THE CBT CRAFTSMAN

Front End Drive

One way to solve the traditional power-speed dichotomy in CBT authoring is to build "bridges" between simplified and complex authoring languages.

Jesse M. Heines

My last two columns have discussed two techniques for taking advantage of the helpful features that authoring systems offer. Those two techniques can also be used to remove the restrictions that authoring systems often impose. I've already written about making authoring systems callable and providing multiple authoring levels. This column will discuss a third technique, designing custom authoring "front ends" for specific instructional applications.

Let's begin by analyzing how information gets transformed from the way it is represented by a course author into a form that can be represented by a computer system. The human being perceives the course through the abstractions of subject matter, instructional design, and screen design. The computer, however, perceives it only as a set of "wanzanzeroes," the 1s and 0s into which all software must be translated before it can be executed by a computer (see Figure 1).

CBT authoring systems simplify the translation process by providing an easy-to-learn language in which course authors can represent their subject matter, instructions, and screen designs. These systems form a bridge between the author and computer, as shown in Figure 2. Two transformations are required: the first is performed by the author, the second by the computer.

I have argued (relentlessly, some say) about the limitations of this approach. The major problem, in a nutshell, is that in order to make the first transformation as easy as possible for the human being, authoring system developers sacrifice much of the real power available from the computer. Worse still, the system developers actually prohibit course authors from accessing that power, if the authors wish to use the features of their authoring systems.

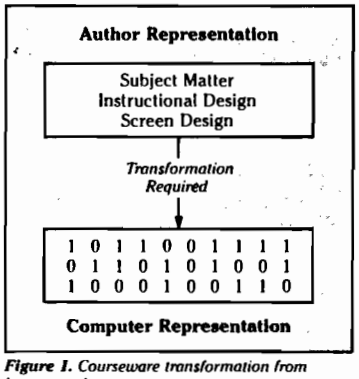

Figure 1. Courseware transformation from human author to computer presenter.

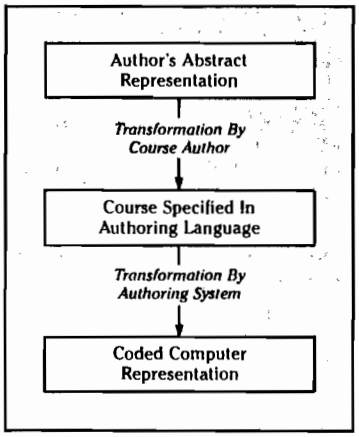

Figure 2. Courseware transformation from human to computer via an author language

The two techniques discussed in my July and September columns require changes in most existing authoring systems. Perhaps these changes will be more widespread in the future, but this month I'd like to concentrate on what we can do with today's tools.

Most of the courseware I've developed in recent years has been done in *TenCORE*, a sophisticated authoring system available from Computer Teaching Corporation in Urbana, Illinois. When I talk to people about *TenCORE*, I always seem to get one of two assessments:

- I like *TenCORE* because it's a real programming language.
- I don't like *TenCORE* because it's a real programming language.

Here we have a classic example of the dilemma I've been talking about for years: you can have ease of learning or you can have power, but it's very difficult to achieve both in the same system.

This is not to say that a powerful system has to be complex and opaque. What it means is that you won't get the most out of a CBT system that requires only two days' training. It also means that you won't learn how to use a powerful system to its full capabilities with two days or even two weeks of training. Getting the most out of CBT requires experience with your authoring system as well as knowledge of its commands and functions.

One way to bridge the difficulty of learning a powerful system is to create a "front end" for that system—a program that takes the authors' course design and translates it into a form that can be read by the authoring system. This procedure creates yet another bridge, as shown in Figure 3. This approach has all of the advantages of the one shown in Figure 2, but with a very important extension; the system is no longer bounded by the capabilities of the simplified authoring language. If course authors wish to add additional capabilities, another front end program can be built to handle those.

In addition, course authors who are also programmers may use the full power of the authoring system *as well as* the convenient features of the front end (see Figure 4). In a sense, this is similar to the ability to jump up and down between two authoring levels as described in my September column. The difference, however, is that the front end may be easily customized to specific applications, greatly simplifying the development process for certain types of courseware.

In practice, the use of front ends requires the writing of two substantial authoring tools: the front end program itself and a driver to interpret the front end's output. The driver is a set of subroutines which read and process the data files generated by the front end. My experience is that these subroutines can usually be made very generic. In fact, to be truly effective these subroutines must steadfastly avoid any dependency on interpretation or intrinsic knowledge of the subject matter. This "generalizability" allows any hand-coded routines written by programmers to call driver rou-

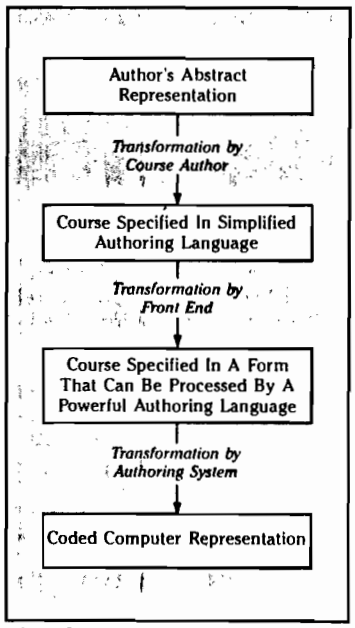

Figure 3. Courseware transformation from human to computer via a front end.

Jesse M. Heines, Ed.D., 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.*

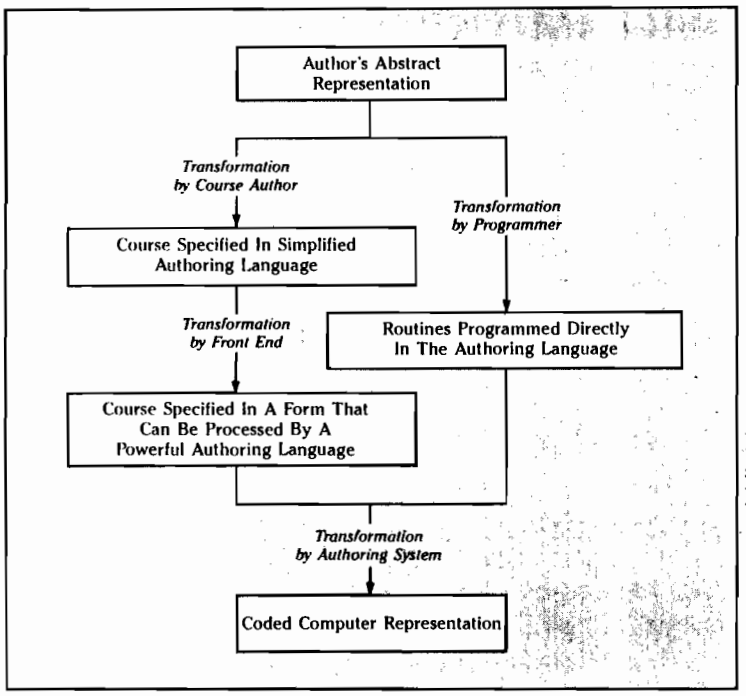

Figure 4. *Courseware transformation via a front end and additional hand-programmed routines.*

tines as well, thus providing a method for achieving standardization as well as simplifying the programming task.

I have developed a *TenCORE* front end along the lines of the concepts presented in this column for Scientific Systems, Inc., of Cambridge, Massachusetts. Dr. Larry Israelite (Scientific Systems' director for training products) and I presented a paper on this software at a recent conference of the Association for the Development of Computer-based Instructional Systems (ADCIS). Readers interested in more information are cordially invited to write to me in care of *Training News* for a free copy of the ADCIS paper. □