On-Line Help and References for Courseware Developers

Brian L. Dear
Senior Courseware Developer
Training Systems Center
Hazeltine Corporation

This paper reports on the increasing use of on-line help and reference programs within computer systems, specifically authoring systems for the development of computer-based training. The advantages of developing a comprehensive help program are explained. The "help" features of two well-known authoring systems are described.

Coping with Computers

Like automobiles, computers must be understood before they can be "driven." We first learn how to sign on; once on, how to move around the system; once finished, how to save our work and sign off. This is never as easy as it sounds. Anyone who has used a computer has faced the familiar error message "Invalid data" or "Invalid response." We type a command and the system says "what?" We eventually learn to adjust to the system's quirks, much as we learn to cope with a temperamental transmission in a car. Recall that triumphant feeling after successfully signing on to a system for the first time. Recall the relief of finding two hour's work still safe, still on-line, and ready to be edited the second time.

Computers for Instruction

Trainers use computers as tools: first to develop instructional material, and second to deliver it, once tested and debugged. Developers of computer-based training (CBT) most likely use specialized editors and programming languages tailored specifically for creating instructional material. These authoring systems, as they are called, range from simple menu-and-prompt programs for developing drills and quizzes, to super-sophisticated environments for developing the most complex of tutorials and simulations.

A Growing Market

Ten years ago, there were perhaps a dozen widely-used authoring systems; five years ago, this number had perhaps doubled. In 1986, there are probably several hundred authoring systems on the market. (See Locatis & Carr, 1985, for more information.)

Many Similarities Among Systems

While the number of systems has grown dramatically in recent years, the differences between today's systems and those of five and ten years ago are not great. A typical authoring system of today comes with text and graphics editors, student input editors, and branching editors. When we remove the whistles and bells, the appetizers and desserts, we discover that the majority of authoring systems offer a standard fare of tools: screen-maker, input-analyzer, and brancher. Some also allow some form of management system and calculator. Much of the current effort among creators of authoring systems is to improve this set of basic tools: make them sharper, more robust, more powerful, more plentiful. It is vitally important, however, that the <u>user</u> of these tools, the courseware author, understand how to use them; otherwise they go unused, however fancy and feature-laden.

Making Productive Use of a System

The degree to which an author can make use of a system relates to the degree to which the author knows the system. Experienced authors may "whiz" through a system, whereas novice users grope and grow anxious. This phenomenon applies generally to any computer application: accounting, word processing, graphics editing, statistical analysis, and database management. The increased productivity promised by computer systems only begins to appear when users finally understand the hows and whys of what they are doing.

The Ideal Scenario

If you visit a foreign land and do not know the native tongue, you quickly will realize how helpless and alone you are. Little can be accomplished when people cannot communicate. The ideal solution to language problems is having an interpreter, who speaks both languages, constantly at your side. Most of us cannot afford such a luxury. Hence, we rely on quick-reference bilingual dictionaries, complete with lists of common idioms and phrases.

In many cases of training, the ideal scenario for individualized instruction would consist of one live, expert tutor for every student. Since there will always be many more students than teachers, the only way to approach this ideal is to automate the tutor -- hence, CBT.

In the realm of using computers in any application, the ideal scenario for learning to maximally utilize a system would consist of one live, expert "helper" for every computer user. Again, such a scenario seems unlikely. Computer programmers, courseware developers included, generally rely on help from peers, training seminars, reference materials, and general trial-and-error experience. Automating the reference material has proven to be a great boost to any user's productivity: hence, on-line "help" programs.

Natural language interfaces (Dear, 1986; Rich, 1984; Harris, 1984) are another method for improving communication and understanding between human and computer. This method involves a computer program that takes English (or some other human language) requests from a user, translates them into a language that the computer can understand, and then converts the results back into human langauge for the user.

On-Line Help

On-line, embedded help programs have become "standard equipment" in many computer applications. Usually, the on-line material comprises a subset of a larger group of printed reference manuals. The trend seems to favor inclusion of both in a software package. There are advantages to both on-line and off-line materials. The first advantage to on-line help programs is the very fact of their being on-line. Reference manuals take up physical space, and in the typical courseware developer's work environment, desk space is scarce. In most cases, on-line programs can be revised and distributed at less cost than republishing printed manuals. Computer software is constantly being updated; version numbers change frequently. Obviously, reference material must match the software it refers to. Typically, on-line help can be updated easier than off-line manuals.

Two Examples: PLATO and TICCIT

In the mid-1960s, the PLATO system was only a few years old. Anyone interested in developing lesson material for delivery on PLATO had to first understand machine and assembly language (Lyman, 1981). Eventually the TUTOR language and its set of editors was born. As the PLATO system grew, so did the number and complexity of TUTOR commands. The developers of PLATO at the University of Illinois realized the need for documenting TUTOR in an on-line help program. The result was a program called "Aids," still in use today. The Aids program (Control Data Corporation, 1981) allows PLATO authors to find definitions and explanations of most of the PLATO system features, and all of the TUTOR language commands. For example, if an author needed help on the -compute- command, Aids would offer a set of different examples of the command, an explanation of what it does, and a description of the effects of executing the command.

The TICCIT system (Merrill et al., 1979; Wilson, 1984), while maintaining a different philosophy and approach to developing courseware, as compared to PLATO, had a similar on-line "help" program available. The authoring environment on TICCIT has changed over the years. The original APT (Authoring Procedure for TICCIT) system grew into an authoring language called TAL, which has in the last five years evolved into ADAPT (Mudrick & Stone, 1984). ADAPT, on the current MicroTICCIT system, includes a set of author language commands which posses the same syntax and sequence attributes as those of PLATO's TUTOR language. "Advice," as the on-line help program is called on MicroTICCIT, basically serves as a reference for authors who need quick, concise information about a feature of the system.

Revising "Advice"

The ADAPT language is designed as a multi-level authoring system. The menu-based Level 1 of ADAPT permits novice authors to develop courseware with no programming skill. Level 2, while maintaining a menu format, is designed to provide a more complete set of tools, including some programming and syntax. At Level 3, advanced authors are able to use the authoring language directly, with access to all the powerful programming functions.

One of the problems with traditional on-line help programs in authoring environments such as those on PLATO and MicroTICCIT is the preponderance of text-based, page-turner type information. "Screenfuls" of text that have been transferred from a reference manual generally do little to help a struggling author. In regards to the student environment, it has long been argued in the field of CBT that transferring a textbook to the computer does not make for better instruction -- in fact, it is generally agreed that in such cases textbooks are still more effective.

It has been argued that books are a passive medium (Merrill, 1985), whereas computers should be highly interactive media. Making help sequences interactive and developing individualized, adaptive help sequences for authors is an extremely, usually prohibitively, costly endeavor. Current trends seem to indicate that creators of authoring systems are aware of such problems and are beginning to improve their systems.

The New Advice

In order to make on-line help truly helpful for all authors, regardless of skill level, we must spend a lot of effort designing interactive programs that ultimately become instructional courseware for the author. The Advice program within the ADAPT environment presents an interesting case. During the past two years, a large effort has been undertaken to rewrite the help programs for the entire ADAPT authoring system to make them more interactive. The first working version of this new Advice was recently released to users.

The Advice package now consists of three highly integrated components: a "course" full of lessons covering every ADAPT topic, a "TOPICS" program that gives authors access to the entire "course," and a "Where Am I" program which shows the author where he or she is in the editor.

The Advice Course

It was decided that the vast collection of Advice topic discussions use the same course structure that actual ADAPT courseware uses. The Course-Unit-Lesson-Segment structure inherent in the architecture of MicroTICCIT was implemented for the new Advice course. Topics were arranged in a hierarchy of related subjects. Lessons and segments were designed and developed like regular courseware.

Every author has a different reason for requesting help. Because of this, it is generally difficult to meet each author's needs without impeding another author's request. Authors press the ADVICE key for many reasons:

- to understand how to use a command or feature
- to understand terminology
- to find out what to do next
- to find out where they are and how to get to somewhere else.

Because of these and many other reasons for requesting help, the designers of Advice decided to provide authors initially with a menu of subtopics. Typically, when an author requests Advice about a feature of ADAPT the first thing he or she sees is a short description of what the feature is, followed by a menu of further information, such as a full description of the feature (including how to use it, when to use it, and where to use it), a set of examples of the feature, and a description and solution of possible error messages associated with the feature.

Authors may not actually need help on the feature from whence their Advice request originated. They may actually want help on a completely different part of the system. The Advice program allows authors to access the TOPICS and Where Am I programs from anywhere within the program.

TOPICS

The TOPICS option provides several different lists of Advice topics. The first lists, in alphabetical order, all topics relating to the ADAPT authoring environment. A second lists the same ADAPT topics, but in a hierarchical order. The third topic listing covers, in alphabetical order, all topics concerning MicroTICCIT's IBM-PC-based graphics editor. TOPICS allows authors to access any Advice segment at any time.

The PLATO system has a similar feature in its Aids program (Control Data Corporation, 1981). However, in Aids the author must supply the name of the topic he or she wishes to see: there is no one central location where all "Aids" topics are listed. The ADAPT designers chose to provide two lists, one alphabetical and one hierarchical, of all individual topics.

Where Am I

A common problem faced by computer users is knowing "where" they are in the program. Most systems fail to provide a sense of "where," in relation to everywhere else. The designers of Advice decided to borrow the "You Are Here" maps found in airports and shopping malls, and implement them as "Where Am I" in the authoring environment. Where Am I allows authors to find out exactly where they are in relation to everywhere else in the editor. This is accomplished using a series of high resolution graphical displays depicting maps of the editor. In addition, authors can mark (with a light pen or touch screen) any location on the map to instantly access information on that feature of the editor. A future option will allow authors to be able to move directly from one location to another using the Where Am I maps.

The Future of Authoring Environments

Much has been written about the future of authoring systems and the impact of techniques borrowed from the field of artificial intelligence (AI) (Dear, 1986; Kearsley, 1985; Tennyson and Park, 1984). There is a growing consensus among developers of CBT that more on-line pedagogical help is needed. Few on-line help programs have much to say on the instructional aspects of courseware. It is generally assumed that all instructional design considerations have been thought out before the programming stage. In most cases, however, there is no fine line between where the design ends and the development and implementation begin. It is hoped by many that future courseware development environments will exhibit a blending of "help program" and "programming tool" into one highly interactive system.

Conclusion

The techniques described in the "Advice" program on MicroTICCIT and the "Aids" program on PLATO can be applied to a wide variety of computer uses. Options like "Where Am I" and "TOPICS" give users a better understanding of the system and make the system more non-threatening.

In the CBT environment, the student and the author both have jobs to perform.. A student's job is to sit down in front of the computer and use it to get information, learn new concepts, and master new skills. To do this the student needs to first know how to use the system -- not just the external shell of log-ons and log-offs, but also the actual courseware itself. An author's job is to sit down in front of the computer and use its set of authoring tools to create instructional material for students. To do this an author first needs to know what tools are available and how to use them. With good, understandable on-line help and reference programs, we can accomplish these jobs sooner and more effectively.

REFERENCES

Control Data Corporation (1981) PLATO User's Guide.

Dear, B.L. (1986) Artificial Intelligence Techniques: Applications for Courseware Development. Educational Technology. 26(7),7-15.

Harris, L. (1984) Natural Language Simplifies Computer Access. Systems & Software, 3(1), 206-212.

Kearsley, G.P. (1985) Training for Tomorrow. Reading, Mass.: Addison-Wesley.

Locatis, C., & Carr., V. (1985) Selecting Authoring Systems. Journal of Computer-Based Instruction, 12(2), 28-33.

Lyman, E.R. (1981) PLATO Highlights Computer- based Education Research Laboratory, University of Illinois.

Merrill, M.D. (1985) Where is the Authoring in Authoring Systems? Journal of Computer-Based Instruction, 12(4), 90-96.

Merrill, M.D., Fletcher, K., & Schneider, E. TICCIT Englewood Cliffs, NJ: Educational Technology Publications.

Mudrick, D., & Stone, D. An Adaptive Authoring System for Computer-Based Instruction. Journal of Computer-Based Instruction, 11(3), 82-84.

Rich, E. Natural Language Interfaces. IEEE Computer, 17(9), 39-47.

Wilson, L. (1984) Presenting TICCIT: State-of-the-Art Computer-Based Instruction. Training Technology Journal, 1(2), 27-32.

ABOUT THE AUTHOR

Brian L. Dear is a Senior Courseware Developer at Hazeltine Corporation's Training Systems Center in Reston, Virginia. His current responsibility is developing the Advice program for the ADAPT authoring environment. Prior to working at Hazeltine, Mr. Dear was a PLATO Systems Analyst at the University of Maryland. He is a member of the Association for the Development of Computer-based Instructional Systems, the IEEE Computer Society, and the American Association for Artificial Intelligence.