TrainingNews
Volume VII. Number 11 July 1986

THE CBT CRAFTSMAN

Can I Call You?

It's a sign of CBT's immaturity that there are dozens upon dozens of authoring systems on the market.

Jesse M. Heines

I have a house. It's not the most expensive house in town, and it doesn't have all the latest gadgets advertised to make living easier. I doubt you'd see my house advertised in *Better Homes And Gardens.* It is well-kept, though, and I've done some remodeling to make it extremely functional for my lifestyle. I know my way around the rooms and storage areas and can accomplish my normal daily chores very efficiently. It's home.

I have a computer, too. It's not the most expensive computer on the market, and it doesn't have all the latest hardware and software advertised to make computing more user-friendly. I doubt you'd see my computer featured in *Personal Computing.* It is well-kept, though, and I've done some customizing to make it extremely functional for my work style. I know my way around the directories and files and I can accomplish my normal daily work very efficiently. It's home.

As a homeowner, I receive advertising flyers regularly from Sears, Zayres, and a host of other businesses trying to sell me things for my house. Sometimes they go straight into the wastebasket, but more often I look them over quickly to see if there's anything on sale that might be applicable to one of the projects that I always have going.

For example, I recently remodeled a bathroom in my house, and I looked through every arriving flyer for a suitable medicine cabinet. While price was certainly a consideration, my major reason for rejecting most cabinets was that they didn't fit into the environment of my home.

As a personal computer owner, I receive advertising flyers regularly from Borland, IBM, and a host of other vendors trying to sell me things for my computer. Like the house advertising flyers, some go straight into the wastebasket, but most get a quick look from me to see if there are any sales I can apply to one of my ongoing projects.

For example, I recently became disenchanted with my spelling checker program. I looked through every arriving flyer for a suitable replacement. While price was certainly a consideration, my major reason for rejecting most replacement software was that the programs didn't fit into my computer's environment. That is, using some spelling checkers would have forced me to change the way I work.

As the "CBT Craftsman," I have the opportunity to evaluate new authoring systems regularly. I can honestly say that almost every system I've seen has some feature that I'd like to include in my courseware. ForceTen has a semantic parser. Trainer 4000 has a music editor. PC Pilot has *Storyboard*-like graphic display options. TenCORE has sophisticated programming capabilities. And SAM has a videodisc interface.

Unfortunately, if I write a course using one authoring system, I can't use the features provided by another authoring system. I have to decide which system I'm going to use and I have to live with its shortcomings if I want its advanced features. I share the lament expressed by my colleague Gloria Gery in the March issue of *Data Training*—"I want it all!"

For me to adopt a new authoring system, it must fit into the environment of my courseware development process. It must not change the way I work, and it certainly must not require me to throw away all of the courseware and course development tools that I've developed in the past or to reprogram them in a new language.

We need to get away from developing new, self-contained authoring systems which are incompatible with other software. In *Data Training*'s fifth "Annual Survey of CBT Authoring Systems" (May), there are 50 different systems listed for microcomputers alone! This is an appalling situation that clearly shows the immaturity of our craft. Developers seem to think they can build a better CBT authoring system, while virtually ignoring all the precepts of good software engineering. By comparison, consider the total number of high-level languages that are used throughout the entire field of computer science—about a dozen.

We should not be building new authoring systems for every application. Rather, we should be building new subroutines that are *callable* from existing authoring systems. For a subroutine to be callable, the authoring system must be able to execute the code in the subroutine without losing the context of the authoring system environment. The authoring system should not have to exit and re-enter, the screen mode should not have to be reset, and the values of variables *after* the subroutine is called should remain the same as they were *before* it was called. When the need for a new CBT function is identified, or a new CBT feature is invented, the developers' first strategy should be to write a subroutine that can be called from every authoring system they may be asked to use. That first inclination should *not* be to write a new authoring system that incorporates the new function or feature.

Now, some authoring systems let you call *out*, but they don't let you call *in*. That is, they have the ability to call subroutines written in assembly language or even to execute operating system calls, but the authoring system itself has to be the controlling process. This is fine if you can program in assembly, but not if you are using one authoring system and want to use a feature in another authoring system. Both systems can't be on top of the pyramid.

Due to the advantages of certain features for certain types of courses, it's not uncommon for me to work with companies that employ different systems for different courses. This situation creates a software support nightmare, especially when employee turnover is high. My usual advice to companies in this situation is to adopt one authoring system and to employ programmers to write subroutines for those functions that are not intrinsic to the system. Course developers can then call these subroutines to perform their desired functions.

My usual advice to authoring system vendors is to make their systems truly callable, so developers can take advantage of their new features regardless of the authoring system in which the major portion of their courseware is programmed. This is the approach taken by virtually all software developers outside of the CBT community, particularly in the area of computer graphics. For example, the relatively new device-independent Graphical Kernel Standard is not a new language. It's a set of subroutines callable from Fortran, Pascal, and other standard compiled languages.

When someone shows me a new authoring system, I am always wary of getting excited about its new features. If I can't integrate the system into my existing course development process, I will be hard pressed to justify adopting it due to the large investment I've already made in my existing CBT courseware. My first question to a new authoring system, then, is seldom, "What can you do?" More often, it's a very quiet and apprehensive "Can I call you?" □

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 CBT, develops custom training programs on contract, and writes "The CBT Craftsman" every other month.*