heard today: how can schools become more effective? Since the U.S. Office of Education's Follow Through Project targeted low-income primary grade students, the question posed a particularly serious challenge. It meant examining resource allocation, instructional design characteristics of the curriculum, and procedures for implementing innovative practices. The independent evaluation of Follow Through models indicated significant positive results on both achievement and affective measures for direct instruction students (Abt Associates, 1977).

Recently, about 1,500 of those originial direct instruction students (who are now high school age) were subjects of a follow-up study carried out in schools in six states. These students scored significantly higher than control students on one or more measures in each district—academic achievement, attendance, or the likelihood of graduating from high school or applying for post-secondary education (Gersten and Carnine, 1983).

The findings are encouraging. Our assumption in 1968 that every student could learn to read, given enough careful instruction, was seemingly borne out. However, a closer look reveals that the demands placed on schools, in terms of the amount and intensity of teaching, were in many cases too great. Teaching reading, mathematics, and language to groups of students in a thorough, even tenacious, manner is very demanding. And with repeated draconian budget cuts, just a few of the 22 original Follow Through districts are providing more than a fraction of education services they delivered ten years ago.

Jamieson McKenzie

Frustrations in creating interest and support for labor-intensive instructional practices led us to computers. Trying to reform schools by asking educators to do more and more each year is futile. If increases in educational productivity are possible, they will come as they have in other fields—as outgrowths of technological innovations. Our task then, in 1982, was to discover and describe how computers could increase teacher productivity, rather than serving as learning adjuncts or games.

The planning in 1982 grew out of the same two assumptions that guided our thinking in 1968: (1) the teacher (rather than the principal or child) is pivotal in any school improvement effort, and (2) any intervention must be as comprehensive as possible. The administrator's role is important, although indirectly, in creating conditions in the school that

Douglas Carnine is Professor, College of Education, University of Oregon, Eugene.

allow teachers to successfully implement the intervention. As curriculum leaders attempting to create conditions for effective computer use, we again looked at resource allocation, instructional quality, and methods of implementation.

Our most significant *resource allocation* decision in 1968 was to hire paraprofessionals as math and language teachers in order to increase the amount of adult teaching in each classroom to two or three "teachers." In 1982 we diverted our resources to computers, hoping that they would both provide and support instruction.

Our concern about the *instructional quality* of curriculum materials, in Follow Through classrooms led us to develop the DISTAR curriculum. Today, concern about software quality is manifested in a proliferation of software evaluation forms, and systems and languages to aid teachers in writing their own software. The future of educational computing depends largely on educators' ability to set forth explicit standards for quality in software.

Our beliefs about *implementation* of innovative practices have changed very little since the 1968 study. It's still true that innovations must not pursue esoteric objectives or be too time consuming; that teachers need classroom-based training to become proficient in a new innovation; and that teachers need to observe students succeeding as a direct result of the innovation.

As in Follow Through, the Mainstreaming Computers Project's treatment of the three key variables is hardly exhaustive; the goal is a comprehensive, yet *workable*, model that offers focus and specificity. Therefore, several important dimensions in applying computers to schooling (such as computer literacy and computer languages) are not included in the model. In fact, the particulars of the project may be of less significance than the model itself—a comprehensive, concerted effort at planning, developing, and implementing a project that addresses variables important to any effort to mainstream computers.

Resource Allocation

Curriculum leaders reviewing budgets for computer applications face several obvious problems. The first is lack of funds. The second problem is confusion about how to spend what money there is. A lack of strategies for using computers, once purchased, is even more disturbing. About 75 percent of schools with computers actually utilize them less than half of the day (Euchner, 1983). The Mainstreaming Computers Project sought an interim solution to the problem of limited access, a compromise between the "ideal" of a computer for every student and the "reality" of a few computers for every school. A clue lay in the problem itself. In the rush to provide students with computer access, the ways in which computers can help *teachers* were never seriously considered. For instance, teachers will benefit greatly by being relieved of the need to score assignments. Time freed up from these grading chores can be more profitably used for reading and analyzing compositions, working individually with students, and so on. Another benefit of automated scoring is timely feedback about how well students are learning; for example, analyses of diagnostic test results or performance summaries of students' answers to lecture questions. These computer applications illustrate how technology can support teachers.

> "The failure of computers in education will not result from applications that are too ambitious, but from applications that are not ambitious enough."

Not only can computers support instruction, but they can also deliver instruction. Typically, each student receives CAI instruction on an individual computer, which is relatively expensive. One way to decrease the cost is for several students to enter answers on individual keyboards, with all answers processed by a single computer. The computer will advance or branch according to the responses of the students as a group.

The ways computers can increase teacher productivity—by supporting instruction or directly providing instruction—tend to be either expensive (with a computer for each student), or time consuming (with students taking turns or with the teacher entering students' answers on a single computer). While looking for other options, another option found us. Elwyn Rees, the inventor of *Teacher Net*, telephoned while on a visit to the U.S. and suggested a collaboration that would link his hardware (Teacher Net) with our instructional design expertise.

Teacher Net is a computer peripheral that allows numerous keyboards to operate from one computer. The relatively low cost of a Teacher Net peripheral and number pads (less than $70 per student for a class of 30 students) is one way of addressing the problem of limited access to computers in schools. The device can help teachers monitor student learning, provide CAI to entire classes, streamline testing, and provide practice in discriminating complex behaviors that are to be learned.

There are many ways to configure Teacher Net and a computer in a classroom. A conventional arrangement appears in Diagram A of Figure 1. The teacher is at the front of the classroom, and the computer is at the back. One student at a time works at the computer.

In Diagram B, cables connect Teacher Net with five students for small-group instruction. Students view a monitor, which presents information, questions, and feedback. The teacher, while working with the rest of the class, can view a second monitor, which summarizes the group's performance—both level of mastery and content covered.

In Diagram C, Teacher Net is used with 30 students to administer a group test, monitor independent work, or receive feedback about students' mastery of a lecture. Students view a single line display on their key pad (similar to that found on a calculator), which shows the last student entry.

The most elaborate application of Teacher Net appears in Diagram D. Groups of students are at different places in different CAI programs. The power of this configuration is that a teacher could respond to a wide ability range of students in a single class. The teacher

could work with one group of students while other groups progress through different CAI programs.

Applications of Teacher Net require software that is compatible with the computer linked to Teacher Net. The software is simple to use, requiring the teacher to answer such questions as: How many students will be responding? What type of answers will they make: single digit (for true/false, yes/no, multiple choice); multiple digit (numerical answers to math problems). After entering the answers to these questions and the students' names, the teacher selects a form for summarizing student perforamance; such as, percent correct for the group, percent correct for each individual, items missed by each student, and so on. The group can then begin responding to a variety of applications. For example, an entire class can take a diagnostic test at one time; and at the end of the testing session, the teacher can receive individual diagnoses of skill deficiencies and prescriptions for instruction. Results can be printed out, and possibly prescriptions as well (Hofmeister, 1982).

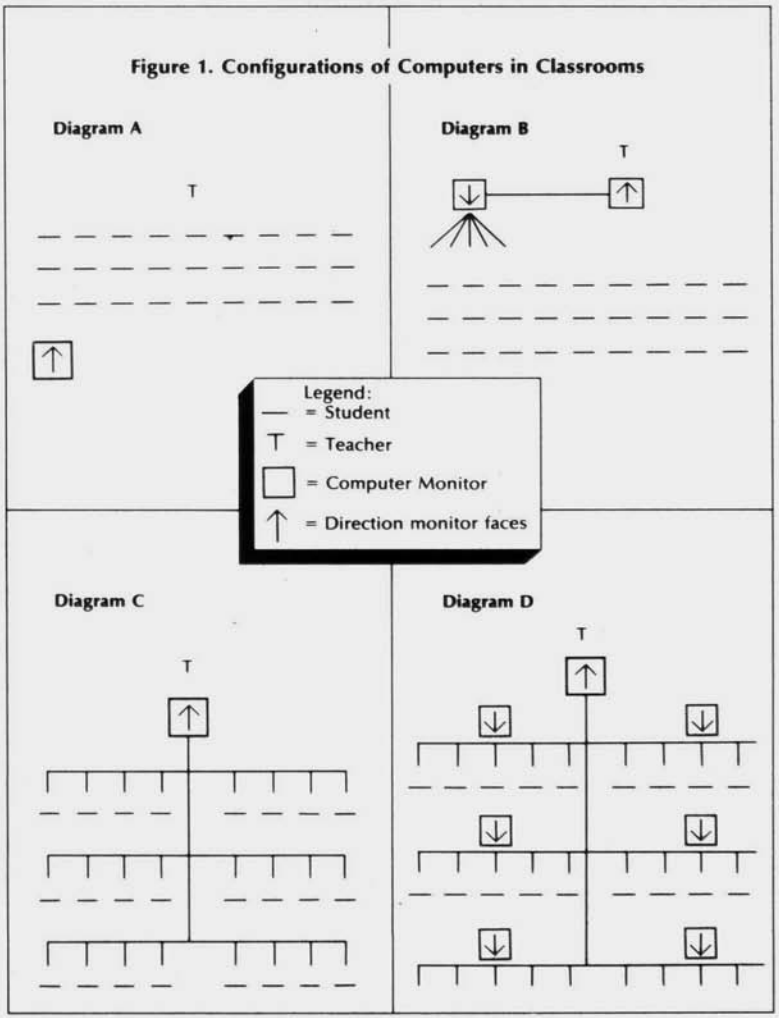
A recently developed software program merges branching and diagnostic testing. Tests are structured so that students first respond to two or three relatively difficult screening items from a given skill or concept domain. The computer screen tells students which item to answer next. If any item is missed, the computer instructs the student to answer a series of diagnostic items, in order to pinpoint specific skill deficiencies. While that student is going through a series of diagnostic items, another student who correctly answered screening items is told by the computer to skip to the set of screening items for the *next* skill. Students move quickly through mastered domains, but receive a thorough assessment in their weak areas.

Teacher Net can also be used with videotapes or disks for teaching complex behaviors such as public speaking, laboratory procedures, or basketball plays. The teacher explains the key behaviors to be learned and a rating system for evaluating them. While a tape is played, each student hits a key when a violation of a basic behavior is observed on the tape. At the end of a tape segment, frames over which students disagreed in their ratings are automatically called up. The teacher and students then discuss reasons for the disagreements.

Some preliminary applications have explored how Teacher Net might facilitite collaborative learning. In one very simple application, university students listen to a statistics lecture on the concept of correlation. Next, a computer screen diagrams examples of different correlations. The teacher then asks the students to approximate a certain correlation by plotting points on the screen. Students take turns entering data points on their key pads. When a student's point that deviates from the assigned correlation appears on the screen, students discuss why it is inappropriate and what other point might be preferable.

More important than the Teacher Net hardware is the concept it exemplifies—an affordable way of increasing teaching productivity by processing student responses with a computer and by providing instruction to groups of students. Although Teacher Net is not suitable for many of the computer's more interesting and powerful applications, like programming or word processing, it does provide teachers with timely information in a usable (summarized) form and can actually increase the amount of instruction provided during the school day.

Figure 1. Configurations of Computers in Classrooms



Quality of Software

When we initiated the Mainstreaming Computers Project, we were puzzled by the vigorous attacks on the quality of computer-assisted instruction (CAI) software. Quality Software (1981) reported, "Only 3–5 percent of the educational programs . . . available are worth looking at." Criticisms of printed curriculum materials are tame by comparison. Publishers of print material are relatively casual about the issue of quality since they assume teachers usually modify curriculum materials to suit their needs.

But teachers cannot modify commercial software. "What is in the CAI program is exactly what gets taught" (Steely, 1981, p. 33). A teacher's role as educational software user is different than the teacher's role as adapter/modifier of print material. Because of this difference, selection procedures are more crucial for CAI software than for print materials. In many instances, textbook committees have based final decisions about print materials on such factors as cost, durability, general educational philosophy, and range of topics covered, and have given little attention to instructional effectiveness. These rather capricious selection practices for print materials have caused many software developers to be "clearly skeptical about whether teachers would appreciate quality [software] if they could get it" (Olds, 1981, p. 20). The Mainstreaming Computers Project responded to the need for quality software in three ways: by specifying criteria for selecting software (Vachon and Carnine, 1983); by developing sample CAI programs (Carnine and others, 1983); and by creating an authoring language for writing software (Engelmann, Carnine, and Weiss, 1983).

1. *Criteria for Evaluating and Selecting Software.* We believe that trained teachers can intelligently evaluate software if they are provided with empirically validated means of rating instructional effectiveness. This is an important component of the Mainstreaming Computers Project. Our work in this area was grounded in a theory of instruction, practical experience in developing instructional materials, and a procedure for field-testing and developing printed material. In *Theory of Instruction*, Engelmann and Carnine (1982) articulate a taxonomy that links instructional objectives to specific instructional design procedures. A taxonomy of instructional objectives is hardly novel, but a taxonomy tied to procedures applicable for designing instructional software is.

To select software, teachers often rely on software evaluation forms or reviews of such forms. In a review of software evaluation forms, Vachon and Carnine (1983) found that five out of seven forms included less than half of the following 11 instructional design dimensions: a clear and logical presentation, clues for key concepts, opportunities for frequent interaction, appropriate difficulty level, effective feedback, additional practice on missed items, learner controls of rate/sequence, learner controls of number of problems, integration of instruction with previous experience, generalizable strategies, and evaluation components. Software could receive favorable ratings on five of the forms and still have serious instructional design flaws.

The criteria for software evaluation and selection (Vachon and Carnine, 1983) relate the key design principles from *Theory of Instruction* to the primary modes of computer-assisted instruction: drill and practice, tutorial, and simulation. One criterion identified by Vachon and Carnine (1983) for *drill and practice* is individualization: programs should tailor the items presented in the computer to each individual's instructional needs. For *tutorials*, which teach rule and concept applications, a criterion is the need to teach explicit, easy-to-follow strategies. Vachon and Carnine's (1983) criteria for *simulations*, the third type of CAI, include capturing an array of interactions among rules and providing activities of graded difficulty to allow all users to understand the nature of the interactions. The summary evaluation form (Vachon and Carnine, 1983), based on ratings of 58 individual criteria, appears in Figure 2. While the criteria are traditional in several respects, the attention paid to instructional design characteristics is somewhat unusual.

Since teachers' inability to modify commerical software makes software selection extremely important, curriculum leaders should be particularly sensitive to problems caused by incomplete evaluation of software. We encourage districts, regional agencies, or state education agencies to share the costs and benefits of systematic procedures for examining software, with an emphasis on instructional design features.

2. *Sample CAI Programs.* Paralleling the evaluation criteria for the three types of CAI is a set of software programs, designed to exemplify the criteria. This need is acknowledged by software producers themselves (Quality Software, 1981). The *Learning Vocabulary* program (Carnine, Granzin, and Rankin, 1983) creates individualized drill and practice lessons composed of exercises involving only the words a particular student doesn't know. The program also keeps track of the words that cause a student particular difficulty and periodically reviews them.

The tutorial *Reasoning Skills I* (Engelmann and Carnine, 1983) teaches

Figure 2. Evaluation Summary Form

Title: _____

CONTENT	Excellent	Average	Marginal	Unsatisfactory
INSTRUCTIONAL DESIGN **1. Objectives** **2. Individualization** **3. Presentation** **4. Feedback** **5. Review** **6. Motivation** **7. Reinforcement**				
PROGRAM UTILITY				
STRENGTHS:				
WEAKNESSES:				
RECOMMENDATIONS:				

students strategies for constructing and critiquing syllogisms, based on a few simple rules and procedures for applying them:

1. For syllogisms using only the word *all* (rather than the words *some* or *not*), each statement names a smaller class and places it in a larger class.

2. The conclusion names the smallest class and places it in the largest class. The middle-sized class isn't named in the conclusion.

Reasoning Skills I teaches that an argument that leads to a contradiction is unsound.

These rules allow students to critique arguments like the following, which has only one premise and a conclusion:

> The sports programs at Acme University are terrible.
>
> Therefore, all the programs at Acme University are terrible.

Using rule 2 leads to this analysis: the first-named class in the conclusion, *programs*, should be the smallest class, and *terrible* the largest class. Thus, *sports program* must be the middle-sized class. However, *programs* is not a smaller class than *sports programs*. This contradiction indicates that the argument is unsound. Slight variations of the strategy can allow students to critique hundreds of syllogisms.

The simulation program *Health Ways* (Carnine, Lang, and Wong, 1983) incorporates over 50 health rules (such as people with diabetes in their family should limit their caloric intake). The simulation also accounts for possible outcomes based on concepts relevant to changing health habits, such as the effort required to make changes, the stress inherent in making changes, and the importance of maintaining changes. Users make decisions in attempting to extend a hypothetical character's expected age to the winning age before "time runs out." To prepare students for the instructional demands of the simulation, a CAI tutorial explains concepts and rules and provides applications of the health rules. Students also go through tutorial versions of the simulation before playing the actual simulation.

Certain practical shortcomings of CAI programs become apparent only when programs are field-tested. The CAI programs developed as part of the Mainstreaming Computers Project were designed not only as models of certain instructional design features but also as protypes for development and field-testing procedures:

1. A team is formed consisting of people with expertise in instructional design, the content area to be covered in the program, and computer programming (if an existing authoring language was not used).

2. An analysis of the content to be covered in the program is conducted under the direction of the team's instructional design expert.

3. The analysis is translated into specific instructional activities appropriate to the computer.

4. A meeting with potential users is held to determine if any revisions are needed.

5. The program is developed.

6. The software program and documentation are field-tested. Trained observers note design, technical, and logistic problems (a lesson is too long, or a newly introduced principle lacks adequate opportunity for application). Teachers are interviewed about the adequacy of the documentation.

7. The program and documentation are revised.

8. The program and documentation are field-tested again. Steps 7 and 8 are often repeated.

9. The program and documentation are revised a final time.

10. The program and documentation are prepared for distribution.

11. Feedback on the program is recorded for subsequent revision.

A simple example of the fruits of field-testing may help bring this rather stodgy 11-step process to life. All CAI programs are field-tested with older educationally handicapped students and with younger nonhandicapped students. *Reasoning Skills I* was field-tested with high school special education students and middle school regular education students. In responding to the preskill instruction on class inclusion, some special education students didn't respond according to class size but rather word length (since the word *roses* is shorter than *flowers*, *roses* is the smaller class). The student arrived at the correct answer, but for the wrong reason. In the revised program the word for the smaller class is not always the shorter word. While this feedback was not profound, it does illustrate the practical, meticulous type of information that can be aggregated to revise and improve CAI programs.

3. *Authoring Software*. With the tradition of teachers modifying print material, creating options for teachers to write or modify software seemed a reasonable component for the Mainstreaming Computers Project. We began by exploring how teachers might go about creating their own software.

It takes a good deal of time for most people to acquire a working knowledge of computer languages, not to mention the problem-solving skills involved in translating instructional tasks into acceptable educational software. Specially designed authoring languages provide an alternative. *Authoring languages* such as Pilot provide a simplified code for creating modules of instruction. The code (or set of commands) is "friendlier" than BASIC or Pascal and more readily understandable to the novice.

An alternative to authoring languages is *authoring systems*. Authoring systems do not require a user to have any programming skills; most are what Kearsley (1982) calls the "prompting" type. The teacher simply answers prompts or questions generated by the system. The authoring system translates the teacher's answers into a CAI lesson.

In evaluating a naive teacher's experience with several authoring languages and systems, Woodward and Carnine (1983) found that the prompts and questions in authoring systems became tiresome, requiring teachers to include certain steps even when the teacher didn't want them. Also, certain important functions, such as reviewing missed items later in a lesson or in the next lesson, were missing in the system and were extremely difficult to program with the languages. Finally, only very specific types of instructional programs, such as tutorials, could be created.

We concluded that if teachers are to write software without becoming computer programmers, authoring languages (or systems) are needed with these characteristics: different languages for different instructional purposes, provision for a transition from the state of novice to experienced user, and inclusion of instructional design features important to effective CAI. The Mainstreaming Computers Project designed DIAL—the Direct Instruction Authoring Language (Engelmann, Carnine, and Weiss, 1983)—to incorporate those features. Instructional design procedures applicable across a range of tasks were included, such as automatically

reviewing difficult items later in a lesson and in the next lesson, and automatically presenting extra practice items to students who have difficulty with a particular explanation. The transition from novice to experienced user was accomplished by providing a tutorial and programmed workbook on how to use DIAL, a summary of DIAL commands, prompted worksheets for writing DIAL lessons, and a multistaged introduction to DIAL commands. At first a new user learns only the essential ten commands for constructing lessons. Later, interested users can learn over 15 additional commands. Even if DIAL turns out to be relatively user friendly, most teachers will not author software. Criteria for selection of software is the key issue.

Realities of Implementation

It is difficult to predict whether the recent infatuation with computers and the enthusiasm of some teacher hobbyists will be enough to make computer technology an integral part of education. Additional hardware and improved software will not in and of themselves move computers from the periphery into the mainstream of American education. Schools have been referred to as cottage industries, in that each classroom is an autonomous unit merely housed in a building with other classrooms. This image underscores the importance of teacher acceptance of computers (or any innovations) if they are to pass beyond the current fad status. Wide-scale teacher resistance to computers can be substantial, because many teachers may not: (1) have a technological orientation, (2) have a clear notion of how computers fit into the school day, (3) see any personal payoff for assuming the added responsibilities that accompany computers, (4) believe that computers will truly improve students' learning.

To enlist teachers' support, it appears that computer applications must:

- Assist teachers with their current work, not just in providing individualized instruction for some students, but also in keeping track of where students are in instruction, in scoring assignments and determining grades, and in monitoring student learning on an ongoing basis.
- Not impose unreasonable demands on teachers' time.
- Not impede social interaction and, in fact, *enhance* it.
- Provide learning experiences that take advantage of the computer's capability, and thus go far beyond what is now possible in classroom instruction.
- Improve learning across the spectrum of student abilities.

These goals can be met through training in:

- Appropriate hardware configurations (Figure 2).
- Software evaluation and selection.
- Procedures for monitoring, testing, grading, and gathering feedback during lectures.
- Incorporation of these applications into the daily schedule.
- Design of print material to accommodate monitoring devices like Teacher Net (material suitable for independent work, tests, and lecture questions).

Administering this training won't be easy or painless. The potential reward, however, is great in terms of student learning.

Computer Complacency?

The Mainstreaming Computers Project is still in its infancy. Its relevance to curriculum leaders is probably more in raising questions to ask rather than in delivering final answers. Questions such as:

- How can we strike a balance between cheaper, lower level computer applications that reach many students and expensive, sophisticated applications that serve fewer students? For example, for which applications will devices like Teacher Net suffice, and which applications will require local area networking?
- How can we help teachers select/develop software that incorporates important instructional design features?
- How can we interest, train, and support teachers in the use of technology, and where should we *concentrate* inservice efforts?

When an idea like computing first emerges, enthusiasm is easy because frailties and limitations are not yet known. Educators must actively seek limitations and determine how to deal with them. The failure of computers in education will not result from applications that are too ambitious, but from applications that are not ambitious enough. Evaluations of the role of computers in schools point to an almost complacent view toward the capability of computers. Sheingold, Kane, and Endreweit (1983) caution that their study "more strikingly illustrates the assimilation of technology by school systems than the impact of technology on them. . . . microcomputers on their own are unlikely to promote any particular outcome" (p. 431). Should computers earn a nitch in the history of education as a mere fad, much will have been lost—the powerful concept of applying technology to schooling will have been discredited.□

References

Abt Associates. *Education as Experimentation: A Planned Variation Model*, Vol. IV. Cambridge, Mass.: Abt Associates, 1977.

Becker, W.C., and Carnine, D.W. "Direct Instruction—An Effective Approach to Education Intervention with Disadvantaged and Low-Performers." In *Advances in Child Psychology: Vol. 3*. Edited by B. Lahey and A. Kazkin. New York: Plenum, 1980.

Carnine, D.; Granzin, A.; and Rankin, G. *Learning Vocabulary*. Eugene, Oreg.: Teaching Wares, 1983.

Carnine, D.; Lang, D.; and Wong, L. *Health Ways*. Eugene, Oreg.: Teaching Wares, 1983.

Engelmann, S., and Carnine, D. *Reasoning Skills I*. Eugene, Oreg.: Engelmann-Becker Corporation, 1983.

Engelmann, S., and Carnine, D. *Theory of Instruction: Principles and Applications*. New York: Irvington Publishers Inc., 1982.

Engelmann, S.; Carnine, D.; and Weiss, A. *Direct Instruction Authoring Language*. Eugene, Oreg.: Engelmann-Becker Corporation, 1983.

Euchner, C. "Students Spend Little Time on Computers." *Education Week*, July 27, 1983.

Gersten, R., and Carnine, D. "The Later Effects of Direct Instruction Follow Through: Preliminary Findings." Paper presented at AERA, Montreal, Quebec, April 1983.

Hofmeister, A. *Microcomputer Instruction Management System*. Logan, Utah: Utah State University, 1982.

Kearsley, G. "Authoring Systems in Computer Based Education." *Communications of the ACM* 25, 7 (1982): 429–437.

Olds, H.F. "The Making of Software." *Classroom Computer News* 8, 10 (1981): 20, 33–34.

Quality Software, *Electronic Learning* 1,2 (1981): 33–36.

Sheingold, K.; Kane, J.H.; and Endreweit, M.E. "Microcomputer Use in Schools: Developing a Research Agenda." *Harvard Educational Review* 53, 4 (1983): 412–432.

Steely, D. "Instructional Design and CAI." *The Computing Teacher* 8, 1 (1981): 33–34.

Vachon, V., and Carnine, D. "Software Evaluation: A Study in Design." Unpublished manuscript. University of Oregon, Eugene, 1983.

Woodward, J., and Carnine, D. "What Do Today's Authoring Languages and Authoring Systems Mean to Today's Educators?" Unpublished manuscript. University of Oregon, Eugene, 1983.