Machine-Mediated Learning, Vol. 2, pp. 239–250
Printed in the UK.

0732-6718/88 $3.00 + .00

The Development of a Software Teaching Tool

JESSE M. HEINES
STUART SMITH

University of Lowell
Department of Computer Science
Lowell, Massachusetts 01854

Abstract *This paper discusses the development of a software teaching tool that teachers can use to create exercises adapted to the needs of their curricula and individual students. It explains how we assessed the teaching needs for a program in adult literacy and describes the tool we developed to present teacher-created exercises on a computer equipped with a voice synthesizer and audio digitizer.*

Teaching Machines vs. Teaching Tools

Any device or technique that facilitates teaching can be referred to as a teaching *tool*. Books are teaching tools, as are chalkboards, movie projectors, writing slates, and even pencils. While some would extend this list to include such devices as hickory sticks, all would agree that the range of devices and techniques that creative teachers can use to expand their capabilities is virtually boundless.

The factor common to successful teaching tools is that they are usable within a teacher's traditional purview, in his or her classroom and under his or her control. None is a teaching *machine*, designed to be used independently of the teacher. Nonetheless, the market is replete with "educational" computer programs that purport to teach without teachers. Many of these programs are indeed excellent examples of instructional technology, yet it is a sad fact that most gather dust in teachers' filing cabinets because they are difficult to integrate into existing curricula. While teacher training is certainly a factor in this dilemma, the programs themselves are also to blame because they operate in a fixed format with a fixed strategy on fixed subject matter. Even teachers who can program are unable to adapt them to their specific classroom needs.

Teaching Tools and Generative CAI

This paper describes a different approach to developing software teaching tools: the development of a tool that allows teachers to build exercises adapted to the needs of their curricula and individual students. The tool we developed can be considered an example of generative CAI in the original sense of the term proposed by Utah [1] in that it uses *algorithms* to produce lessons from high-level lesson descriptions supplied by the teacher. However, unlike such generative systems as "ILIAD" [2], "Latin Skills" [3], "PUNCT2-CW" [4], "MALT" [5], "COMSEQ" [6], and Bales' music dictation system [7]—systems which can produce many different lessons from one set of teacher inputs—this tool produces exactly one lesson from a set of inputs. Thus, the tool represents the "limiting case" of generative CAI.

The essential difference between our tool and generative CAI systems like the others mentioned above lies in the characteristics of the knowledge-bases and the lesson generation/presentation facilities provided. In the other systems, a fixed knowledge-base on a single topic (e.g., English or Latin grammar, punctuation, machine-language programming, digital logic, common practice harmony) is coupled with a lesson generation/presentation program that can produce many exercises on that one topic. In our tool, a knowledge base on any appropriate topic is entered interactively by the teacher. This knowledge-base is coupled with a lesson generation/presentation program that produces a single lesson on the selected topic but with options that allow the teacher to tailor the appearance and behavior of the lesson to meet specific instructional requirements.

The remainder of this paper explains how we assessed the teaching needs for an adult literacy program and describes the tool we developed to present teacher-created exercises on a computer equipped with a voice synthesizer and audio digitizer.

Assessing Teaching Needs

The main school we worked with had an established adult literacy program that was serving approximately 120 students. Many of their instructional materials were created by the teachers, and active teacher involvement in the curriculum was a large part of the school's personality. This school was a participant in a partnership study that provided them with a personal computer laboratory. Each computer in the laboratory was equipped with a voice synthesizer and an audio digitizer so that instructional programs could "speak" to students and "read" them information that appeared on the screen.*

The school had access to a number of high quality, externally-produced computer-assisted instruction (CAI) programs that took advantage of the computer's speech capabilities, but none of these programs fit precisely into their curriculum. We were therefore asked to assess the school's teaching needs and to devise a plan for better use of the computer laboratory in the school's curriculum. We began by talking to the teachers.

The teachers showed us a number of worksheets similar to that in Figure 1, which was used to present the concept of root words and demonstrate how root words may be changed when endings are added. Such presentations led into exercises such as that shown in Figure 2, which helped students practice these skills.

The teachers had also developed a series of exercises that asked students to tell whether adding certain suffixes to specific root words created meaningful words. An example of this type of exercise is shown in Figure 3. Finally, the teachers told us that they also taught "sight words," i.e., words that appear so frequently that good readers recognize them immediately on sight. Examples of sight words are listed in Figure 4. These figures represent but a very small sample of the materials the teachers had developed.

The teaching need, then, was *not* for another CAI program, but for a tool that teachers could use to implement their *own* exercises on the computer systems. The tool needed to be easy to use, of course, but it also needed to be flexible enough to allow teachers to create a large variety of exercises. As the reader will see in the ensuing

*The software described in this paper is in use at several test sites. However, it is not yet available for acquisition by other individuals or educational institutions. We apologize for not being able to report the names of the schools we worked with or the company that provided the computer laboratory, but the company made such anonymity a condition of our being able to publish this work.

Root Word	Root Word + ending	Root Word + ending
1. silly	silliest	
2. tiny	tinier	
3. crazy	crazier	
4. dizzy	dizziness	
5. brain	brainiest	
6. taste	tasty	tastier
7. luck	lucky	luckiest
8. sleep	sleepy	sleepiness
9. destroy	destroyer	
10. copy	copier	copying
11. dry	drier	drying
12. glory	glorious	

Figure 1. A teacher-made sheet for presenting the concept of root words.

discussions, the flexibililty of the resultant tool actually stimulated teachers to create new types of exercises that they had not previously envisioned.

Designing the Tool

Review of the materials created by the teachers revealed a consistent pattern. With a few exceptions, the exercises could be set up as matrices—two-dimensional tables—with letters or words along the left and top axes and the result of combining these letters or words in the corresponding matrix cells. This pattern provided us with a powerful yet easily understood instructional paradigm. We therefore began the design of our tool based on the construction of word matrices, and we determined what information teachers needed to provide for the computer to build a word matrix exercise.

The basic components of the matrix are the axis elements and the contents of the intersecting cells. Since the axis elements could be any characters, we designed the "what-you-see-is-what-you-get" (WYSIWYG) editor shown in Figure 5 (the ▮ character represents the cursor). To enter an element along either axis, teachers used the arrow keys to position the cursor in the appropriate axis cell and then typed the characters to appear in that cell. A new row or column of cells could be created by positioning the cursor on the line between two rows or columns and pressing the Ins key. As new rows and columns were created, the screen was redrawn. Columns expanded and contracted automatically as characters were typed and deleted. Additional editing options are shown in the KEY/ACTION Help Window in Figure 5.

To specify cell contents, we created the second WYSIWYG editor shown in Figure 6.

DIRECTIONS - Add endings to the root word. Remember that in all cases you must first drop the t, add an i, and then add the ending.

Example: happy - happier, happiest, happiness

Root Word	er	est	ness
1. funny			
2. sunny			
3. needy			
4. nutty			
5. lazy			
6. ready			

Figure 2. A teacher-made exercise on root words.

(In this figure, the box in the first cell represents the reverse video cursor.) The numbers at the bottom of the screen represent the rules for forming cell contents. In Figure 6, Rule 2 has been applied to all cells in the matrix. This rule says that the cell contents are formed by doubling the last letter of the characters at the side and appending the characters at the top. This rule is appropriate for all cells except the first, which requires Rule 11: change the last non-word-terminating vowel in the characters at the side to *a* and do not append the characters at the top. The complete set of *grammatical* cell formation rules is as follows:

(1) add the characters at the top to the characters at the side (side + top), e.g., "sing" at the side + "ing" at the top forms "singing"
(2) double the last letter of the characters at the side and add the characters at the top, e.g., "run" + "ing" forms "running" and "chap" + "ed" forms "chapped"
(3) drop the last letter of the characters at the side and add the characters at the top, e.g., "take" + "ing" forms "taking," and note that this rule also allows "full" + "ly" to form "fully" and "face" + "ed" to form "faced"
(4) add *e* to the characters at the side and then add the characters at the top, e.g., "class" + "s" forms "classes"

	Can you add -less to the root word?	Can you add -ful to the root word?	Add -ness to the two forms of words you have just made.	
1. care				
2. use				
3. help				
4. fear				
5. pain				
6. color				
7. harm				
8. need				
9. form				
10. hate				
11. love				
12. weight				

Figure 3. A teacher-made exercise on suffixes.

(5) drop the last letter of the characters at the side, add *i*, and then add the characters at the top, e.g., "happy" + "ly" forms "happily"
(6) drop the last letter of the characters at the side, add *ie*, and then add the characters at the top, e.g., "try" + "s" forms "tries"

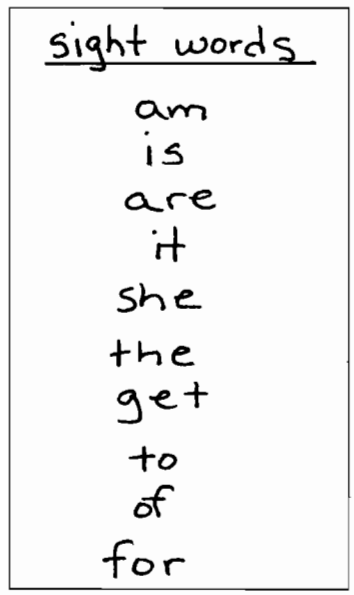

Figure 4. A teacher-made sheet for presenting sight words.

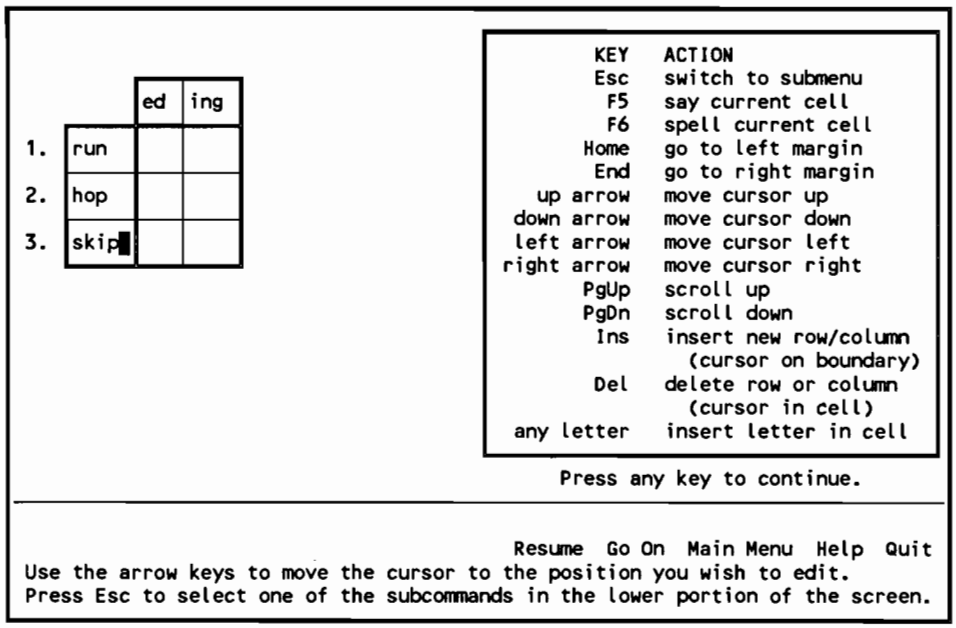

Figure 5. What-you-see-is-what-you-get axis element editor and its help window.

(7) drop the last letter of the characters at the side, add *ve*, and then add the characters at the top, e.g., "leaf" + "s" forms "leaves"
(8) drop the last two letters of the characters at the side, add *ve*, and then add the characters at the top, e.g., "life" + "s" forms "lives"

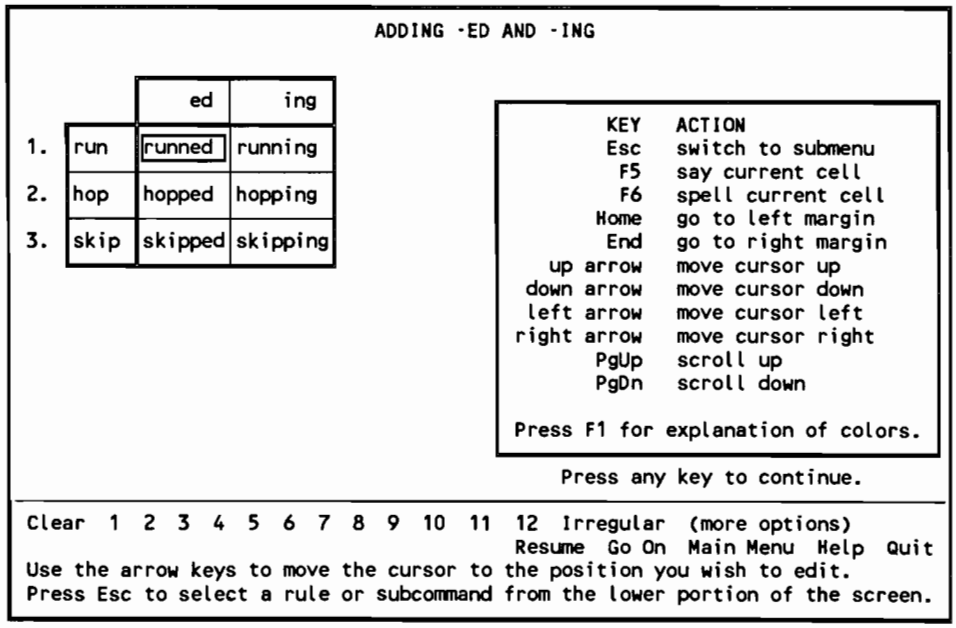

Figure 6. WYSIWYG cell contents editor and its help window.

(9) use only the characters at the side without appending the characters at the top, e.g., "sheep" + "s" forms "sheep"
(10) change the last non-word-terminating vowel in the characters at the side to *e* and do not append the characters at the top, e.g., "blow" + "ed" forms "blew"
(11) change the last non-word-terminating vowel in the characters at the side to *a* and do not append the characters at the top, e.g., "become" + "ed" forms "became" and "run" + "ed" forms "ran"
(12) add the characters at the side to the characters at the top (top + side)—this rule is used for prefixes

One very important rule that appears in Figure 6 is missing from the above list: the "Irregular" rule. This rule allowed teachers to specify the contents of a cell directly rather than through the use of a pre-programmed rule, thus enabling them to make matrices for virtually any subject matter. Consider, for example, a matrix in which students are to give the results of mixing primary colors. Although we have no rule for "blue + yellow = green," a matrix containing this rule could still be built by highlighting the cell at which the elements "blue" and "yellow" intersect, selecting the "Irregular" rule, and then typing "green."

These two WYSIWYG editors, together with an overall menu system that allowed teachers to move from one editor to another, were the basic components of our initial design. We also built a program for students which presented matrix exercises that teachers built. As teachers began to use the tool, however, their creativity spawned requests for more and more options and greater control of the matrix presentation strategies. For example, at teachers' requests we implemented the following *arithmetic* cell formation rules:

S + T	add the number at the top to the number at the side (integer addition)
T − S	subtract the number at the side from the number at the top (integer subtraction)
S − T	subtract the number at the top from the number at the side (integer subtraction)
S * T	multiply the number at the side by the number at the top (integer multiplication)
S/T	divide the number at the side by the number at the top and return both the quotient and remainder
T/S	divide the number at the top by the number at the side and return both the quotient and remainder

The next section recounts how we expanded our original design to meet needs that teachers identified as they worked with the tool.

Expanding the Tool

The matrix paradigm turned out to be extemely flexible, especially after options were added to control how the matrix was displayed and what the computer did as students moved from cell to cell. Matrix display was controlled by the Layout options shown in Figure 7. Teachers selected options by moving a highlighting bar (indicated by the box in Figure 7) to scroll through the choices and pressing RETURN to make a selection. For example, moving the highlighting bar to "Grid color" and pressing RETURN caused

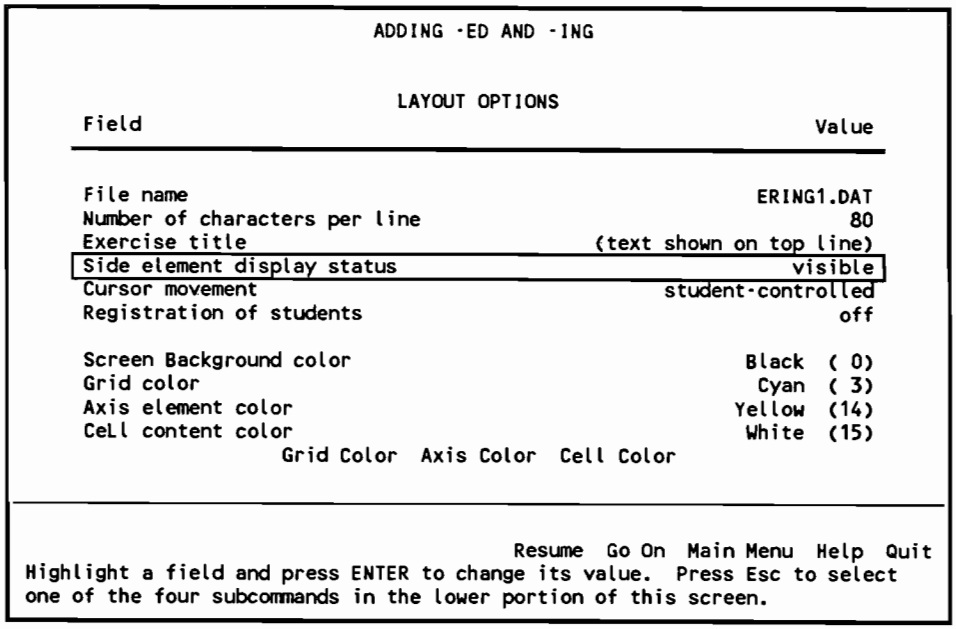

Figure 7. Exercise layout options menu.

"Cyan (3)" to change to "Red (4)" with a corresponding change in the representative color bar just below the color options. For options without fixed choices, like "File Name" and "Exercise title," teachers would be prompted to enter a string and press RETURN.

At the teachers' suggestions, we added the two options "Side element display status" and "Cursor movement." Side element display status could be either "visible," the normal case, or "hidden," to accommodate sight word exercises. In the latter case, teachers could specify words as side axis elements, but they wouldn't appear on the screen. Students could then hear these words spoken by the voice synthesizer and practice spelling them without the benefit of any on-screen cues to the right answer. For the cursor movement option, we allowed teachers to specify whether movement of the cursor to the cell to be filled in next was to be under program or student control. We had originally programmed cursor movement to be under student control, but our interaction with the teachers indicated that it was sometimes preferable for the cursor to move automatically to the next cell after the student had responded. We therefore gave the teachers the options "student-controlled," where students could use the arrow keys to move the cursor, "auto-horizontal," where the cursor moved automatically to the next cell to the right, and "auto-vertical," where it automatically moved to the next cell down.

What the computer did as students moved from cell to cell was determined by the Actions options shown in Figure 8. Virtually all of these options were the result of our interactions with teachers as they used the word matrix development tools. We had "hard-coded" one version of many of the actions listed in Figure 8, but teachers indicated that they needed alternate actions and that they needed to be able to change the matrix actions for different exercises. Thus, for example, we allowed teachers to select one of ten actions for the program to take when a cell was highlighted:

- don't do anything

```
                          ACTION OPTIONS
Field                                                              Value

Action to take when cell is highlighted                  don't do anything
Wait for student to enter cell response?                               yes
Text to say for correct answers                    "Yes, that is correct."
TExt prefix to say for incorrect answers         "No, that is not correct."
TeXt suffix to say for 1st incorrect answer    "Please try again.  You ..."
Text Suffix to say for 2nd incorrect answer    "Please try again.  You ..."
Present "cell group" query?                                             no
Cell group choice                                                      Y/N
PRompt for "cell group" query          "Is this box correct?  Yes or No:  "
Text tO say for correct Y response               "Yes, this box is correct."
Text to saY for correct N response             "Yes, this box is not correct."
Text to say For incorrect Y response            "No, this box is not correct."
Text to say for Incorrect N response               "No, this box is correct."


                                  Resume  Go On  Main Menu  Help  Quit
Highlight a field and press ENTER to change its value.  Press Esc to select
one of the four subcommands in the lower portion of this screen.
```

Figure 8. Exercise actions options menu.

- say the correct response
- spell the correct response
- say and spell the correct response
- say the side element
- spell the side element
- say and spell the side element
- say the top element
- spell the top element
- say and spell the top element

Because teachers also wanted to be able to set up "demonstration" exercises, we provided a way for the program to fill in the answers for cells automatically. This feature was selected by changing the "Wait for student to enter cell response?" option to "no." Likewise, we made all of the synthesized voice messages into variables so that teachers could change them.

One of the biggest program enhancements that teachers requested was the addition of a second query that could appear *after* the cell had been filled in. This feature was selected by changing the 'Present "cell group" query?' option to "yes." (This option is boxed in Figure 8.) We originally designed this feature to accommodate exercises such as that shown in Figure 3, where the program asked "Is this a real word?" and the student would respond "yes" or "no." Teachers came up with so many different ways to use this feature, however, that we implemented several "cell group" choices: yes/no, true/false, A/B, and 1/2. Of course, this change meant that the prompt for the cell group query as well as the text that the program would say for correct and incorrect responses had to be under teacher control. While there were defaults for the prompt and for all of the texts, these could be changed by selecting the appropriate options in Figure 8.

The final major component of the word matrix development tool was a facility for

creating digitized spoken messages. Once again we provided a menu-driven system. Teachers selected the message they wanted to work with and then specified whether they wanted to change, hear, or clear (delete) that message. If they wanted to change it, they could select the system default message to be "spoken" by the voice synthesizer, type their own message to be "spoken" by the synthesizer, or record a message by speaking through a microphone connected to the audio digitizer. Digitized speech was played back through the same audio output facility used by the speech synthesizer. Figure 9 shows the screen during the recording of a message. The third-level submenu is at the bottom of the screen.

Our original design called for teachers to be able to record only the messages to be read when students begin and end an exercise. Work with the teachers revealed the need for a message to be read when students terminated the exercise before they had completed it. In addition, the teachers wanted to be able to change the help messages for all of the cell formation rules. Therefore, we made these available as well.

As the creation of matrices became more complicated, teachers identified the need for a facility to make sure that all matrix parameters were specified and that they were logically consistent. Consequently, we implemented a check on the integrity of the matrix as a top level command of the word matrix development tool. In addition to these major facilities, the tool had facilities to store and recall exercises, to allow teachers to try exercises, and to review data gathered during student use of the exercises.

Making the Tool Truly Generative

The word matrix tool could be modified to do generative CAI in the usual sense. The two major components of the knowledge-base of any generative CAI system—a dictionary of facts and a set of rules for manipulating the facts—are managed in the word

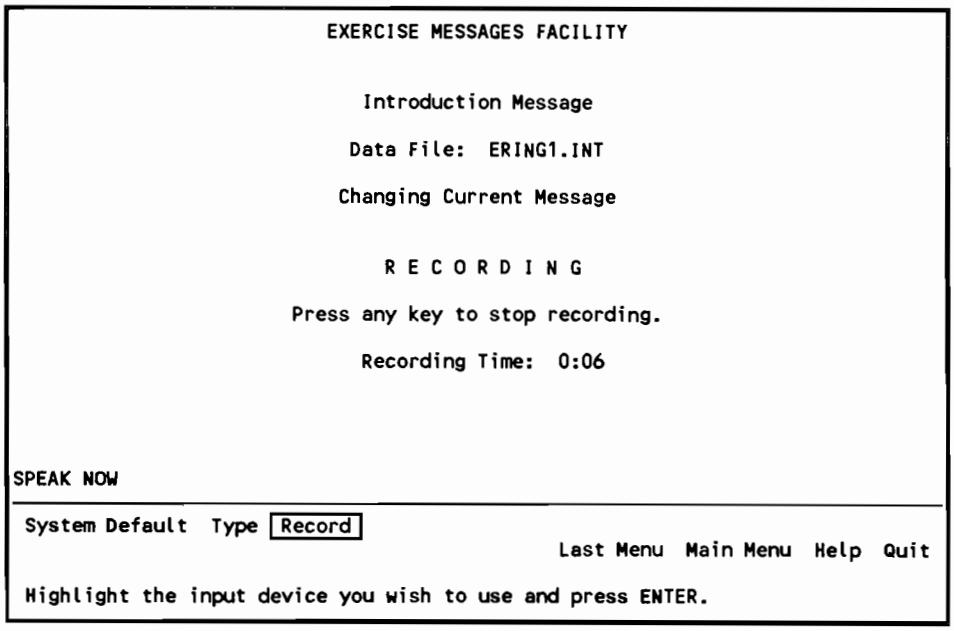

Figure 9. Exercise message recording facility.

matrix tool by semi-independent modules. Although each module obtains its component of the knowledge-base interactively from the teacher, there is no reason why these modules could not be modified to select facts and rules from a pre-stored knowledge-base. With this modified word matrix tool a teacher could, for example, create many spelling-bee type exercises by:

(1) requesting that the system randomly select the desired number of words from its dictionary
(2) specifying that each word be "spoken" by the speech synthesizer but not displayed to the left of the corresponding answer box on the screen (this behavior is already an available option in the word matrix tool)

Arithmetic drills could be generated in a similar fashion by having the word matrix tool apply the rules of integer arithmetic, which are already part of its knowledge-base, to sets of random integers.

Teachers' Reactions

The word matrix tool was used by teachers during the 1987–1988 winter months in adult literacy programs in New York and Philadelphia. One teacher created over 30 exercises, while others generally created one or two. The range of exercises that teachers created was quite large. One teacher created an exercise on the names of metric units (centimeter, decimeter, etc.), where students simply moved the cursor from one cell to another to hear the computer "read" the names of the metric units. Another teacher created an exercise on binary numbers with powers of 2 across the horizontal axis and decimal numbers along the vertical axis. Students indicated the binary equivalent of the decimal number by entering 1 or 0 in each cell. Teachers even tried to create multi-language exercises, such as ones in which Spanish words appeared along the vertical axis and students were to enter the equivalent English translations. They reported that Spanish to English worked well since the computer was generally asked to speak the word typed in English, but English to Spanish was not as effective, because when the computer was asked to speak a Spanish word students reported that it sounded like a "gringo!"

Teachers in both programs pointed out the need for additional teacher training. We discovered that even teachers who are computer-literate need help in devising ways to link computer-delivered exercises with their classroom lessons. Not surprisingly, the coordinator of one program told us that the greatest factor in the success or failure of applying the word matrix tool was teacher enthusiasm. Classes of some teachers didn't want to use the computers at all, she reported, while classes of others didn't want to stop using them.

Both adult literacy programs that tested the word matrix tool are staffed by part-time as well as full-time teachers. For this reason, it was often days or even weeks between the times that teachers could find to use the word matrix tool. During these delays teachers of course had trouble remembering the many aspects of the tool, but they told us that relearning time was virtually eliminated by the clarity of the hierarchical menu structure. Teachers at both sites reported that it took time to "get into" the tool, but that once they had learned to use it effectively, the hierarchical program structure allowed them to go a long time without using it and still take over virtually from where they left off whenever they decided to return. "If you call some of the other programs we use around here 'user-friendly,' " one teacher told us, "you'd have to call this program 'user-lovable!' "

Learning From Our Experience

Many developers have written about the need for an iterative design-implement-test cycle in software development. When working with teachers, the need for such a cycle appears to be even more critical, because many teachers cannot foresee the full potential of a software tool until they begin to work with it. Likewise, it is difficult for developers to foresee the myriad ways in which a software tool might be used until it is integrated into the classroom environment. A close relationship between teacher and developer is critical.

We believe that the success of this project was rooted in its emphasis on the development of a flexible teaching *tool* that teachers could easily integrate into their curricula, rather than a teaching *machine* with fixed CAI applications. We also had the luxury of working with teachers who were experts in their fields, highly receptive to using CAI in their classrooms, and truly creative in developing matrix exercises.

References

1. Uttal, W., T. Pasich, M. Rogers, and R. Hieronymus. 1969. *Generative computer assisted instruction*. Communications 243. Mental Health Research Institute, University of Michigan, Ann Arbor, MI.
2. Bates, Madeleine, Jack Beinashowitz, Robert Ingria, and Kirk Wilson. 1981. *Generative tutorial systems*. Proceedings of the 1981 Conference of the Association for the Development of Computer-based Instructional Systems (ADCIS), 12–21. Bellingham, WA.
3. Culley, Gerald R. 1987. *Generative CAI for Latin: assessing a ten-year project*. Proceedings of the 29th ADCIS Conference, 229–233. Bellingham, WA.
4. Freed, Michele M. 1970. *Generation of punctuation and usage exercises in freshman English using a sentence pool*. PUNCT2-CW Technical Report No. 6 ERIC Document No. ED 084 880. Alexandria, VA.
5. Blount, Sumner E. 1972. *A generative CAI monitor for teaching machine-language programming*. ERIC Document No. ED 078 647. Alexandria, VA.
6. Koffman, Elliot, Sumner Blount, Thomas Gilkey, James Perry, and Martin Wei. 1972. *An intelligent CAI monitor and generative tutor: an interim report*. ERIC Document No. ED 078 681. Alexandria, VA.
7. Bales, W. Kenton. 1980. *A model for generative harmonic dictation*. Educational Resources Information Center (ERIC) Document No. ED 194 057. Alexandria, VA.