TrainingNews
Volume VIII, Number 7 March 1987

THE CBT CRAFTSMAN

Authoring Productivity Revisited

Along with everything else, CBT developers worry about their productivity ratios. One antidote to that worry may be the use of symbolic authoring systems.

Jesse M. Heines

A few weeks ago I received a call from a course developer in St. Louis. She had just finished reading my November 1986 column ("Front End Drive") on the use of authoring system front-ends to increase course development productivity. She told me that she and a colleague had been using Pascal for CBT development, and that it had taken the two of them a full year to produce one eight-hour course. "Management here thinks that's too much time," she said, and she told me that she was desperately looking for a tool that would help increase her productivity as a developer.

I replied that two man-years was a bit high for an eight-hour course, but only by a factor of about two. A man-year is about 1920 hours—48 weeks (52 weeks minus two weeks of vacation minus 10 holidays) times 40 hours per week—so 2 man-years is about 3840 hours. Dividing 3840 by 8 yields 480 development hours per hour of instruction.

That number is not surprising at all, particularly for first-time developers working with a general purpose language without the special routines provided by specialized authoring languages. Including all the time spent on planning, subject matter research, and instructional and graphic design, experienced CBT developers typically require between 100 and 250 development hours per hour of instruction, depending upon the complexity of student-computer interactions.

We developers can argue all we want to about instructional quality, but the bottom line will always cause management to try to increase our productivity. Before we try to see how this might be done, let us establish the following postulates.

1. There is no substitute for time spent on design.
2. Authoring languages and systems generally address the programming aspect of CBT development, not the design.
3. The production of quality CBT requires materials to be developed, tested, and revised in an iterative manner.

Given these concepts, you should see that no matter how hard you get pushed to increase productivity, you shouldn't compromise on design time. You can't make a good picture from a bad negative. If you cut corners at the design stage, your boss or client may get the course more quickly, but when students fail to learn from it, the onus of their failure will ultimately be directed toward you. While authoring languages and systems will certainly decrease the amount of time it takes to program a CBT course, neither will be able to tell you how a course should be developed. I personally guarantee that you will quickly outgrow any authoring system that allows only a small set of inflexible instructional designs.

The key to increasing productivity, then, is to attack the iterative process of development, testing, and revision. Mass manufacturer Henry Ford and economist John Kenneth Galbraith have shown us that people in any system—computerized or not—can usually achieve greater productivity increases by shaving small bits of time from processes that are done over and over, rather than by sawing large chunks of time from processes that are only done once.

Working with symbolic authoring systems allows a course developer to do just that, because they allow you to get a handle on the entire course structure and to manipulate layout and sequence easily at both the lesson and individual interaction levels. These are the parts of a CBT course that change the most as new lessons are added between existing ones, and lessons or sequences that teach poorly are eliminated or replaced. I used *Course of Action* (from Authorware, Inc., in Bloomington, Minnesota) to illustrate points in my last column ("Not Just Another Pretty Authoring System," January), so I'll use *Maestro* (from AIMtech Corporation in Nashua, New Hampshire) in this one.

Symbolic authoring systems such as *Maestro* and *Course of Action* have the potential to affect course development productivity significantly, but it is difficult to show this without reviewing them in considerable detail.

Now, it is not my intention to promote either *Maestro* or *Course of Action* commercially, nor is it my intention to rate them one against the other feature-by-feature. The most significant differences between them are direct functions of their host computers: *Maestro* runs on the IBM PC (and other systems) with medium resolution color graphics, while *Course of Action* runs on the Macintosh with high resolution black and white graphics (the Mac does not support color). *Maestro* also supports the videodisc, while *Course of Action* does not. (As far as I understand, the Macintosh video board is incapable of being interfaced to videodisc signals.)

A Partial *Maestro* Walk-Through

Figure 1 shows the author's top level when *Maestro* is initially invoked. *Maestro* requires a software windowing package from Microsoft Corporation called *Microsoft*

Jesse M. Heines is an assistant professor of computer science at the University of Lowell in Lowell, Massachusetts. He is the author of Screen Design Strategies for Computer-Assisted Instruction *as well as numerous articles on courseware development. Dr. Heines provides training and consultation on computer-based training, develops custom training programs on contract, and writes "The CBT Craftsman" every other month.*

Windows which, in turn, requires a Microsoft mouse. These tools make the *Maestro* editing environment virtually identical to that of the Macintosh. Courses are built by using the mouse to move icons from the icon library in the icon library window to appropriate positions on the course flowchart in the course flowchart window.

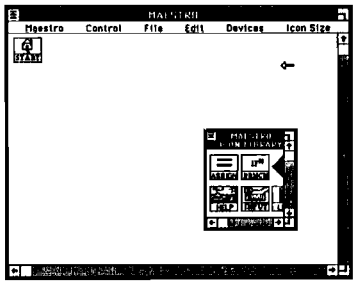

Figure 1. *The top level of AIMtech's* Maestro *system as seen by a course author. (Copyright 1986, 1987, AIMtech Corp., reprinted with permission.)*

As a course is built, the course flowchart expands, as shown in Figure 2. When the course becomes too large to fit in the course flowchart window, *Maestro* allows the size of the icons to be reduced, or you can just scroll the course flowchart window.

Now comes the productivity enhancement stage. Let's say that testing of the course with sample students reveals that the material currently presented in the third section would better be presented second, and that the material in the current second section should be revised to build on the material currently presented in the third. That is, we want to switch the order of presentation of the current second and third sections and then revise the new third section.

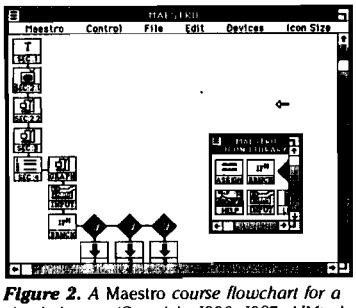

Figure 2. *A* Maestro *course flowchart for a simple lesson. (Copyright 1986, 1987, AIMtech Corp., reprinted with permission.)*

To switch the current second and third sections, the author begins by "touching" the first icon of the second section. (One "touches" an icon by putting the mouse pointer on the icon and pressing the mouse button.) The author then touches the word EDIT at the top of the course flowchart window and a menu of editing functions pops onto the screen (see Figure 3). Next, the author touches the RANGE ON function, and then touches the last icon of the second section. All icons between the first one touched and the last one touched now change color, indicating that they have been "selected."

The author now touches the CUT function in the edit menu and the selected icons disappear into a buffer, or temporary holding area. Figure 4 shows how the flowchart is redrawn to eliminate the icons that have been cut out. Note, though, that the integrity of the course structure is automatically maintained.

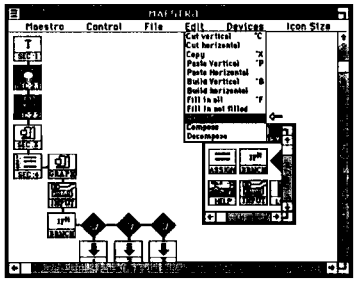

Figure 3. *Menu of editing functions. (Copyright 1986, 1987, AIMtech Corp., reprinted with permission.)*

By using the editing paste function, the author can reinsert the icons in the buffer back into the course at the desired position. The author first selects (by touching with the mouse) the icon in the course flowchart to which he or she wants the icons in the buffer appended, and then touches the PASTE function in the edit menu. Once again, the flowchart is redrawn to reflect the changes while maintaining the integrity of the course structure (see Figure 5).

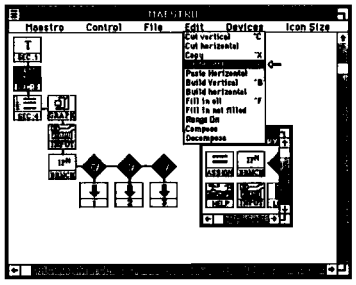

Figure 4. *The course flowchart after the icons for the second section have been cut out. (Copyright 1986, 1987, AIMtech Corp., reprinted with permission.)*

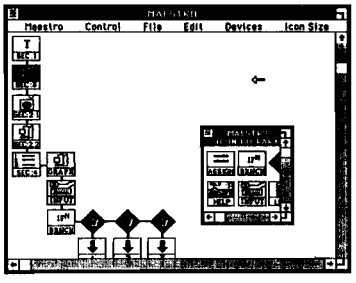

Figure 5. *The course flowchart after the icons for the old second section have been pasted in at the end of the third section. (Copyright 1986, 1987, AIMtech Corp., reprinted with permission.)*

Revision of the old second section (which is now the new third section) is carried out in a similar manner. The author selects the course component icons he or she wishes to edit, chooses the appropriate functions from the editing menu, and performs the desired changes in an interactive manner. The author may delete or modify existing icons or add new icons, and *Maestro* will automatically maintain the integrity of the course structure at all times.

At any point in the revision process the author may touch the word CONTROL at the top of the course flowchart window to display a menu of control functions. One of these functions allows the author to run the course as a student to try out his or her revisions. Thus, the iterative process of development, testing, and revision is streamlined considerably, and the developer's productivity enhanced accordingly.

As far as I know, no full courses have yet been developed with any symbolic authoring system, although small demonstration programs are certainly available for both *Maestro* and *Course of Action*. I have tried in my last two columns to emphasize the *potential* I see for these systems to affect course developer productivity positively, but I have no empirical data to support this view.

One of my colleagues (whom I respect greatly) thinks the constant use of the mouse will be very tedious in an eight-hour-per-day production environment, and perhaps he's right. Personally, I think that using the mouse wouldn't be at all tedious, but I admit the constant shifting from mouse to keyboard and back would become tiresome for an experienced typist. (The people at AIMtech Corporation demonstrated one procedure in which the mouse interface could be circumvented and the editing actions read from a control file—but this was definitely not standard procedure.)

One fact I'm quite sure of is that the use of these systems requires considerable computer power. There are a number of instances in which I have seen courses I developed on my own IBM XT run on an IBM AT, and the overall effect of the increase in speed is startling. Even though the course really runs only two or three times faster, the impression is that the increase in speed is much greater. The whole "feel" of the course changes. I'm sure that most developers who have gone from systems with floppy drives only to hard disk systems would revolt if they were asked to give up the faster access time they now enjoy.

In fact, research at IBM shows that two easy ways to increase users' productivity are to give them faster system response time and more disk space. Both of the symbolic authoring systems discussed in this column require windowing and complex screen management in author mode, and I'm sure that their performance would be unacceptable on very small personal computers. *Course of Action* clearly requires the Macintosh, while *Maestro* requires a system with enough "heft" to support *Microsoft Windows*.

I invite readers to write to me c/o *Training News* to share additional views on increasing course developer productivity. Readers interested in further information about *Maestro* should write to John Olapurath, chairman and CEO, AIMtech Corporation, 77 Northeastern Boulevard, Nashua, New Hampshire 03062, or call (603) 883-0220. □