DOCUMENT RESUME

ED 281 504 IR 012 621

AUTHOR Reeves, Thomas C.
TITLE Computer-Assisted Instruction: Authoring Languages. ERIC Digest.
INSTITUTION ERIC Clearinghouse on Information Resources, Syracuse, N.Y.
SPONS AGENCY Office of Educational Research and Improvement (ED), Washington, DC.
PUB DATE Dec 86
CONTRACT 400-85-0001
NOTE 4p.
AVAILABLE FROM ERIC/IR Clearinghouse, 030 Huntington Hall, Syracuse University, Syracuse, NY 13244-2340 (free while supply lasts).
PUB TYPE Guides - Non-Classroom Use (055) -- Information Analyses - ERIC Information Analysis Products (071)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Authoring Aids (Programing); *Computer Assisted Instruction; Costs; Instructional Development; *Programing Languages
IDENTIFIERS ERIC Digests; Transportability

ABSTRACT

One of the most perplexing tasks in producing computer-assisted instruction (CAI) is the authoring process. Authoring is generally defined as the process of turning the flowcharts, control algorithms, format sheets, and other documentation of a CAI program's design into computer code that will operationalize the simulation on the delivery system. Used in this sense, authoring is essentially a euphemism for programming or coding. The principle alternatives for the authoring process are a programming language (e.g., BASIC, Pascal, or C); an authoring language (e.g., PILOT-Plus, Coursewriter, or TUTOR); or an authoring system (e.g., QUEST, IMSATT, or TICCIT). Programming languages should not be considered for authoring complex CAI unless the development team includes professional systems level programmers. At their current stage of development, authoring systems must be scrutinized with care before use. Authoring languages afford the best compromise between the flexibility of programming languages and the ease-of-use of authoring systems. Specific advantages and disadvantages of each of these alternatives are outlined, and transportability and cost are identified as important issues regardless of which approach is selected. A series of steps to be followed when considering adoption of a CAI authoring approach is provided. (MES)

ERIC Clearinghouse
on Information Resources

Syracuse University • School of Education
Syracuse, New York 13244-2340
(315) 423-3640

ERIC DIGEST

COMPUTER-ASSISTED INSTRUCTION: AUTHORING LANGUAGES

What is authoring?

Computer-assisted instruction (CAI) is generally produced through application of a systematic approach to instructional development, such as the Instructional Systems Design (ISD) model (Gagne and Briggs, 1979). Any instructional development approach includes many stages from initial needs assessment to summative evaluation, but one of the most perplexing tasks is the "authoring" process. Authoring is generally defined as the process of turning the flowcharts, control algorithms, format sheets, and other documentation of a CAI program's design into computer code that will operationalize the simulation on the delivery system (Merrill, 1985). Used in this sense, authoring is essentially a euphemism for programming or coding.

What are the alternative approaches to authoring?

There are three principal alternatives for the authoring process:

(1) a programming language (e.g., BASIC, Pascal, C),
(2) an authoring language (e.g., PILOT-Plus, Coursewriter, TUTOR), or
(3) an authoring system (e.g., QUEST, IMSATT, TICCIT).

Programming languages such as C and Pascal offer the advantages of faster processing time and considerable flexibility when compared to CAI authoring languages and authoring systems. But they have the disadvantage of requiring more time-consuming and expensive coding efforts than the other approaches. Most programming languages are written in whole words rather than in the numbers or mnemonics found in machine and assembly languages, but these languages do not produce code resembling "prose." In most cases, considerable effort is required to attain expertise in these languages. Programming languages are also somewhat easier to transport from one hardware delivery system to another than are authoring languages and authoring systems. However, CAI authored in a programming language does not automatically operate on different computers which claim to run the same language, and some additional coding may be necessary.

Until recently, BASIC was the most frequently used programming language in the authoring of CAI, but many of the most recent CAI programs are authored in Pascal or C. While BASIC is initially easier to learn than Pascal and C, it is difficult to manage long programs of the type required to run sophisticated CAI tutorials and simulations with BASIC, tasks that the structured nature of Pascal and C handle more readily. At the same time that Pascal and C are gaining popularity among those using programming languages to author CAI, indications are that newer, so called "artificial intelligence" (AI) languages, e.g., FORTH, PROLOG, and SMALLTALK, are being explored as potential programming languages to author CAI (Dear, 1986). These newest programming languages offer such advantages as easy construction of windows, icons, and other tools for enhancing a CAI program's user interface, and they also may enable designers to include natural language processors, voice recognition, expert systems, and knowledge bases in CAI.

Authoring languages such as PILOT-Plus and Coursewriter offer a middle ground between programming language and authoring system approaches to coding CAI. Authoring languages are interpreted languages, i.e., they are usually written in programming languages such as Pascal and C. Therefore, an instructional interaction written in an authoring language must go through several levels of translation before the desired interaction occurs: from the authoring language to the programming language, from the programming language to the machine language, and finally from the machine language to the electronic signals that actually operate the computer. All these translations take place very rapidly, but there is a cumulative effect and noticeable lag time can occur in complex CAI programs.

Merrill (1985) provided a thorough critique of authoring languages including an analysis of the pros and cons of this middle ground approach to authoring interactive instructional programs. The one real advantage of authoring languages is their inclusion of special purpose subroutines that facilitate instructional interactions. Merrill debunked some of the other supposed advantages of authoring languages such as the claim that they are easier to learn than programming languages. He concluded that the skill and practice required to make authoring languages perform certain complex instructional interactions often exceed the expertise possessed by professional programmers. He also critiqued authoring languages for high demands on disk space and memory overhead, retarded execution speed, frequent disk access, and the high royalties sometimes associated with their use.

Romiszowski (1986) lists the following as desirable features of authoring languages:

- Easy design of text messages using spacing, multiple font styles and sizes, color, etc.
- Acceptance of multiple types of user responses, e.g., keyboard, light pen, touch screen, mouse, track ball, etc.
- Accurate and fast analysis of user responses including sophisticated interpretation of misspellings, synonyms, etc.
- Efficient record keeping and management of response data for purposes of student guidance, formative evaluation, etc.
- Accurate and fast branching based on user responses to feedback, alternative instructional sequences, help, databases, etc.
- Creation of high resolution graphics or access to such graphics which have been created with other programs.
- Expanded color palette for design of foreground and background materials including overlay on video signals.
- Creation of sound effects and synthesized voice patterns for audio enhancement of instructional materials.

- Options for building new command subroutines into the authoring language to handle unique instructional requirements.
- Easy access to instructional sequences and other programs (e.g., a calculator) written in programming languages such as C or Pascal.
- Test generation facility including the maintenance of item banks and generation of random samples of questions.
- Options for integrating peripheral hardware such as videodiscs, CD-ROM, etc.

There do not appear to be any existing authoring languages that include all the options noted above to their maximum potential, and therefore, anyone considering adoption of an authoring language for authoring complex CAI should look for the best possible mix of the features.

Authoring languages, especially derivations of PILOT (Starkweather, 1986), enjoy considerable popularity in the authoring of all types of CAI programs. (There is even a PILOT Special Interest Group within the Association for the Development of Computer-Based Instructional Systems (ADCIS), the largest professional society in this field.) In many interactive development contexts, authoring languages may offer an attractive mix of the flexibility of programming languages and the facility of authoring systems.

The last few years have seen the development of computer-based **authoring systems** that designers can use online to more or less automatically generate code which will "run" CAI for delivery on specific hardware systems. Major advantages of authoring systems are that they eliminate the possibility of miscommunication between the design and programming stages and that they greatly decrease the costs for expensive programmers on a project development team. However, authoring systems are a source of controversy in the field of CAI. CAI journals frequently feature advertisements for these systems which seemingly promise to make anyone a superior CAI designer with a few simple keystrokes. (One, for example, features an image of a monkey and includes text which reads: "The _____ Authoring System makes computer programming...so simple, fast, and cheap that even a chimpanzee can do it!") At the same time, these same journals publish articles (cf., Merrill, 1985, in press) which criticize authoring systems for a host of shortcomings including:

- lack of flexibility,
- complex command structures,
- constraints on disc storage and memory,
- slow execution speed,
- difficult portability among delivery systems,
- costly royalties,
- limited instructional design options, and
- frame-oriented presentation and interaction.

The last two shortcomings are the most serious problems with respect to using CAI authoring systems. In order to produce an authoring system, the developer usually identifies a specific set of instructional design options or interactions which the system will allow (e.g., the ubiquitous multiple choice question). If the options included in the system are all those required for a particular CAI program, then there is considerable advantage in using the authoring system instead of the more difficult to learn and use programming or authoring languages. However, most CAI designers do not want to be limited to a predefined set of instructional options.

The most prevalent feature of existing authoring systems is the frame-oriented approach for instructional interaction, an approach derived from the programmed instruction popular in the 1960s. This frame-orientation can be traced to the logical control and branching limitations inherent in the paper-based origins of programmed instruction. Most authoring systems are very capable of effectively and efficiently implementing frame-oriented instruction, i.e., instruction in which a screen of text and/or graphics is presented, a question is asked (usually in multiple-choice format), and feedback as to the correctness of the learner's response is made before branching to the next frame of text/graphics. Most existing CAI programs have been designed and programmed in this way, resulting in the "electronic workbook" phenomenon for which interactive instruction has been severely (and justifiably) criticized.

Merrill (in press) provided a thorough analysis and critique of existing authoring systems and identified the following prescriptions for an authentic instructional authoring system:

- wide range of instructional interaction archetypes to serve as models for designers,
- options for combining instructional interaction archetypes into unique sequences of instruction,
- selection of content structures for organizing and accessing subject matter,
- tools to specify student management parameters and evaluation options,
- various levels of interaction with the authoring system including templates, parameter menus, and template creation,
- on-line guidance to the designer in the selection of goals, content structures, strategies, and learner management structures.

Unfortunately, no such authoring system exists.

What is the best approach to authoring?

Obviously, there is no simple answer to the question of which authoring approach is best. It is clear that programming languages should not be considered for authoring complex CAI unless the development team includes professional systems level programmers. Instructional designers, teachers, and subject matter experts should not expect to develop sufficient expertise to make the best use of a programming language for authoring CAI. At least one authority in the field, Bork (1985), argues persuasively for the exclusive use of high level programming languages for CAI authoring and thus, for the inclusion of professional programmers on CAI development teams.

Authoring languages may afford the best compromise between the flexibility of programming languages and the ease-of-use of authoring systems. This is particularly true of derivations of the PILOT authoring language such as PILOT-Plus (Ford, 1987). PILOT-based authoring languages provide a good mix of relatively easy to learn and use commands for CAI as well as the capacity to readily call in routines coded in programming languages for unique instructional interactions. For example, an interactive videodisc simulation that is primarily coded in an authoring language can contain a natural language processor authored in a programming language. The use of two authoring approaches can be invisible to the user.

At their current stage of development, authoring systems must be scrutinized with great care before adoption for complex CAI development efforts. One recent project, involving the development of many hours of interactive courseware, was originally intended to be authored with a prototype authoring system. However, when it was discovered that the authoring system's inefficient method of processing graphics would require each user station to have a 30 megabyte hard disk capacity, the authoring was switched to a PILOT-based authoring language (Reeves & King, 1986). Of course, this does not mean that developers should abandon consideration of authoring systems for future applications. All three authoring approaches are undergoing exceptional growth and development, and newer and better approaches seem to be announced every week. For example, one new authoring system, IMSATT (Kannan, 1986), represents an attempt to incorporate artificial intelligence features into an authoring structure.

Developers evaluating any authoring approach must be concerned with the issue of transportability (i.e., on which hardware delivery systems can a given programming language, authoring language, or authoring system be used?), and the issue

of costs (i.e., what are the initial investments and distribution royalties involved in use of the approach?) Most CAI developers desire their simulations to enjoy as wide a use as possible and will therefore want their authoring software to run on as many computer systems as possible. The transportability issue is complicated because users must consider more than just the computer used in the delivery system; every additional card or peripheral device (e.g., graphics cards or videodisc players) must be factored into the compatibility evaluation. Two CAI delivery systems can be identical in every way except the graphics cards, and therefore not run a particular program without considerable program modification.

With respect to costs, authoring approaches range from being free "public domain" software to programs costing many thousands of dollars. More importantly, some authoring approaches require run time licenses of $500 or more per station. In other words, if someone wants to buy a CAI program, he/she may also need to buy the right to run the authoring software used to develop the program. Licensing agreements are very complex, but many vendors will work out special agreements for specific applications. There are also public domain and commercial authoring approaches which entail no licensing fees.

Final Recommendations

Given the lack of an "ideal" programming language, authoring language, or authoring system, selecting the "best possible" authoring approach for development of CAI is dependent upon identification of the types of instructional treatments and instructional interactions to be included in the programs as well as certain other factors (e.g., portability, royalties, etc.). If a particular development context possesses no programming expertise (or the motivation or time to develop such expertise), an authoring system may be desirable as long as it is robust enough to implement the desired instructional options without undue memory constraints and/or retarded execution speed. However, if an authoring system is appropriate, the CAI designers must be able to learn to use the system commands easily, the system should be transportable to a number of desirable delivery systems, and it should not entail costly royalties.

If no authoring system can be found that possesses the characteristics desirable for developing CAI of a specified complexity, it may be necessary to identify a mix of authoring and programming languages. Obviously, the whole question of the best possible authoring approach is related to the choice of CAI delivery system(s). Some hardware delivery systems include authoring language and system options while others exist for which no authoring options have been developed. In the final analysis, while the availability of an authoring approach will influence the choice of delivery hardware, the hardware question must be answered before a final decision can be made about authoring approaches.

Anyone considering adoption of a CAI authoring approach is advised to follow these steps:

- Find out if the authoring approach has been used for CAI in your field or related fields. Contact the users and question them concerning their experience with the approach.
- Require the vendor to delineate exactly which combinations of computers, monitors, extension cards, etc. will allow application (authoring and delivery) of the approach.
- Require the vendor to specify all present and future costs involved in authoring and distributing authored CAI. Consider both the initial investment and any licensing fees.
- Evaluate the system against the criteria stated above by Merrill (1985) and, most importantly, in reference to the types of objectives and interactions desired for your CAI.
- Evaluate each approach in comparison to other authoring approaches.
- Negotiate a test period with the authoring approach so that it can be evaluated in the context of your actual operation.
- Investigate provisions for user support by the vendor or users' groups. Find out if the primary developer(s) of the authoring approach is still involved with it so that bugs can be eliminated or enhancements added.

Obviously, not every vendor of authoring approaches will be willing to cooperate in such a systematic evaluation, and anyone considering authoring systems should be wary of systems that promise "instant success" or "no effort necessary." There is no such thing as an effortless approach to authoring CAI. Users must also consider the scale of their CAI development and the scope of its distribution. If relatively simple tutorial or drill-and-practice programs are to be developed to enhance teaching in a local setting, an inexpensive or public domain authoring approach may suffice. On the other hand, developers of complex CAI programs intended for commercial marketing will do well to carry out the type of extensive evaluation outlined above.

References:

Bork, A. (1985). *Personal computers for education.* New York: Harper & Row.

Dear, B. L. (1986). Artificial intelligence techniques: Applications for courseware development. *Educational Technology, 26*(7), 7-15.

Ford, W. H. (1987, January). *An integrated authoring environment for optical videodisc, CD-ROM, and CD-I.* Paper presented at the Outlook for Compact and Video Disc Systems and Applications Conference, The Institute for Graphic Communication, Key Biscayne, FL.

Gagne, R. M. & Briggs, L. J. (1979). *Principles of instructional design* (2nd ed.). New York: Holt, Rinehart, and Winston.

Kannan, N. (1986, July). The true power of interactive video: The role of the authoring language. *The Videodisc Monitor,* 18-19.

Merrill, M. D. (in press). Prescription for an authoring system. *Journal of Computer-Based Instruction.*

Merrill, M. D. (1985). Where is the authoring in authoring systems? *Journal of Computer-Based Instruction, 12,* 90-96.

Reeves, T. C., & King, J. M. (1986, November). *Design considerations for an interactive videodisc system to teach adult literacy.* Paper presented at the 28th International Computer-Based Education and Training Conference, Washington, DC.

Romiszowski, A. J. (1986). *Developing auto-instructional systems.* New York: Nichols.

Starkweather (1985). *A user's guide to PILOT.* Englewood Cliffs, NJ: Prentice-Hall.

Organizations:

Association for the Development of Computer-Based Instructional Systems, Gordon Hayes, 409 Miller Hall, Western Washington University, Bellingham, WA 98225, (206) 676-2860.

SIG PILOT (ADCIS), W. Kirk Richardson, University Plaza, Georgia State University, Atlanta, GA 30303, (404) 658-2283.

Lister Hill National Center for Biomedical Communications, Craig N. Locatis, National Library of Medicine, Bethesda, MD 20209, (301) 496-6280.

Prepared by Thomas C. Reeves, Instructional Technology, College of Education, The University of Georgia, 607 Aderhold Hall, Athens, GA 30602, (404) 542-3810. December 1986.

OERI Office of Educational Research and Improvement US Department of Education

This publication was prepared with funding from the Office of Educational Research and Improvement, U.S. Department of Education, under contract no. 400-85-0001. The opinions expressed in this report do not necessarily reflect the positions or policies of OERI or ED.