The Right Stuff

What does it take to excel at courseware development? Where did today's masters come from? Where will they come from tomorrow?

JESSE M. HEINES

Jesse Heines is an assistant professor of computer science at the University of Lowell. Dr. Heines also conducts workshops and consults on designing and authoring computer-based training programs. He is the author of Screen Design Strategies for Computer-Assisted Instruction *(Digital Press, 1984, Bedford, MA).*

When I look around at the courseware developers I consider masters, a few things quickly become apparent. Most, for instance, come from technical backgrounds rather than from education or instructional development. They have degrees in the pure and applied sciences or other fields quite unrelated to the development of courseware. Many of them are college or university educators, however, and have a history of using media in their classes, sometimes in the form of individualized instruction, sometimes simply as an enhancement to more traditional lecture or classroom instruction.

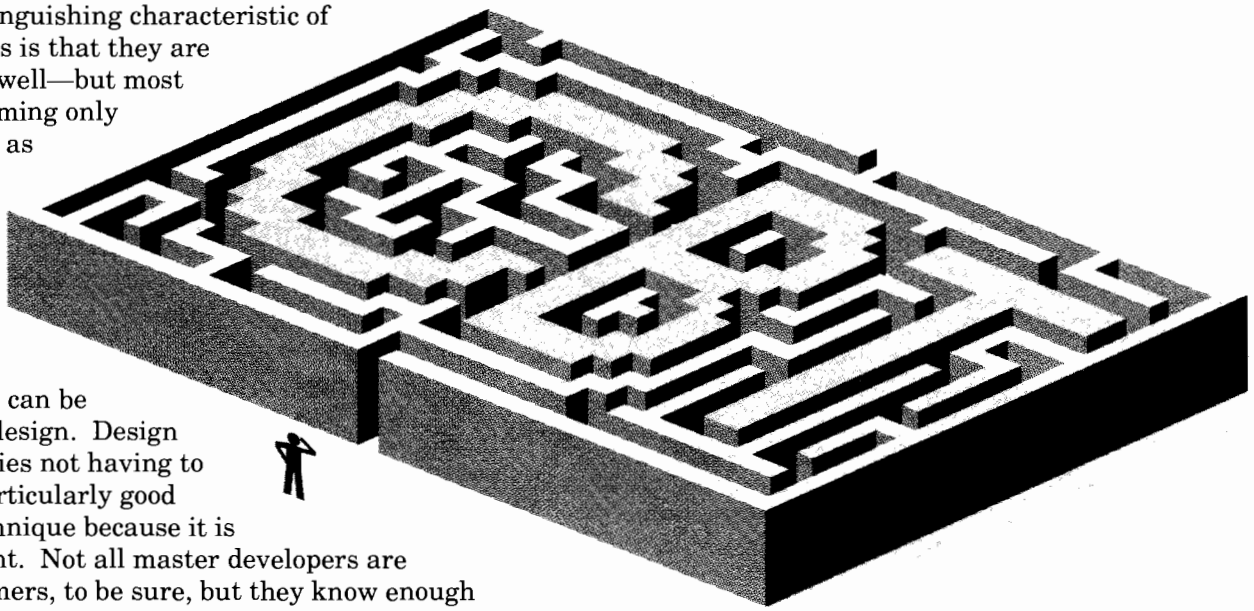
How does their technical background help? For one thing, a strong technical background usually lets master developers function as their own subject-matter experts. They are able to do their own task analyses and they possess the knowledge to fill in the technical details.

Another thing you notice about today's courseware masters is their ability to design attractive computer displays, a talent not explained by their technical backgrounds. Where does this innate creative flair and eye for design come from?

One answer might be the willingness of exemplary courseware developers to scrutinize and learn from the work of others. In my own case, I've discovered that every piece of courseware I have looked at—no matter how poorly designed—has had at least some "redeeming social value" (to borrow the words of the Supreme Court), and has taught me something either positive or negative about display techniques.

> Any discussion of training must begin with a task analysis. What are the major skills that one can reasonably expect a CAI/CBT developer to possess?

(The PLATO community of courseware developers, for example, was a tightly knit group that constantly shared and critiqued each other's courseware. They still do. Not surprisingly, the best developers I've worked with always had PLATO experience.)



Another distinguishing characteristic of master developers is that they are programmers as well—but most came to programming only because it served as a tool for implementing their instructional designs. Their philosophy: The better one can program, the more faithful one can be to one's original design. Design faithfulness implies not having to compromise a particularly good instructional technique because it is hard to implement. Not all master developers are master programmers, to be sure, but they know enough to ask the right questions of master programmers and not to accept "it can't be done" as an answer. No programmer worth his or her salt will tell you that something can't be done; they'll tell you which things can be done easily and which will take more time.

Two more characteristics of master developers—the ability to design tests and the ability to evaluate program effectiveness—are easy enough to "buy" from another party (indeed, many of the original PLATO developers used in-house specialists in these areas rather than relying on their own expertise). But like programming, the more one knows about test construction and statistical analysis, the better one will be at improving courseware from the feedback that these instruments provide.

A strong technical background usually lets master developers function as their own subject-matter experts. They are able to do task analyses and possess the knowledge to fill in the technical details.

The last major characteristic of master developers is experience. All master CAI and CBT developers can show you examples from their courseware that are "wrong," that is, things that they would have done differently if they "knew then what they know now." There is simply no substitute for developing lots of courseware and evaluating its effectiveness with a variety of students. As one of my best friends likes to say, "I reserve the right to be smarter tomorrow than I am today!"

How Tomorrow's Master Developers Might Learn the Trade

The learning process today's masters went through is as valid now as it was then. The problem many developers face, possibly yourself, is time. If a formal course of study is not available at your university or place of employment, I recommend the following methods to decrease the time needed to learn the trade:

- Study as many CAI and CBT programs as you can and use your own judgment as to which screen displays and student/computer interactions are interesting and informative and which are not. Look especially at programs written for systems other than the one that you will most likely develop courseware for. Developers who work primarily on IBM systems should look carefully at Macintosh courseware and vice versa, and both should look carefully at the Amiga. Every system has capabilities that make some techniques easier to implement than others, but one can achieve very interesting effects by simulating in software techniques what another system has in hardware. Developers familiar with only one system often develop tunnel vision that limits their creativity and makes their courseware repetitive and boring. If possible, spend time at a PLATO site going through some of the classic lessons as well as the newer ones developed at that site.
- Learn as much as you can about computer programming. Despite the ease of today's authoring systems, all authoring involves programming concepts

ILLUSTRATION BY MARK SIMONSON

such as data storage, branching, iteration (looping), and modularization. Knowledge of these concepts will help you develop more varied and interesting courseware regardless of the authoring system you use.

> No programmer worth his or her salt will tell you that something can't be done; they'll tell you which things can be done easily and which ones will take more time.

• Take instructional design courses offered as practicums so that you get maximum feedback from your instructor on the materials you develop. This is an excellent way to gain a relatively large amount of experience in a short time, even though the feedback will be from a specialist rather than from actual students. Work on individualized lessons without the computer to concentrate on presentations and interactions without having to deal with the complexities of actual lesson authoring. Study other students' work carefully to learn from their creativity and from their mistakes.

• Read the few books on courseware development available, even though some will be relatively out of date. Kearsley, Steinberg, Hofstetter, and Bork are among the more prolific authors who have written materials readable by beginners.

• If you can, take training seminars that offer hands-on experience. I recommend such seminars over the lecture kind for two reasons. First, I think you will learn better when you try to implement the techniques being discussed, even if you are unfamiliar with the seminar's authoring system. Second, I find that those instructors who offer hands-on experience are usually more experienced themselves and provide more practical advice than those who espouse theory without at least showing courseware that demonstrates those theories. If seminar instructors don't use media in their presentations, how much can they really know about using media in CAI and CBT?

• Go ahead and start developing your own courseware as soon as possible. Pick your favorite hobby and develop one short module on it for your kids, friends, or significant other. Let them tell you what they think, and go back and revise the module based on their input before you develop another module. Revision is a huge part of CAI/CBT development, and you must learn early how to build your courseware so that it can be easily revised.

Your Turn

What do you think? Do you agree with my ideas or does your company or university have a better plan? What have you found useful and what a waste of time? What do you wish you had more time to study? I am especially interested in hearing the details of formal industrial or university programs designed to train CAI and CBT developers.

Please write or send me materials on your program:

Jesse M. Heines
The CBT Artisan
18 Courtland Drive
Chelmsford, MA 01824

The Right Stuff—One View

What are the major skills that one can reasonably expect a CAI/CBT developer to possess? I offer the following list:

✓ The ability to perform a task analysis to determine what needs to be taught.

✓ The ability to interact with subject-matter experts to extract the technical details of the instructional content.

✓ The ability to design attractive and informative computer displays to present the instructional material.

✓ The ability to tie those displays together into an instructive, consistent, and comprehensive whole —that is, to create a storyboard that serves as the blueprint for the on-line course.

✓ The ability to translate that blueprint into a working computer program.

✓ The ability to design tests that measure students' comprehension of the material.

✓ The ability to evaluate the effectiveness of the program as a teaching tool.

These are the major skills, but we need to go a bit further: What talents make a CBT developer good?

✓ Strong knowledge of what the computer can do.

✓ The ability to work as an effective member of a team of developers and programmers.

✓ A highly creative flair for presenting instructional material using the computer medium.

✓ A solid gestalt encompassing the specific goals of a particular teaching program and the general facets of learning by computer.