Authoring System for Courseware Development: What Should Beginners Look For?

Carol Chapelle
Iowa State University, carolc@iastate.edu

Joan Jamieson
Northern Arizona University

Authoring System for Courseware Development: What Should Beginners Look For?

Abstract
This paper is an introduction to courseware authoring systems, which are defined as software devised for an instructional environment that simplifies courseware development. The process of courseware authoring is described as consisting of two basic elements: computer and lesson. These two elements are further subdivided and the use of authoring systems to simplify each of these subcomponents is described. Suggestions are made for educators who are considering authoring systems to be used to develop courseware.

Keywords
courseware authoring systems, evaluation of, description of

Disciplines
Bilingual, Multilingual, and Multicultural Education | Curriculum and Instruction | Educational Methods | Instructional Media Design

Comments
This article is from *CALICO Journal* 3 (1986): 14, doi:10.1558/cj.v3i3.14-19. Posted with permission.

AUTHORING SYSTEMS FOR COURSEWARE DEVELOPMENT: WHAT SHOULD BEGINNERS LOOK FOR?

Carol Chapelle and Joan Jamieson

ABSTRACT

This paper is an introduction to courseware authoring systems, which are defined as software devised for an instructional environment that simplifies courseware development. The process of courseware authoring is described as consisting of two basic elements: computer and lesson. These two elements are further subdivided and the use of authoring systems to simplify each of these subcomponents is described. Suggestions are made for educators who are considering authoring systems to be used to develop courseware.

KEYWORDS: courseware authoring systems; evaluation of, description of.

Many language teachers have the opportunity to use their schools' microcomputers and would like to provide their students with computer-assisted learning activities. Yet, access to the computer alone is insufficient; teachers must also buy (or lease) ready-made lessons for the computer or write original materials. There is a variety of commercially available courseware for language instruction (see, for example, Wyatt, 1984; Jamieson and Chapelle, 1984). Even though these numerous lessons are available, the ones that will actually be usable on a given computer account for a much smaller number. Generally speaking, software today is machine specific so that if a lesson is purchased for a Commodore 64, for example, it will typically not run on an Apple IIe. Even when the right lesson-machine match is made, the lesson may not be appropriate to the needs of the students so that the teacher may want to modify or discard it. As a result, many educators are looking into the possibility of writing their own materials in order to obtain appropriate courseware.

Writing courseware is an alternative to using ready-made lessons; yet, this approach is riddled with its own problems. Teachers are usually hired to teach, not to develop courseware. Consequently, because courseware development time must be kept to a minimum, it is very difficult for teachers to produce the kinds of materials they would really like their students to use. A second and more serious stumbling block is that language teachers seldom have the skills needed to produce the courseware they want.

Authoring systems have been developed to help teachers to overcome the problems of insufficient time and skill (Holmes, 1983; Wyatt, 1983). Some authoring systems, called templates, provide a very structured framework into which a teacher need only type the text. For example, a template system might provide the teacher with a multiple choice test program. The teacher could spend twenty minutes typing in his own questions, alternatives and answers and a class would have an appropriate on-line test for the next day. Other authoring systems might provide the teacher with tools, such as an instructional programming language, to make courseware writing easier. Using these tools, the teacher can develop materials that perform sophisticated answer judging, for example. Such lessons would have otherwise been beyond the bounds of his technical expertise. These examples illustrate two different kinds of authoring systems, a term which will be defined here an any software, devised for an instructional environment, that simplifies the development of courseware.

Authoring systems are available on many different types of computers including mainframes (such as PLATO at the University of Illinois), mini's (such as the VAX at Iowa State University) and microcomputers (such as Apple IIe's in language laboratories all over the world). On mainframe and minicomputers, an authoring system is typically accessed through the use of commands typed in at a terminal. An authoring system for a microcomputer, on the other hand, comes on floppy disks that are bought at a computer store or from a publisher or a consortium. The software included in the package frequently consists of an instructional sequence designed to instruct the teacher how to use the system. In addition to this on-line documentation, the package will also include a set of manuals that describe the system and how to use it.

The purpose of this paper is to clarify the ways in which authoring systems simplify courseware writing. It will not attempt to review the numerous authoring systems whose advertisements fill the pages of magazines and educational computing journals (see also, Jones, 1984). Instead, this paper will provide educators with a framework for analyzing the authoring systems they find.

COURSEWARE WRITING

In order to understand how an authoring system can simplify courseware writing, it is necessary to define the elements of courseware creation as illustrated in Figure 1. In the process of writing courseware there are two principal components that the author needs to consider. Courseware is an instructional lesson that is delivered through the use of a computer. Thus, aspects of both computer and lesson come into play. With respect to the computer, the author must be concerned with three basic elements: 1) the set of instructions given to the computer, or the program, that makes it perform the desired operations; 2) the means by which these instructions are given to the computer or the editors; and 3) the system that manages the use of the computer, or the file management system.

The two aspects of the lesson are its design and content.

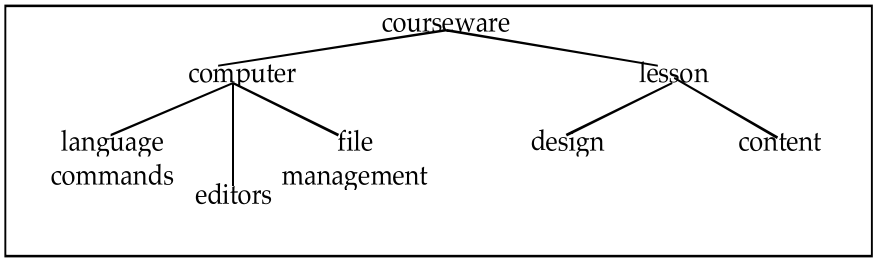

Figure 1: *Elements of Courseware Writing*

Computer Programming Language

An author communicates with the computer through the use of a programming language, interpretable by the computer, which consists of a set of commands or instructions. The author can choose from this set of commands in order to compose a program that makes the computer perform a specified series of operations. An example of a computer program is listed in Figure 2. This is a program written in BASIC, a language understood by most microcomputers, which instructs the computer to display the sentence "THE SUN___________(RISE)." on the screen, wait for the student to type in an answer, and then compare the student's answer to the correct one and give him a message.

Specifically, line 10 tells the computer to print " THE SUN___________(RISE)." on the screen. Line 20 stops the program and waits for the student to type something at the keyboard and press RETURN; whatever the student typed in is saved in a location of computer memory which is named AN$ (as specified on line 20). Lines 30 and 40 compare the contents of AN$ (where the student's response was placed) to the correct answer "IS RISING"; if it matches, says line 30, then print "OK" on the screen; if it doesn't match (doesn't equal), says line 40, then print "NO" on the screen. Line 50 identifies the end of the program.

```
10 PRINT " THE SUN___________(RISE)."
20 INPUT AN$
30 IF AN$ = "IS RISING" THEN PRINT "OK"
40 IF AN$ <> "IS RISING" THEN PRINT "NO"
50 END
```

Figure 2: Example of a BASIC Computer Program

Although instructional programs are written in BASIC, there are some functions that this language does not automatically provide that are desirable in instructional programs (Pusack and Otto, 1983). For example, when the computer executes the INPUT command in line 20, it will place the student's answer in the variable (name AN$) and then judge the contents of that variable as matching or not matching "IS RISING." However, the student might type the correct answer but leave two spaces between "IS" and "RISING"; that is, he may type "IS____RISING" instead of "IS RISING." A simple BASIC program will judge this answer as incorrect. A programming language provided by an authoring system may have special commands to avoid such a situation from occurring.

The computer language commands made available to an author through the use of an authoring system are, in principle, the same as those used in other computer programming languages; they instruct the computer to perform operations. The essential difference is that an authoring language has additional commands that are useful for courseware development—namely, special commands for 1) scanning the student's answer; 2) accepting approximate answers; and 3) displaying alternate characters on the screen.

These special commands require some explanation. First, the command that accepts the student's answer must find and delete irrelevant characters such as extra blank spaces; it must also identify special function keys (e.g., keys which change from upper to lower case or indicate a diacritic mark). When the answer is assigned to a variable (such as AN$) it must be in the form that the student intended to type in so that the program can judge it accurately. An authoring language provides commands that perform these routines automatically.

Second, there are a number of decisions to be made concerning how the student's answer should be judged. For example, the author may want to allow the student to type in an answer that is misspelled or that contains extra words in it or is a synonym for the correct answer. These are the options that are frequently made available to the author through the use of special authoring commands.

In the partial program listed in Figure 3, authoring language commands (Avner, Smith and Tenczar, 1983) are used to perform several of the functions named above. Line 200 instructs the computer to print the question on the screen. The &I command in line 300 is similar to the INPUT statement in the program in Figure 2; it stops the program, waits for input from the terminal and places this input into a location in the computer's memory. However, this command also scans the student's answer for irrelevant or special characters. The &A command in line 310 tells the computer to compare the student's answer with the one specified: "electron." The "X" means that it is all right if there are extra words in the student's response; the "M" means that misspellings of the correct answer are acceptable. Therefore, a student could give one of many variations of the correct answer and the program would consider them all "OK." Possible correct answers include: "electron," "an electron," "its electron__."

Finally, in foreign language materials it is frequently necessary to display a variety of different kinds of characters (e.g. regular text, subscripts, foreign character and diacritics) in different sizes (large ones for titles and emphasis) and different colors. One may also want to use different writing "modes" to print on the screen (e.g., plotting in erase mode over a solid color). Courseware writers may also want to display graphics and animation in order to illustrate particular points. An authoring language can have additional commands that allow the author to display things other than standard text on the screen.

```
Answer the following question:

What is a negatively charged particle called?
>
            Press RETURN to continue.
```

Figure 3: Screen display as seen by the student (Left) from a program using authoring language commands (Right)

```
200 PRINT "What is a negatively charged
particle"
210 PRINT "called?"

300 &I
310 &A"<X><M.electron>"
```

Editor

An author uses an editor to enter program commands into the computer's memory by typing them at the keyboard. The editor is the software that accepts each character as the author types it and sends it to the computer's main memory, thereby allowing each line to collect in memory forming a program that, when used by a student, will instruct the computer to display questions on the screen, accept the student's answers, etc. As Figure 4 illustrates, the author types in (inserts or enters) commands on the keyboard. If he then wants to delete something that has been typed, he can enter an editing command that will tell the editor to stop accepting information and to get rid of (delete) the part specified. Similarly, if the author remembers that he needs to enter something that was forgotten when the code was typed in, he can give the computer another command telling it to add the part that had previously been forgotten. The editor will move the characters that were already there to make room for the new characters.

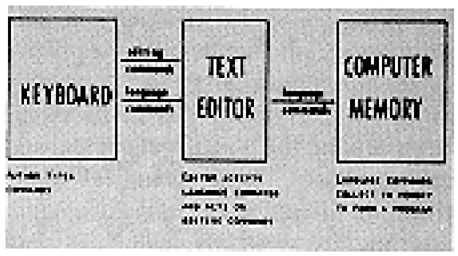

Figure 4: Functions of a Text Editor

Every computer that supports program writing has some kind of editor; however, an authoring system may include a more powerful editor that the one that is standard with the system. A more unique feature of an authoring system than its text editor is its other editors: editors for character sets (alternate alphabets), graphics (pictures), sound (a set of musical notes) and video (film).

A character set editor is a very useful feature of an authoring system for FL courseware (see, for example, Kachru, Nelson and Hart, 1981). In order to understand how a character set editor works, it is first necessary to understand how a character (letter, number or symbol) appears on a computer's screen (e.g. an Apple II), which is composed of a grid of rectangles consisting of 24 rows and 40 columns. Each of these (24 X 40 =) 960 rectangles is composed of a set of dots: 8 rows long and 7 columns wide. A character appears on the screen of the microcomputer as a combination of dots within the grid that have been lit up. For example, if the lit (filled in) dots on the grid in Figure 5 appeared on the screen, the student would see the letter "1". If the lit dots formed another configuration the student would see that on the screen.

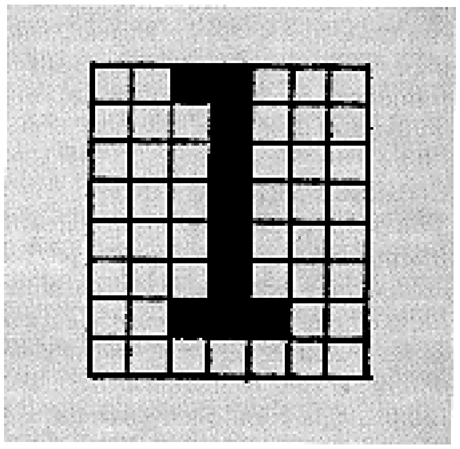

Figure 5: Diagram of the Numeral "1"

Through the use of a character set editor, the author can define the configuration that the lit dots will take in each of the positions on the screen, thereby creating the special characters that a lesson may need. In principle, a character set editor works in the same way as a text editor (Figure 6). It differs in that instead of presenting the author with a blank "page" onto which he writes programming language commands, it presents the author with a 7 X 8 grid on the monitor into which the author enters character information. The author can then move the cursor around on the grid filling in, or turning on, the individual squares that he wants to show as part of the character. The editor will let him create and store a whole set of these alternate characters that can then be used in a program.

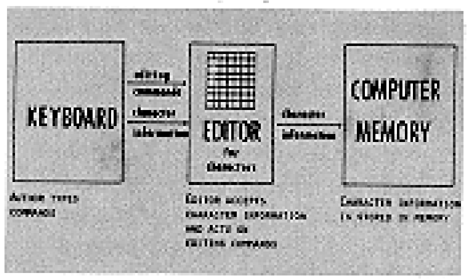

Figure 6: A Character Set Editor

A graphics, sound or video editor is similar to a character set editor; however, rather than filling in squares on a small grid, an author fills in dots on the entire screen, notes on a music scale or positions on a film.

File Management

Giving the computer information to be stored in memory is only part of the task needed to write courseware. On a microcomputer system, when the power is turned off, the information that is stored in some parts of memory is lost. Thus, it is necessary to have a means for transferring information from the main memory to a secondary storage location where it can be stored permanently. On a microcomputer system, this secondary storage location is typically a diskette. The software needed for transferring information will be called a file management system.

A programmer who uses a microcomputer system must know a number of commands (e.g., SAVE, LOAD) and understand the fundamentals of how the computer works in order to use the file management system. An authoring system can free the teacher from the knowledge of the computer and how it works by incorporating file management options in a menu form. The author can simply select the desired option from a menu (Figure 7). The built-in management system will respond to the teacher's choice to, for example, load an editor and a file into the computer's memory so that the teacher can begin to work on it.

```
TO THE TEACHER:

What would you like to do?

                Edit a lesson
             Edit a character set
                Edit a graphic
            Duplicate a lesson disk

                      >
```

Figure 7: File Management Options Presented to the Teacher in Menu Form

LESSON
Design

The lesson that a student uses consists of two major elements: design and content. The design refers to the organization and presentation of the material to be covered (Dick and Carey, 1978; Steinberg, 1984). Lesson design can be exemplified by outlining six of its major features. First, a lesson can be defined as having one of a combination of design types: drill and practice, tutorial, simulation, game, or some combination of these four. Second, sequencing of material must be considered as part of the design. Will the lesson be linear or branched? Will there be an index from which the students choose what they want to work on? Third, if there is to be a sequence of material, decisions for choosing a path through the material must be made. These decisions might be made by the computer, based on student performance, or by the student, based on how he/she wants to work. Another element of design is planning for records to be kept as the student works. A fifth consideration is planning how students' answers are to be processed and judged. The last element of design is screen display. Each part of the script must be drawn to the scale of the computer's monitor so that it will fit on the screen and look pleasing.

Some authoring systems, often called template systems, provide an author with a program that has already been written, thereby eliminating the need for the author to be knowledgeable about lesson design. A template program prompts the author, telling him/her to enter particular pieces of content at certain points in the program. Figure 8 illustrates the kind of prompts the author would get from a template that allowed for the creation of three kinds of drill exercises. The program begins by offering the teacher a choice of what kind of lesson he wants to make. In this case the teacher chooses a fill-in-the-blank drill (c) and then the program responds with a prompt asking for the first item. After the teacher types in the first question, the program asks for the correct answer to that question. When the teacher types in the answer, the program will continue by asking for the second question, etc.

```
TO THE TEACHER:

What kind of item would you like to write?

a. multiple choice
b. True and False
c. Fill in the Blank
```

```
TYPE ITEM NUMBER 1:

>A negatively charged particle is called an _________.
```

```
TYPE ANSWER NUMBER 1:

>electron
```

Figure 8: A template program that allows for development of different types of items, and lets the teacher type in items and answers.

As the teacher is typing in this information, the program is storing it on disk so that another program, which is also part of this authoring system, can use it to present a lesson to the student. A diagram of how this process works appears in Figure 9.

CONTENT

The lesson content is the subject material covered in the lesson. Content for a computer lesson is a script which contains everything a student sees on the monitor including questions, right and wrong answers, titles, headings, messages, directions, rules, or feedback messages. For example, the content of a lesson on inferring the meanings of vocabulary words in context would consist of the passages for the student to read, questions on the meanings of the vocabulary items, right and wrong answers that the student might make, and messages informing the student whether or not he/she was correct and why as well as titles, directions and other lesson management details. The content of the lesson is defined by the lesson author; in fact, presentation of the teacher's choice of content is the reason for undertaking the process of courseware development. Therefore, the authoring system will do little to aid in the definition of content to be taught.

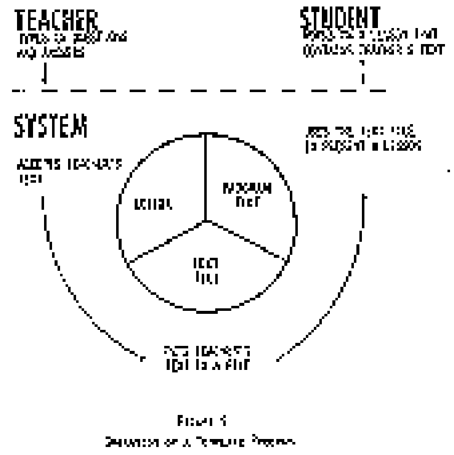

Figure 9: Operation of a Template Program

CHOOSING AN AUTHORING SYSTEM

Because authoring systems available today tackle the task of courseware writing through a combination of approaches, it is necessary to look at two factors in order to decide what kind of authoring system would be appropriate for a particular situation. First, the expertise of the courseware developer needed to use the system must be considered. Second, there must be an analysis of the kind of software that is intended.

Although the purpose of using an authoring system is to simplify the process of courseware writing, there remains some level of requisite expertise for using these systems. The level of expertise needed depends on the degree to which a particular system simplifies the aspects of courseware writing described above: programming, editing, file handling, and designing.

An authoring system that provides an author with a language alone leaves the author with a number of tasks to work out on his own. He will have to use whatever methods are available on the computer system for entering text, character sets and graphics. The computer's file management commands will have to be learned and used. The author will also be responsible for creating the lesson design. In order to implement the desired design, the author must learn to use the language provided by the authoring system.

Editors and file management systems make interacting with the computer much easier. However, when an author uses a system that has this kind of software, he must, of course, learn how to use them. In addition, of course, the author must know how to use the authoring language.

An authoring system that provides a design for the lesson already intact requires the least amount of expertise on the part of an author. Using this kind of system, the author doesn't have to write a computer program using language commands. In addition, all editor and operating system functions are prompted so that the author simply types in answers to questions. Yet, in order to work with such a system, an author must have some knowledge of the fundamentals of computer use; moreover, he must know how to select appropriate content material to fit into the design given.

In short, the claim that an authoring system is easy to use must be met by the question: Easy for whom? It is imperative that the people who are going to use the authoring system assess its ease of use for the application they have in mind (O'Neal and Fairweather, 1984).

Along with the expertise needed to use a system an author must consider the kind of software to be produced. The goal may be an innovative series of lessons to be marketed or it may be a few lessons for students at one school to use. In the former case, the author will want to use a system with a large set of powerful commands whereas in the latter case a system in which the lesson design is already specified may be the most viable alternative.

An authoring system for which a design has already been defined differs greatly from one in which an author creates the design and composes it from the commands that the language offers. For example, if a template program for producing multiple choice tests and close passages is used, these are the only types of lessons that will be developed. A computer language within an authoring system, on the other hand, can give the author the capability to implement any number of designs of his choice using the commands of the language. Because one authoring language differs from another, it is necessary to note that the degree to which an author can implement a particular design depends on the commands that the language provides. For example, if it is crucial to the design of the lesson that students' spelling or word order errors be detected, it is reasonable to use a programming language that has commands which perform these operations (Marty, 1984).

SUMMARY

The essential elements in writing courseware have been outlined in terms of components of computer (computer language commands, editors and file management) and lesson (design content). A description was given of how authoring systems can simplify some or all of these elements by providing an author with additional software. Finally, because the usefulness of an authoring system is project dependent, relevant points to consider in choosing an authoring system were discussed in terms of the expertise needed and the kind of software to be produced. This introduction should serve as a guide indicating what the author-to-be should look for as he explores a constantly growing number of authoring systems.

REFERENCES

Avner, A., S. Smith, and P. Tenczar. 1982. *ENBASIC: Enhanced BASIC for improved human-computer interaction*. Wentworth, NH: COMPress.

Dick, W. and L. Carey. 1978. *The Systematic Design of Instruction*. Glenview, IL: Scott, Foresman and Co.

Holmes, G. 1983. "Creating CAL courseware: Some possibilities." *System* 11, 1: 21-32.

Jamieson, J. and C. Chapelle. 1984. "Prospects in computer-assisted language lessons." CATESOL Occasional papers 10: 1-22.

Jones, C. 1984. "Teachers, authoring systems and computer-assisted language learning." *Medium* 9(3): 81-88.

Kachru, Y., C. Nelson, and R. Hart. 1981. "Computer-based instruction in elementary Hindi." *Studies in Language Learning* 3,1: 54-73.

Marty, F. 1984. "Computer-aided instruction: Language teachers and the man of the year." in Savignon, S. and M. Berns. *Initiatives in Communicative Language Teaching*. Reading, MA: Addison-Wesley Publishing Co.: 155-67.

O'Neal, F., and P. Fairweather. 1984. "The evaluation and selection of computer-based training authoring systems." *CALICO Journal* 2(2): 27-46.

Pusack, J. and S. Otto. 1983. "Stringing us along: programming for foreign language CAI." *CALICO Journal* 1(2): 26-47.

Steinberg, E. 1984. *Teaching Computers to Teach*. Hillside, NJ: Lawrence Erlbaum Associates.

Wyatt, D. 1983. "Three major approaches to developing computer-assisted language learning materials for microcomputers." *CALICO Journal* 1(2): 34-8.

Wyatt, D. 1984. *Computers and ESL*. Orlando, FL: Harcourt, Brace, Jovanovich, Inc.