A Suggested Model For Development of Computer Assisted Instruction For Higher Education

By
Terry Dixon
October 11, 1984

A Suggested Model For Development of Computer Assisted Instruction For Higher Education

By
Terry Dixon
October 11, 1984

Introduction

The microcomputer is a powerful tool capable of many things which can benefit the instructional process. When chosen to be used to communicate or teach the appropriate objectives through Computer Assisted Instruction, it becomes a powerful tool, one which offers characteristics unsurpassed by other media tools. The power of the Microcomputer, as an instructional tool, comes from six basic features. These include the ability to; analyze and prescribe (systematize); provide active involvement in learning; allow the student to pace the instruction; provide exploration of time and space; free teachers for human and personal work; provide instruction at a cost effective level. To summarize, this means the power of the computer comes from the fact it is possible to individualize instruction to the nth degree, i.e. it is possible to evaluate each and every student response in detail, then alter following instruction to best fit the particular student's needs, whether it be a change in method of presentation, or progression through the lesson. Unfortunately the very characteristics which provides the powerful potential for CAI, also poses a problem. How do you aquire quality software which is objective specific to a particular classroom teachers needs?

One would think it possible to purchase such software from the various educational material supply companies, and indeed it is possible to purchase a large variety of CAI software. However, the software available is usually very general in nature and its quality is usually questionable. As William Montaque stated in his paper presentation at the 1982 annual meeting of The American Educational Research Association, "The most interesting and potentially important things a computer can do are seldom utilized." If quality, objective specific software is hard or impossible to find, then what option is left for the teacher who desires to incorporate and take advantage of the potential of microcomputer instruction?

The purpose of this response is to provide one such option, a model for CAI development for the classroom teacher. It would be foolish to think all teachers desire. or have the capacity to design their own CAI software. However by providing a model which encourages the development of local, effective and educationally sound software, perhaps the quality and quantity of CAI software will increase.

Because of the classroom teacher's limited time it is necessary to keep the model as simple as possible, and yet include those techniques and suggestions research has shown to increase effectiveness. This paper will attempt to synthesize a model which meets these requirements.

Need

Why anotner model for CAI development? The Association for Educational Communications and Technology identified four issues related to the development of software technology. These included; a lack of adequate theory base supporting the discipline of instructional technology; too limiting and narrow definition of the field; a lack of courage among instructional technologies to stand up and take a strong position influencing the future of training and instruction; and, inadequate models which are too unsophisticated to be viewed as analogs for designing quality software systems. (De Bloois, 1982). This suggest the need for CAI design models which are usable, incorporate sound instructional theories and from which more models with further improvements can be developed. It also suggest CAI software development should be viewed as evolutonary, no model being final, but simply one of a series of models, which allows designers to get closer to perfection, and to share knowledge in the technologies of other software designers who may bring expertise from various other areas.

Martin and Davis state:

> "We must pass through a period in which the new medium befuddles us, in which it is applied intuitively to familiar tasks and ways of doing things. The educational computer is doomed, in other words, to spend some time imitating teachers and books and, because the copy is never as good as the original, to do their tasks less effectively." (Martin & Davis, 1983).

This would clearly indicate a need to go beyond the traditional paradigm and begin to explore the new attributes which the computer brings to instruction. Before the development of the computer and CAI it was not feasible to individualize instruction to the extent possible now, yet as we take a close look at the software on the market we find few, if any, taking advantage of this computer attribute. This is probably accounted for by the fact CAI software, as Robert Burke states in his <u>CAI Sourcebook</u> (1982):

"... is a very complex thing to produce, since several

> highly sophisticated technologies must be integrated in such a way as to produce an organically complete entity which is capable of achieving a specific result. The CAI author must draw upon several fields of knowledge to put together an effective lesson."

It would take a book, and indeed many have been written, to include a study of all of the variables associated with successful CAI. Abraham Koplan sates in the text, The Conduct of Inquiry:Methodology for Behavioral Science:

> "Models never include all possible elements of a process, but are selective. A model generally includes those elements which seem to occur with the process in ways which fit the particular situation being examined."(Koplan, 1964)

It is not the purpose of this response to provide such a book. It is the purpose of this response to synthesize, through a survey of research and readings, as well as experience, a model for designing CAI. A model which is simple in design, yet rich in its incorporation of instructional attributes which are best suited for the microcomputer.

A Suggested Model for Computer Assisted Instructional Design

Where do models come from? Many models used in the design of software today have their roots deeply planted in the fields of engineering, whether the software to be produced is CAI, business software, or a video game. They have provided many techniques valuable to the development CAI models. Emphasis in these models concerns controlling the development of the software in such a way as to insure the proposed end product and to decrease the time and cost of producing the software. Because they provide guides for quality control and techniques for incorporating "later improvements" during the development of software, a breif discussion of the more common models is beneficial to the CAI designer.

Common Software Designs Models

Chu, in his text, Software Blueprint and Examples , mentions several of the more common software design models found today (Chu, 1982). These include the Composite or Structural Design Model, the Jackson Model the META Stepwise Refinement Model and the higher-Order Software Model.

The Composite or Structural Design Model

The Composite or Structural Design Model was described by G.J. Meyers in 1973 and consists of a discussion of the underpinnings of a good design and methods of module decomposition. Since it does not include an underlying mathematical theorem it is not very appealing to many computer scientists. Major components of the Composite Design Model are the discussions of partitioning, hierarchy and independence, as related to module design.

The Jackson Model

The Jackson model, developed by M.A. Jackson in 1976, is popular in England. It views a program as the means by which the input data are transformed into output data. The proper choice of the data structure of a program is supposedly to lead to a good design.

The META Stepwise Refinement Model of Design

H. F. Ledgard, 1973, and B. Schneidman, 1976, first described this method. It is based on the experience of many designers who beleive a better design can be acheived by successfully refining a simple design until the design is at the desired level of detail. The program is designed in levels and the details are postponed to lower levels.

Many advanced forms of this methodology have been developed. and indeed this is probably the most used method of software design. Because it decomposes a program into digestable pieces it provides the most potential for teacher CAI design.

The High-Order Software Model

The Higher-Order software Model was initially developed and promoted by M. Hamilton and S. Zeldin in 1976. It involves a set of formal laws and a specific language to assist in the development of software. It is based on a set of axioms which explicitly define a hierarchy of software control flow. An automated analyzer program is available which checks the solution to the design once it is finished, if written in metalanguage. It has similar qualities to PILOT, a CAI authoring program written for classroom teachers.

From these engineering models many valuable design principles can be developed, and when properly incorporated into a CAI design model, can improve their effectiveness. However, models are synthesized from basic accepted assumptions or principles and to understand the models it becomes necessary to have a clear understanding of the basic principles involved.

Chu describes four design principles which encourage the design of "good" software. These are the principles of: Modularity; Abstraction; Localization; and Hiding.

Modularity

Modularity, considered the best-known principle, is the decomposition of the data flow and control flow of a software organization into well defined modules and their interfaces. Modules, according to this principle, should be structured in a hierachial structure to allow the designer to easily visualize and understand the organization. It is also important to have a high degree of independence from other modules so as to encourage a low degree of interaction between data and modules.

Modularity contributes greatly to the attainment of the goals of a program in that it simplifies understanding and reliability, since errors can be localized into modules. It also contributes to modifiability since a change may only affect a few modules.

Abstraction

Abstraction is the removal of essential properties, while implying essential details. Each level of hierarchy presents an abstraction of those lower in that details are restricted to the lower levels. The larger and more complex the design, the more valuable this principle becomes.

Localization

When related software elements are physically placed close to each other it is referred to as localization. Avoiding the use of 'goto' statements in a program assist in accomplishing localization.

Hiding

The hiding principle is similar to the principle of abstraction in that details are delegated to lower level procedures. All details of a module and its task are described within the module for easy accessibility. Hiding makes the program easy to follow and customize since the information concerning the function and variables used within a module are kept within each module.

A Sugested Computer Assisted Instructional Design Model

The design model which follows is based on the philosophy of the META Stepwise Refinement Model described earlier. It goes from the general to the specific and is designed in such a way as to encourage efficiency and simplicity in designing CAI. The model is composed of 5 phases: Problem Clarification; System Design; Blueprinting; CAI Synthesis; and Documentation Development. Each of these phases is then subdivided into two or more subphases.

In this particular model there are two reasons for dissecting the model into phases. First the phases allow processes which are similar in task result to be grouped together, thereby providing a focal point for mental concentration. Without these logical divisions it becomes difficult to keep the processes effecting each task in mind.

The second, and probably most important purpose of the use of phases, at least to classroom teachers, is that it allows the CAI to be developed during collections of short sittings over a long period of time, rather than requiring one or two long sessions, sessions which are not readily available to the classroom teacher, while at the same time not sacrificing the quality of the software.

Phase I: Problem Clarification

The problem Clarification Phase is composed of three tasks:objective development; content research, and: narrative synthesis. It's basic task is to verbally describe exactly what the software is to accomplish, actually listing the expected outcomes or products of the CAI program, and to determine the necessary content needed to accomplish these outcomes.

OBJECTIVE DEVELOPMENT refers to the development of the instructional purpose of the CAI. It is developed just as any other instructional objective would be developed, and should be stated in behavioral form, incorporating the terminal behavior expected of the student, criterion for determining successful attainment of the terminal behavior. and the conditions under which the terminal behavior is expected to occur. At this stage the objective would look no different than any other objective since the best method of instruction has not been determined.

When developing the objectives it is important to keep the student population in mind and to know their abilities and characteristics so as to not have expectancies of the students which are for any reason impossible for them to accomplish, or spend time developing objectives which the students have already accomplished. One of the most common mistakes made in CAI software development today is the development of software for certain age and ability groups in a vacuum, i.e. development of software without even studing the characteristics of the student population. This should be the strength of the teacher authored CAI, since this is the area of expertise the teacher brings.

The process of objective development is the starting point for the development of CAI, and the success of the CAI can be no better then the objectives for which it is used, therefore plenty of time should be set aside to develop objectives and to determine the capabilities of the student for any of the objectives. If careful thought is not used in their development many hours in the development of the CAI may be wasted when the author finds later the objectives are inappropriate for the student population. It is even a good idea to develope a traditional instructional method and pilot test it on the student population to determine the appropriateness of the objectives before devoting the time and effort in the development of CAI. In this way objectives can be altered, added or dropped based on information from the pilot study, before many hours are spent on the development of CAI to teach these objectives.

Once the objective has been determined the CAI author is able to determine the general content necessary to accomplish the behavioral objectives. This will then suggest the mode of instruction, i.e. tutorial, drill and practice, simulation, or problem solving. At this stage it is necessary for the CAI author to have a clear understanding of the characteristics of each of the computer modes of instruction as well as the instructional characteristics and capacities of the computer, otherwise the computer may be expected to accomplish a task which it is physically incapable of accomplishing.

The analyzing tools (pre/post-test) should also be suggested by the objectives at this stage, though until the specific content is determined the tools should not be developed. It is at this stage the determination of whether the computer is the appropriate method for instruction is decided, based on the behavioral objectives. if computers are determined to be appropriate, then it may be necessary to alter the method of analyzing the success of specific objectives so as to take advantage of a computer characteristic. It is at this point where the instruction begins to appear computer specific, i.e. the characteristics of the computer begin to play a part in shaping the instruction.

Reinforcement should also be decided at this stage, since the mode of instruction will hav a bearing on the method, form and appropriateness of the reinforce ment to be used. For example it may be inappropriate to reinforce a drill and practice with a pause in the CAI and introduction of a lengthy song as a reward for correctly responding to each of the drill and practice problems presented, since it would interrupt the students' being on task.

With the objective development complete we have a firm foundation for further CAI development. The stating of the objectives leads to the need for content research.

CONTENT RESEARCH refers to the determination of what content is necessary to accomplish the objectives developed in stage 1. It also involves decision making in terms of determining instructional time each specific piece of content is to be allowed in the completed CAI.

This stage should also include any content oriented motivational material, i.e. any material which would give purpose or develop the desire of the student to want to accomplish the instructional objectives and content of the instruction. Content oriented material refers to subject matter, and not reinforcement material. For example if your CAI was concerning chemistry, then you might introduce the instruction on the computer by stating, "Chemistry is important in all of our lives. In the 1984 world series chemistry was the hero in saving the Cincinnati Reds from losing. As we complete this program we will answer the question how did chemistry saved the Cincinnati Reds from losing the 1984 world series."

The content research should be written as if it were a paper to be read or presented to the student target population. The emphasis at this stage should be on content, though reading ability and vocabulary should also be considered. The content research will become the story line of the CAI, therefore it is important to write the reserach in a report form, as opposed to a collection of facts which have not yet been tied together into a story line.

The type of CAI used will also affect the type of content research which is necessary. For example a drill and practice CAI for multiplication tables will require research on the types of possible incorrect answers, and the most commonly missed problems, so CAI time will not be monopolized teaching problems already learned and which can be controlled by the computer if we instruct it to do so.

Once the content research has been completed the narrative can be develoed. The narrative is a written description of what the CAI will do, its' emphasis is on outcome. It should be written as if describing the outcomes of the CAI, i.e. it should describe any product, whether it be hardcopy, screen display or peripheral products. For example, "The math problem generator will, after taking in the teacher criteria, produce hard copy worksheets for the student and answer sheets for the teacher."

NARRATIVE DEVELOPMENT occurs once the content research has been completed. The narrative is a written description of what the CAI will do. Its' emphasis is on outcome and should be complete, but consise and brief. It should be written as if describing the outcomes of the CAI, i.e. it should describe what resources the CAI will begin with and any product, whether it be hardcopy, screen display or peripheral products which will be produced. For example, "The math problem generator will, after taking in the teacher criteria, produce hard copy worksheets for the student and answer sheets for the teacher."

The narrative does not describe the necessary processes for accomplishing the products, but simply describes the resources which will be given the CAI and the product outcomes expected of it. This stage tells the CAI author when he has completed his CAI program, so it is very important in terms of a mile marker. Without using the narrative as a mile marker the author becomes bogged down into adding this or that process to make the program "better" instead of to accopmplish the original purpose, often causing author frustration. This is not to say the program cannot be altered along the way, but it is to say the narrative can be used as a decision guide to determine if a new process should be added or not.

The narrative should view reinforcers as a product and for this reason the branching strategy for correct and incorrect answers, as well as the planned reinforcers should be described in this stage. Describing the branching refers to the general description, not the computer code. For example, "Upon reponding correctly to the question the computer will display a smiling face to inform the student of a correct response. If the student response is incorrect, the computer will display the phrase "incorrect", then allow the student to respond once again."

The narrative should be sequential in development, i.e. should follow the flow of the program so as to simplify the second phase of CAI development, the System Design Phase.

Phase II: System Design

The purpose of System Design is to prepare the instruction for computer coding. This phase takes the narrative and content of the previous phase and combines it with the necessary computer requirements so as to accomplish the expected narrative. The emphasis in this phase centers around designing the processes (or modules necessary to accomplish the processes) necessary for the computer to cause the narrative to come about. The descriptions are general in nature, but include all the necessary tasks each module must complete.

The MODULARIZATION STAGE is where determination of the various modules necessary to accomplish the narrative described in phase I is accomplished. It is accomplished by taking the narrative and dividing it into large system modules which accomplish the various tasks necessary to accomplish the narravitve. This is accomplished by physically drawing boxes around each logical division of processes described in the narrative. At this stage it is important to concentrate on the "global modules" as opposed to "detail modules". For example you should be concentrating on "takes in the names of the students", as opposed to, "the steps necessary to tell the computer to take in the names". It should not answer "how", but should answer "what".

This stage is also where the development of "hidden modules" are developed. "hidden modules" are those modules which do not themselves produce a product which can be viewed or heard, but are used to evaluate answers, involve decision branching, or prepare or transform information for display or for further processing. An example might be an answer checking routine which would check a students' answer to see if it were correct and then send it to the proper subroutine based on whether the answer was right or wrong. This routine would not really produce a product on the screen, but is very necessary to the successful completion of the narrative and the CAI.

It is also the stage where debugging routines for later use can be designed to simpify debugging once the program has been encoded. For example each module should have an identifying number or letter so problems can be traced. It is also a good idea to place frame numbers on the screen when each frame is displayed so problems in various frames can be easily traced through the listing. By designing a system of debugging such as using a variable such as dt for debugging trace at the beginning of the program being developed, and placing a statement such as 10 IF DT=1 THEN HTAB 38:VTAB 24: PRINT "DT=1". This can save many hours later, just as all the designing phases of CAI development.

Following modularization the author will have accumulated a written list of the necessary modules to cause the narrative to occur. It then becomes necessary to sequence these modules through flow charting.

FLOW CHARTING involves the designing of the relationship of each of the modules to each other. It necessarily involves more detail than the modularization of the narrative, since it includes the division of the narrative modules into functional sub-modules which prepare or alter information in preparation for the next module. Emphasis is on the relationship and sequence of each module to other modules as opposed to the overall process of a module.

Each of the modules should be labeled for easy tracing later and should be consistantly carried throughout the development and encoding process. Otherwise when debugging it will be more difficult to trace a problem to the proper module.

The flow chart should be a graphic representation showing the relationship of the various modules to each other, however a detailed description of the function of each module should be written at this stage to be used by the author in later programming. Without a flow chart and a description of each of the functions each module accomplishes the author is likely to become confused when later programming each module. This stage is also important if another person is going to be doing the programming of the CAI. It provides him a guide for encoding the various functions.

Following flow charting a list of all the modules, in the proper sequence and with the proper sub-modules will be displayed schematically and described verbally in written form. ready for blueprinting.

Phase III: BLUEPRINTING

Blueprinting involves the development of a detailed description of the CAI from frame to frame and function to function. It is the last phase where the descriptions are not machine specific, i.e. following this stage all design will only work on a certain computer and may not be readily changed to another machine.

Blueprinting involves two stages of development in this module: Frame development, and; frame design.

FRAME DEVELOPMENT refers to the functionalization of the system modules. This involves determination of what functions have to occur for each module in order for it to carryout its' function. Emphasis is on detail sub-modules specific in function. This stage also involves the amalgamation of the content into the function of the modules, i.e. the content of each frame will necessarily be matched with specific modules for their display. Earlier emphasis was concerned with designing a module which would display the content. Now we are concerned with what modules will be needed to display each frame, as well as what frames are needed, and in what sequence for each frame.

A distinction needs to be made here concerning the modularization stage and the frame development stage. The modularization stage concentrates on the tasks to be completed to get the <u>narratve</u> accomplished, whereas the frame development phases concentrates on accomplishing the <u>objectives</u> through the presentation of the content. The frame development stage should involve a brief written description of each of the various frames selected for the CAI, and should be written as if the author was expecting another person to program the frame, which means all the detail necessary to get the message across must be included. The frames should be numbered according to the planned sequence.

It is also in this phase where the specific pre/post-test analysis is developed and treated much like the other frames within the CAI program. It should be designed so as to allow the student who has demonstrated, through pre-test, he has accomplished an objective, to pass by instruction dealing with the objectives he has already accomplished.

The post-test should provide information to the student concerning his status on the test and what he needs to work on in order to successfully complete the post-test the next time. If the CAI you are developing will also keep records of students progress, then this will also need to be taken into consideration at this stage.

FRAME DESIGN refers to the actual 2-dimensional design, on paper, of the frame which will appear on the screen, hence a "blueprint" to guide the programmer is developed for each screen. It is important in this phase to be aware of the various learning theories affecting the display and presentation of material to the student target population. An understanding and familiarity with 2-dimensional design theory is also important, in that the way you layout the screen may determine the successs of your CAI.

In this stage it is also important to be aware of the characteristics of the computer you plan to use so you may incorporate the various sense generating devices available on the computer you choose. For example a computer which does not have sound should not be chosen for a CAI program which teaches music.

Frame design is acomplished by using story boards to display drawings of how the frames will appear when completed. Story boards are drawings, in sequence, of the various displays which will be presented on the screen. They allow the author to view how the screen will appear before coding and thereby easily alter any frame deemed necessary without reprogramming the computer. Ideas can be gathered by viewing other software and determining how effective their frames seem to be.

It is suggested all frames be divided into various functional areas so students will not be confused as they move from frame to frame throughout the CAI. One area of the screen may be reserved for directions, another for instructions and perhaps another for titling so the student will be aware of their position within the CAI.

Many computer publishers offer "worksheets" to assist in the design of frames. The worksheets are layout sheets with the necessary display information printed on them and allow the author to design and record important information specific to each frame on each worksheet, simplifyng the design and recording of organizational information. The worksheets have a screen drawn on them in which the dimensions have the same relationship as the actual computer screen, thereby allowing the author to view and make changes on a "screen" very similar to the computer screen before being coded into the computer, where change is much more difficult to accomplish.

Phase IV: CAI Synthesis

CAI Synthesis refers to the actual encoding of the computer, and once the author begins in this stage the CAI becomes machine specific, i.e. it is written to control one computer. There are three stages in CAI Synthesis: Encoding; Debugging, and; Evaluation.

ENCODING is the process of programming a specific computer to accomplish the various tasks described earlier. The encoding should include remark statements to assist in debugging or updating the program at a later date, and should follow the blueprint developed earlier.

Encoding begins by taking the blueprint and programming each module of the program to accomplish the results described by the blueprint. It is important to follow the blueprint as it provides the decision information for determining the completion of encoding for each frame and eventually each module.

It should also be remembered to incorporate the debugging routines mentioned earlier into the encoding to simplify debugging in the next stage. It becomes very difficult in a long program to determine your location within a listing once you have spotted a program error unless you have labeled each frame and used remark statements throughout your programming which matches the labels printed on each frame.

As each module is completed, if possible, it should be tested for errors and these corrected. Once the complete programm has been encoded you are ready to debug the program.

DEBUGGING refers to the identification and correction of errors, either programming, logic, or typographical errors. It is accomplished by running the CAI through all the possible branching routines and determining if there are any errors. Almost all CAI programs will have errors in them when they are first encoded because of the demanding detail of instructions which computers require in order for them to carry out various tasks. Therefore it is very important once you have debugged the program, that you have other people try the program so they may discover bugs which you have missed. It is even a good idea to pilot test the CAI before putting it in full use. Many problems can be caused by not properly testing CAI before placing it in full use.

EVALUATION refers to the critiquing of the program by the author and other collegues to determine its' success in accompishing the objectives developed in the first phase. Suggestions may range from changes in screen design, to changes in frame layout and instructional presentation.

It becomes very difficult to change a CAI program at this stage, and because of this all efforts should be taken to correct or alter CAI before reaching this stage. This may involved the opinion of other collegues during each stage of development.

Generally when evaluating a program it is a good idea to check for spelling errors, missing frames, poorly designed frames, illogical frames and instructional weaknesses. It is best to make notes concerning these errors, recording the frame number and the error which you noticed, then going back into the listing to make the necessary changes.

After completion of this phase all that is left to be completed is the Documentation Development.

Phase V: Documentation Development

The Documentation Development Phase involves the development of manuals, and other technical data for the use of the CAI. It involves the listing of system requirements and development of a manual.

SYSTEM REQUIREMENT refers to the listing of the necessary hardware requirements at the beginning of the CAI so as to allow the user to acquire the necessary equipment before using the CAI. Thi should also be included in the CAI instructional manual.

Generally, the information included in system requirements includes: Type of computer, amount of storage capacity needed, necessary peripherals and input devices. It might also inform the user of specific requirements when responding to the CAI.

MANUAL DEVELOPMENT refers to the development of a manual which describes the system requirements of the CAI, as well as the instructions for the CAI use. It should be written keeping the user in mind and use schematics whenever they might simplify explanations concerning the CAI.

The blueprint should be used as a guide for describing the procedure for using the CAI. It should start with instructions for loading the progrm into the computer and then proceed through the program explaining what occurs at each stage. It should also mention any specific peculiarities of the program and contain a list of keywords, if appropriate, used withnin the program.

The manual should be divided according to the functions within the program so the user will not have to read the whole manual to find the answer to a specific question he may have concerning a certain function. The explanations should avoid Computer jargon and be written in plain English with an emphasis on understanding.

Summary

This then is a suggested model for Computer Assisted Instruction. It incorporates the designing stategies of the engineer and the instructional strategies of the classroom teacher, as well as encoraging the use of layout, instructional, and engineering design theories. As was stated earlier, a perfect model for designing CAI does not exist, but hopefully by continuing to improve on the various models of CAI design and incorporating more and more of the skills from recognized areas complementing the design of CAI, we will come closer and closer to perfection and effectiveness in the development of Computer Assisted Instruction.

Computer Assisted Instruction is not the answer to every instructional problem, and just as other methods may be appropriate for a particular instructional task, and audio/visual tools must be carefully matched with objectives so must the computer. Like film, television and all audiovisual aids computers can make instruction more efficient, effective and motivating, but only when used in an informed and instructionally sound way which takes advantage of its' positive characteristics. There are times when computers are ineffective and we need to learn to recognize this fact. The computer is simply another book on the shelf of the library of instructional methods. To be effective it must be chosen to answer the right questions.

References

Agneberg, Craig. Teaching "Filing Rules" via Computer Aided Instruction. A paper presented at the Microcomputer and High Technology in Vocational Education Conference. Madison, Wisconsin. August 1983. ED 238 402.

Bergland, G. & Gordan R. Software Design Strategies (2nd edit.) Murray Hill, New Jersey; Bell Laboratories,1981.

Bitter, Gary & Camuse, Ruth. Using a Microcomputer In the Classroom. Reston, Virginia: Reston Publishers Incorporated, 1983.

Burke, Robert L. CAI Sourcebook .Englewood Cliffs, New Jersey: Prentice-Hall, 1982.

Chu, Yaohan. Software Blueprint and Examples. Lexington, MA; Lexington Books, D.C. Heath and Company, 1982.

De Bloois, Michael L. Videodisc/microcomputer Courseware Design. Englewood Cliffs, New Jersey; Educational Technology Publication,1982.

Dixon, Robert C. & Clapp, Elizabeth. A Theory Based Computer Tutorial Model. CERL Report E-26. Urbana, Illinois: Illinois University, May, 1983. ED 234 755.

Florentin, JJ. Managing Software Projects . New York, New York: London and American ELSEVIER Incorporated, 1977.

Godfrey, D. & Sterling, S. The Elements of CAL. Reston, Virginia; Reston Publishing Company, Incorpoprated, 1982.

Hunter, Beverly. My Students Use Computers. Reston, Virginia; Reston Publishing Company, Incorporated, 1984.

Mann, William. Text Generation:The State of the Art and the Literature. Air Force Office of Scientific Research, Washington, D. C. December 1981.

Mc Cann, Patrick. Methods for Improving the User-Computer Inetrface:Technical Repport. San Diego, California: Navy Personnel Research and Development, August, 1983. ED 234 771.

Montague, William E. Analysis of Cognitive Processes in the Specification of interactive Nistructional Presenting For Computer Based Instruction. A paper presented at the annual meeting of the American Educational Research

Association. New, York, New York: March 21, 1982. ED 238 416

Norman, Donald A. Design Principles for Human-Computer Interface. A paper presented at the conference on Human Factors in Computer Systems. Boston, Massachussettes, December 1983. ED 238 416

Norman, Donald A. Five Papers on Human-Machine Interaction. Office of Naval Research, Personnel Training Branch. Washington, D. C. May, 1982. ED 222 195.

Osterweil, Leon J. Software Lifecycle Methodology and Tool Support. A paper presented at a workshop concerning software development tools at Pringrie Park, Colorado; May, 1979.

Perlman, Gary. The Design of an Interface to a Programming System and Menunix: A Menu Based Interface to UNIX. Office of Naval Research,Personnel and Training Research Programs Office, Arlinton, Virginia. 1982. Ed 212 289.

Peters, Lawrence. Software Design: Methods and Technologies. New york, New York: Yourdon Press, 1981.

Riddle, W. E. & Fairley, R. E. Software Development Tools . Springa-Verlog, New York, New York, 1980.

Siegal, Martin A. & Davis, Dennis M. The PCPSYSIV Management System: Educational Overview. Urbana, Illinois; Illinois University, May, 1983. ED 234 756.

Sincox, William. Configural Properties in Graphics Displays and Their Effects on Processing. A paper submitted to the National Institute of Education Contracts and Grants Management Division, Washington, D. C., 1983. ED 238 685.

Smith, Patricia & Boyce, Barabara Ann. Instructional Design Considerations in the Development of Computer Assisted Instruction. Educational Technology, July, 1984. Pgs. 5-10.

Willis, J., Johnson D. L. & Dixon, Paul. Computer Teaching and Learning. Beaverton, Oregon: Dilithium Press, 1983. Pgs. 257.