ED 247 609	CS 208 554
AUTHOR	Hansen, Craig; Wilcox, Lance
TITLE	Adapting Microcomputers for Use in College Composition Courses.
PUB DATE	25 Feb 84
NOTE	11p.; Paper presented at the Spring Conference of the Delaware Valley Writing Council and Villanova University's English Department (Villanova, PA, February 25, 1984).
PUB TYPE	Reports - Descriptive (141) -- Speeches/Conference Papers (150)
EDRS PRICE	MF01/PC01 Plus Postage.
DESCRIPTORS	*Computer Assisted Instruction; *Computer Software; Higher Education; *Microcomputers; Programing; Word Processing; *Writing Exercises; *Writing Instruction; *Writing Processes
IDENTIFIERS	*Software Evaluation

ABSTRACT

Generally, there are three kinds of software that could be used in college writing courses: word processing programs, interactive questionnaires, and text parsers. Although the benefits of these programs are well known, they still pose some problems. Word processors, for example, have been designed to "process" existing text; few have been designed to meet the needs of online composing. The problem with interactive questionnaires is that the text is fixed, while the problem with text parsers is their inability to analyze content. A software package that hopes to solve these problems and provide the teacher with the means to create computer based aids, prompts, and exercises is A Computer Composing Educational Software System (ACCESS). Among its features are a number of exercises already designed and written by and for composition teachers. Teachers might borrow an exercise and adapt it to a particular class and students. Or, they might create exercises such as entering sample texts that students could react to in writing. When teachers sit down to create such exercises, they can choose from nested menus. The first menu, for instance, offers exercises for prewriting, drafting, or revision aids, or drill and practice. Although ACCESS is in large part already programed, it still awaits debugging and testing. (HOD)

Adapting Microcomputers for use in College Composition Courses

by
Craig Hansen and Lance Wilcox
University of Minnesota
Program in Composition and Communication
Minneapolis, Minnesota

Supported by a grant from the Fund for the Improvement of Post-Secondary Education (Lillian Bridwell and Donald Ross, co-principal investigators), a research team at the Program in Composition and Communication at the University of Minnesota has been examining the uses of microcomputers in college writing courses. Our investigation, currently in its second year, has consisted of two parts: first, the reviewing of commercial and experimental software relevant to writing courses; and second, the development of new software to meet the needs of our own curriculum. In this article we will briefly summarize our analyses of existing programs and then describe the software system we now have under development.

From our experience in testing and reviewing software, we have found it difficult to make any kind of evaluation without specific applications in mind. In our own case, we are working within a writing program that is strongly process based. That is, we spend most of our time teaching students how to develop, organize, and present their ideas, rather than focusing on the grammar and mechanics of the final written product. Most of our instructors use some version of the Flower and Hayes problem-solving approach to writing. Furthermore, the instructors enjoy great freedom in how they organize their writing courses and in the material and assignments they present. Thus, for software to meet our needs, it must be amenable to teaching writing as a process and be adaptable to our instructors' various approaches.

Very generally, we reviewed three kinds of software that could be used in writing courses. These were word processing programs, interactive questionnaires, and text parsers. We found that there were potential benefits and problems with all three. For the sake of brevity, and since the benefits of these have been well documented, we will focus on the problems and then offer our own solution for incorporating microcomputers into a writing curriculum.

Word Processors. Most word processing systems (including all commercially available ones) have been designed to "process" existing text. This includes various formatting procedures, small scale revisions, and the insertion of new material into an already existing document. The command structure of the typical word processor reflects these priorities. Few have been designed to meet the needs of on-line composing. As ~~a~~ consequence, students find their ability to make their writing look professional very seductive: the most intellectually impoverished essay looks just as good as the one with meaningful content. Furthermore, even students who have learned to make multiple drafts of a paper will be tempted to fall back into the "perfect first draft" syndrome. It is simply too easy to fuss with every word, every idea. This does not affect every student in the same way, but the potential problem does exist.

Realizing that most word processors have not been designed to help student writers compose is the first step toward using them in a pedagogically enlightened way. The instructor cannot remain passive and let the students figure out for themselves how they will write on the machines. Rather, the instructor must actively teach students how best to use these programs; they should particularly encourage students to learn and to use those commands that allow revision on a more global scale (e.g., search and replace, block moves, "reading in" files).

Interactive Questionnaires. Interactive questionnaires allow students to create text in response to leading questions, usually to aid in prewriting. Conceptually, these represent perhaps the most enlightened use of computers in writing instruction. They can successfully model for students question/answer heuristics for zeroing in on a topic, as well as helping the student develop useful information about the topic or intended audience. These too, however, have their limitations.

The problem is that interactive questionnaires, like virtually all software, come as static packages: the text is fixed. Our idea of ten questions that help students begin a paper may not be your idea of ten such questions; but if you buy our questionnaire program, our ten questions are the ones you've got, and there is no way you can change them. Furthermore, our ten questions will necessarily have to be very general, so that they will be relevant to the largest number of paper topics possible. They may show students how to apply questions to begin a paper, but in all likelihood will not serve them as a useful resource time after time.

However, if you, as the individual writing instructor, could enter your own questions or text into one of these (very clever) programs, these problems would disappear. (More on that later.)

Text Parsers. Text parsers are programs that use the computer for what it's best at: counting and categorizing. These programs compare the characters in a text file to those in a data base, and if they match, they trigger certain functions or screen displays. To the student, the computer has found all of her or his passives, fuzzy words, abstract nouns, or missing periods. Unfortunately, the program will usually flag anything it finds in a negative or ambiguous way, like "you may have too many passive verbs," or "your sentences may be too long."

We have found that our graduate students make good use of these programs: they are good writers and make few mistakes. What the computer finds is usually a legitimate problem (most often a typo). But for less experienced writers, text parsers can be intimidating. One program we looked at furnished 6 pages of analysis (much of it statistical) for three pages of text. This is probably not a terribly constructive experience for students. The computer becomes an arbiter rather than a tool, though it is very poorly equipped at this stage to judge anything. The simple truth is that no program can analyze content. Additionally, such parsers stress the surface features of the text, which works against viewing writing as a process. Unless used with great care, we are not optimistic about the contribution text parsers have to make to writing instruction.

Again, one of the major problems we found in using any of these software packages was, the programs' inflexibility. Whether the programs were interactive "mind-picking" questionnaires for use in prewriting, stand-alone drill-and-practice exercises, or text parsers of one sort or another, we always felt the same frustration. We could not get in there and tamper with the programs. We could not add or delete particular questions. We could not get the "fuzzy word" checkers to highlight our students' fuzzy words. We could not get the brain-storming exercises to conjure up the right storms, or even blow them in what we thought the best direction. Nor, so far as we could tell, was this problem being addressed by designers and producers of educational software, with few exceptions.

In considering these problems, we soon realized that even the ability to adapt existing programs was still only a half-way measure. What we finally needed was a software system that would allow the teacher without any previous training in computer lore to write such exercises for his or her own particular students.

In the past, this capability has been available, albeit clumsily, though "authoring systems" and CAI (Computer Assisted Instruction) languages. Both of these approaches, however, are severely restricted in their usefulness for most instructors.

"Authoring systems" provide a teacher a particular instructional format, such as passages of explanation, true-false or multiple-choice questions, and allow the teacher to "pour in" whatever content she chooses. The better systems also allow the teacher to explain all possible answers to the questions, both right and wrong. Their primary use has been in developing training materials for employees in private corporations. For drill-and-practice exercises, they may be useful enough; but in teaching writing, the sorts of exercises an authoring system makes available are rarely very helpful. CAI languages are programming languages that have been stripped down and simplified for ease of learning. They still require, however, more time to master than most teachers will care to invest; and their very "simplification" inevitably restricts the range of instructional programs that can be written with them.

We are presently working on a software package we hope will finesse these problems and provide the teacher the means to create computer-based aids, prompts, and exercises for his students, while requiring no more training than would suffice to learn a word processor.

We call this system ACCESS. The acronym stands for A Computer Composing Educational Software System. We use the phrase "software system" to discriminate ACCESS from the sorts of "authoring systems" described above. It will in fact have most of the capabilities of an authoring system, but these will constitute only a fraction of its powers. The phrase "Computer Composing" indicates the instructional needs we are particularly concerned to address. Learning to write means producing prose, reading prose, adding prose, removing prose, rearranging prose, and throwing it all away and starting over. Any computer-based instructional materials intended to help this process along will, at the very least, require text-handling capacities of some sophistication. Fixed format questions (e.g., true/false or multiple choice) will not fill the bill.

Again, in trying to understand the purpose and function of ACCESS, you need to bear in mind the final "product" we are aiming at as teachers of writing. We already use computers in our courses, though so far only for word processing. We want our students to be able to sit down at the computer and, in the process of working through pre-writing prompts, drafting, and revision exercises, develop a paper (on disk) ready to be printed out and handed in. We want our exercises and prompts under the same umbrella program as our word processor, so that, if the student chooses, he or she need never work away from the terminal. And again, we want the teacher to be able to design and "program" these exercises herself.

When the teacher sits down to create such exercises, he will be presented with a series of choices in the form of nested menus. The first menu, for instance, will offer the teacher the choice of working on, roughly, pre-writing, drafting, or revision aids, or "stand-alone" drill-and-practice exercises. The teacher then decides where in the writing process he intends to intervene through computerized instruction.

The teacher then chooses among exercises within that class of varying degrees of "prefabrication," or moves directly to create an exercise from scratch. Among its other features ACCESS will include a number of exercises already designed and written by and for composition teachers. If, after reviewing them, the teacher decides that one of them will satisfy her own pedagogical purposes exactly, she can simply appropriate it as it stands. ACCESS will record the exercise onto a separate disk; and students can be instructed to use that disk, call up the particular exercise required, and work through it. This is the simplest approach the teacher could take with ACCESS; its closest analog in using printed curricular materials would be photocopying the exercises in somebody else's textbook and distributing them for classroom use just as they are.

A second option would be borrowing an exercise, but adapting it for your own particular class and students. Any of the demonstration exercises accompanying ACCESS the teacher will be able to amend at will. Whether the exercise is a series of leading questions, interactive items, style checkers, or simply instructions to be displayed on screen while the writer is working, ACCESS will allow the teacher to add, delete, or revise however she wishes.

A third option is for a teacher to create exercises. She might choose and enter sample texts that students could react to in writing. Or she might enter a set of passive-voice sentences with instructions on how to rewrite them as active-voice. Here, in fact, if the teacher so desired, she could select, as with a standard authoring system, a straightforward true/false or multiple choice question format and pour in the material to be tested. Both the structure and material of the exercise could be altered at any point.

From the teacher's perspective, the operation would look something like as follows:

1) Enter a name for the students to call the exercise up by.

2) Enter some kind of opening overview. This can be either a text you type in with the ACCESS editor, or a menu presenting various options to the prospective student writer.

3 A) If the teacher so chooses, the actual exercise can be a simple linear sequence of various texts, instructions, questions, and opportunities for the student to respond with written answers of (virtually) any length. Depending on how the teacher arranges the exercise, the student may be compelled to answer in order every question on it, or she might have the option of paging back and forth over the material, responding as she is moved to do so.

3 B) If the teacher introduces a "menu," he will have to specify what activities accompany each choice. These could themselves be linear sequences of the sort described above, or could lead to further menus yet.

4) Finally, each sequence would end with either the student returning to the opening menu or logging off and taking a print- out of the exercise and her answers for future reference. The student's answers would also be available for development into a formal text, using a word processor on a compatible operating system.

Such, at any rate, is our goal. At present, ACCESS is in large part already programmed, though it still awaits debugging and testing. A prototype should be up and running by Fall of

1984 with, perhaps, a final version - complete with exercises - available by mid-1985. This particular version will be capable of most of the applications we have been describing here, though building in the text parsers and true/false, multiple-choice question formats will require a little longer yet.