A Four-Stage Model for Planning Computer-Based Instruction

Gary R. Morrison
Department of Curriculum and Instruction
and
Steven M. Ross
Foundations of Education
Memphis State University
Memphis, TN 38152

Abstract. Computer-based instruction (CBI) lessons require thoughtful design and careful planning during their development. Existing CBI development models are not adaptable to lessons that vary in complexity or to differing roles and skill levels of a project staff. A flexible planning process in which the CBI design is implemented on paper as an intermediate step between the lesson design and the program production is described in this paper. The process consists of four major components: an initial flowchart, storyboards, a detailed flowchart, and an evaluation. Each step is described in detail using examples from recent applications. Emphasis is placed on the adaptability of the system for accommodating CBI projects that vary in scope and orientation.

It is generally agreed that an instructional design project requires a detailed plan before development can begin. Hannum (1986) stated, "It seems a truism that the quality of the final CBI lesson is more a function of lesson design than lesson authoring." He separates the development of a CBI lesson into two distinct parts. One is lesson design—the design of the instruction—and the other is lesson authoring—the development of the computer code. Lesson design includes those steps from the identification of the instructional problem through the design of the instructional strategies. At a minimum, the design would prescribe the sequences and form of interaction, the type of feedback, and the use of graphics and sound. Lesson authoring begins once the design is completed and involves writing the computer code to generate the lesson. Different approaches to the instructional design process are well documented in the literature (e.g., Jonassen, 1988). There is less documentation, however, on the process of translating into a CBI lesson the content defined during the instructional design phase.

There is a need for a well-defined but adaptable planning model for translating instructional design plans into CBI lessons. Such a planning model should meet four criteria. First, it should be adaptable—the model should operate as effectively for designing a simple drill-and-practice program as for a complex simulation. Second, it should provide an efficient means of "prompting" the designer for critical information about branching, data storage, cues, graphics, and sound (Richards & Salisbury, 1987), as well as for the components of the instructional design model. Third, it should provide a clear and accessible view of both the structure of the lesson and the amount of learner interaction. Fourth, it should provide a means for identifying various modules and subroutines to simplify the lesson authoring. A model that has been developed in accord with these four criteria and that has been used in several recent projects is described in this article.

Current Planning Approaches

The most common approach to CBI design involves using grid sheets (a 1:1 representation of the computer screen display) to specify content and display information (Richards & Salisbury, 1987). Each display of a lesson is represented as a separate storyboard in much the same way as scenes or slides are represented on video and slide storyboards. This approach, when applied to CBI displays, concentrates solely on the screen displays while tending to ignore the structure of a les-

There is a need for a well-defined but adaptable planning model for translating instruction design plans into CBI lessons.

son (Bork, 1985; Richards & Salisbury, 1987). One of the major criticisms Richards and Salisbury (1987) specifically note is the lack of cues on the screen design grid to prompt designers to maximize specific computer attributes such as graphics, interactivity, and animation.

Several recent planning models employ grid-based systems. In Allen and Erickson's (1986) interactive videodisc design model, the grid allows the designer to specify the exact placement of the content, prompts, graphics, and other visual or textual information. In Allessi and Trollip's (1985) model, the design of an instructional sequence is followed by the design of screens using a grid-based system. Dean and Whitlock (1983) use a similar grid design for screens, but they also develop a mainline chart that presents the content of the frames in the most direct path through the lesson, and a flowchart that indicates both the direct path and specific branches available to the learner. The Dean and Whitlock model provides for flexibility, but is redundant by repeating the content of the primary frames in both the mainline chart and the storyboards.

As an alternative to the grid sheet, Bork (1985) has proposed the development of a "script" that incorporates screen designs into individual flowchart blocks. The scripting process, however, can be cumbersome for the programmer to translate into screens since it is done in free form, i.e., it does not match the size constraints imposed by the CRT screen.

Another alternative to the grid procedure is Richards' and Salisbury's (1987) Screen Design Syntax (SDS). SDS involves using a code to identify such items as text, graphics, branches, audio, and correct and incorrect answers (see Figure 1). This syntactic code is similar to the pseudo-code used in programming. SDS's main advantage is the use of a word processor for formatting text, making similar or duplicate frames, and facilitating revisions.

SDS has four limitations. First, graphics cannot be incorporated directly into the screen design. Second, there is no coordinate system, as is available with grid sheets, to indicate the exact location of the text or graphic. Third, designers must master the special syntax used for designing the screens. Fourth, SDS fails to

```
A   The child in the example would be classified as

B          a.  Concrete

C          b.  Transitional

D          c.  Formal

E          Answer ==> ==> _____

----------NOTES----------

C1:  Enters B
       B1   Positive feedback
       S1   increment counter by 1
       S2   increment correct total by 1

W1   Enters A or C
       B2   Remediation
       S1   increment counter by 1
```

Figure 1. Example of screen display syntax after Richards and Salisbury (1987).

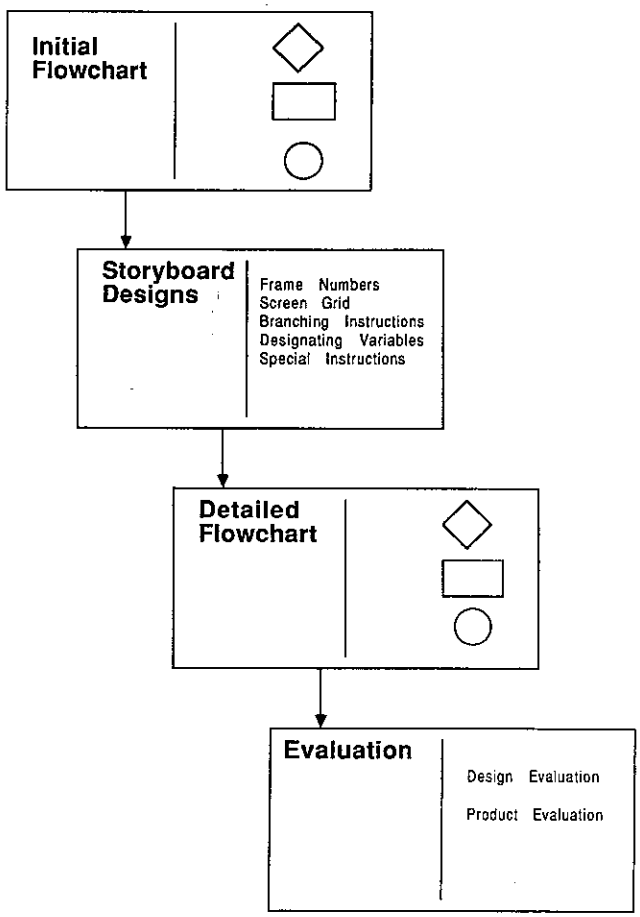

Figure 2. CBI design model.

adequately overcome one of the major limitations of grid designs—the absence of prompting for design considerations (e.g., graphics, animation, highlighting) as the screens are designed.

A Working Model for CBI Planning

Our planning process for translating an instructional design into CBI lessons includes four steps (see Figure 2). First is the development of a general flowchart identifying the major components of the lesson. Second is the development of the storyboards or screen designs with information on the variables and direction of program flow. Third is the development of a detailed flowchart from the storyboards. Fourth is the evaluation of the design plan and the resulting program. A description of each step follows.

Initial Flowchart

For the CBI designer, an initial flowchart is similar to the table of contents of a book. It specifies the major program components and their general organization. For example, the initial flowchart for a simple lesson might identify the title screens, a lesson menu, lesson segment 1, lesson segment 2, data storage, and instructional management components. In designing the initial flowchart, simplicity rather than detail is the key. Three symbols—a rectangle, a diamond, and a circle—are usually adequate for depicting program components. A rectangle is used to represent major sections of the lesson, such as the introduction, instructions, rules, examples, tests, etc. More elaborate print formatting or data storage instructions, if required, are provided on a separate flowchart constructed by the programmer. A diamond is used to indicate answer judging and decision points for program branching. A circle indicates continuation from one page or section to another. Brief descriptive labels are used to identify each component on the initial flowchart. To illustrate, Figure 3 shows a simple flowchart for a CBI unit consisting of three lessons and a posttest.

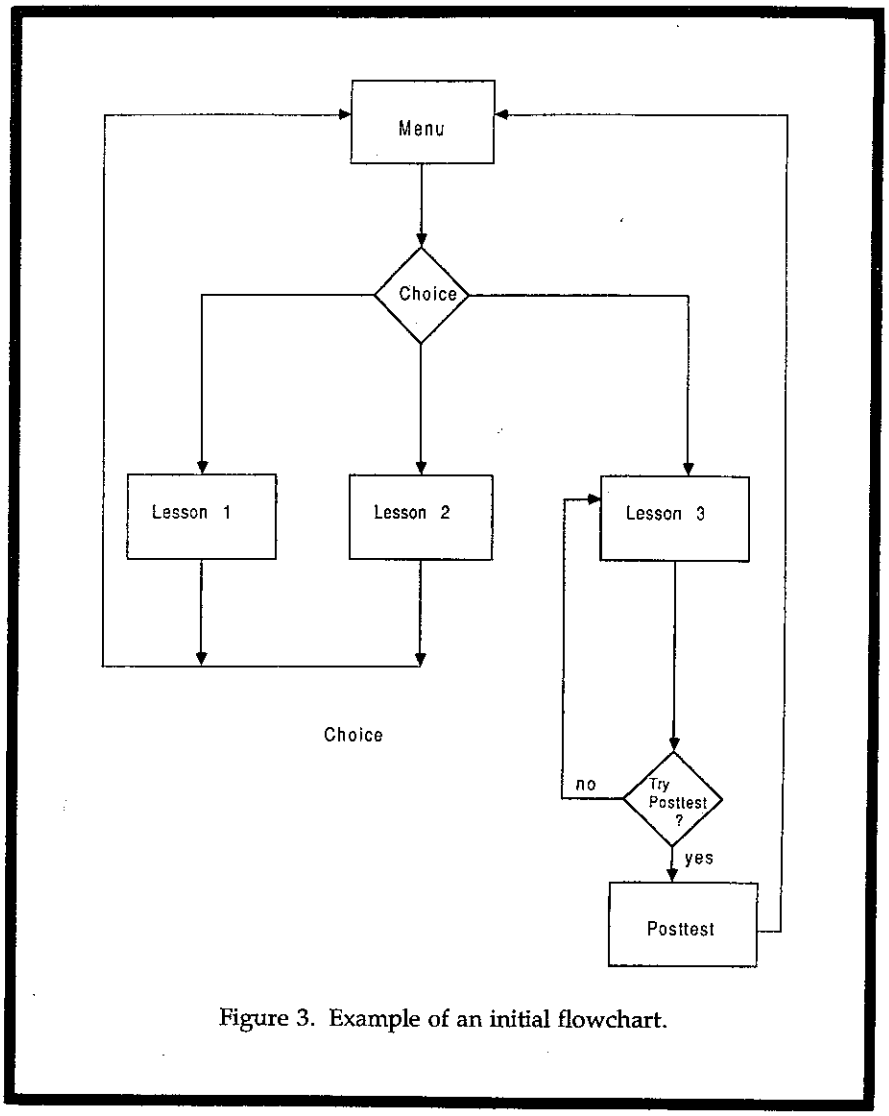

Figure 3. Example of an initial flowchart.

Initial flowcharts serve three purposes. First, they display the major segments or parts of a lesson. Second, they allow a designer to evaluate the flow from segment to segment and determine what transitions are needed between lessons. For example, at the end of a segment, should the program return to a menu or simply ask if the learner is ready to proceed to the next segment? (In the present example, the program returns to the menu following the first and second lessons, and branches to a posttest option following the third.) The third purpose is the identification of the major lesson components for use in scheduling, budgeting, and division of work if more than one programmer will be working on the program.

Once the major components and program flow are determined, screen design can begin, making use of the storyboards.

Storyboard

A storyboard is used with different media such as slides, film, and video to plan the individual scenes of a production. Similarly, a storyboard can be helpful for planning individual frames of a CBI lesson. The storyboard usually consists of a grid representing the CRT screen, frame identifiers, and space for notes. This 1:1 correspondence between the screen and storyboards enhances the efficiency of the storyboard for developing screen designs, translating the designs into code, and evaluating the CRT screens for accuracy. By incorporating a screen grid and related lesson specifications, the present model prompts designers for five types of information on each frame (see Figure 4). Depending on the specific lesson design, all or only some of these prompts may actually be used. A description of each prompt follows.

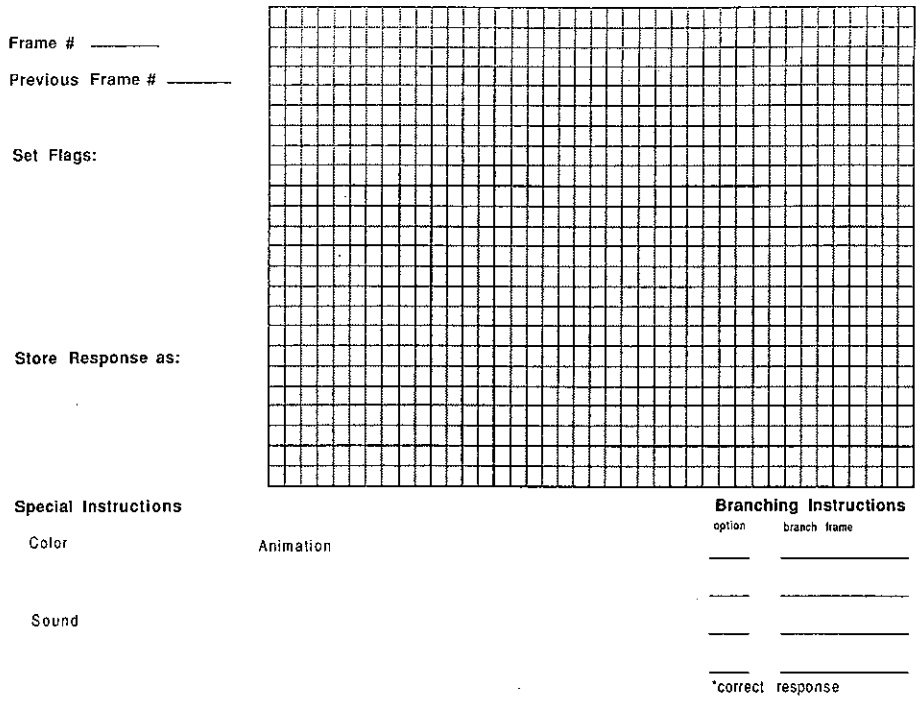

Figure 4. Screen design grid.

Frame Numbers/Identifiers. The first component on the storyboard identifies the number and type of the current frame and of previous contiguous frames. Frame numbers are used to track the flow of the program when making a detailed flowchart. An example of identifiers used in this model are illustrated in Figure 5. Similar identifiers can also be included in REM statements or as labels in the program code to help identify individual frames during debugging and revisions, or they may be printed to the screen to aid program debugging and to check the program's logic. The numbering syntax also serves to identify the following six general frame types.

1. *Information frames* are indicated simply by a frame number (1, 2, 3, and so on).
2. *Question frames* are indicated by the frame number followed by the letter Q (17Q, 38Q, etc).
3. *Remediation frames* for individual questions are indicated by the associated question frame number and the

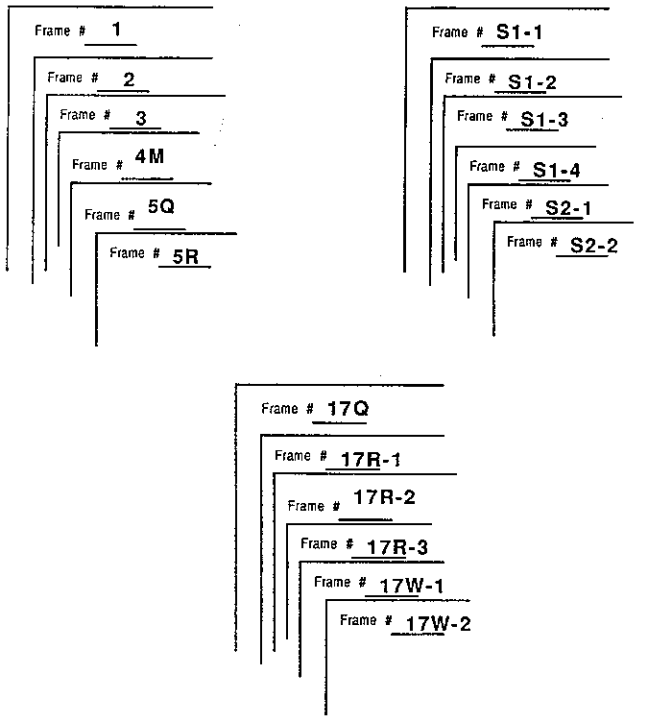

Figure 5. Example of frame numbering scheme.

letter response (38A, 38B, 38C for multiple-choice questions, 21F or 21T for true/false questions). For *branching* sequences which include more than one frame, the individual frames are noted with a hyphenated number (17A-1, 17A-2, etc.).

4. *Feedback frames* are indicated by the question frame number followed by an R for right answer feedback or W for wrong answer feedback (38R, 38W). A feedback frame, for example, might indicate only a sound (bell or tone) given with the question frame, a line or window to be added to the bottom of the question frame, or a new frame. In cases where several sequential feedback frames are used, a hyphenated number (38W-1, 38W-2, etc.) is added to the R or W notation.

5. *Menu frames* are indicated with the frame number followed by the letter M (1M, 83M). *Submenus* add a hyphenated number to the superordinate menu code (1M-1, 1M-2).

6. *Subroutines* are identified with an S and the subroutine number (S1, S2, S3). A hyphenated number is again used to identify sequential frames in the subroutine (S1-1, S1-2, S1-3).

These six general frame types were selected as representative of most frames used in computer-based instructional programs. Individual designers, however, might want to revise or create new frame types to match the components of their instructional strategies. For example, one could devise a notation for preinstructional strategies, example and nonexample concept frames, and coordinate concepts. Selecting frame identifiers that more closely reflect the components of the instructional strategy could also aid in the evaluation of the sequence. For example, a designer using the component display theory (Merrill, 1983) might adopt a snytax oriented to that model to reflect the prescribed primary and secondary presentation types.

Screen Grid. The second component of the storyboard is the screen grid which consists of 40 or 80 columns and 24 rows. This grid is used to design individual frames and templates for repetitive frames. For a simple drill-and-practice lesson, such as one involving identification of map outlines of the 50 states, one or two storyboard templates could specify the location of the map outline, the question stem, learner input area, and prompt area. All 50 drill frames could then be coded by the programmer from the template. Individual frames containing unique formats or content are designed on individual screen grids with text and graphics entered exactly as they are to appear on the screen. The resultant 1:1 correspondence between the grid and the CRT screen allows the programmer to easily understand the intentions of the designer.

Minor changes in subsequent frames or windows are indicated by drawing a box on the new frame and noting the old information (and reference frame) that stays on the screen. Information for the new frame is entered on the grid in the appropriate parts.

Branching Instructions. The third storyboard component is the branching notation that directs the lesson flow according to a fixed sequence or a variable sequence based upon learner responses. Branching locations are identified for each possible response to questions, menus, and other frames

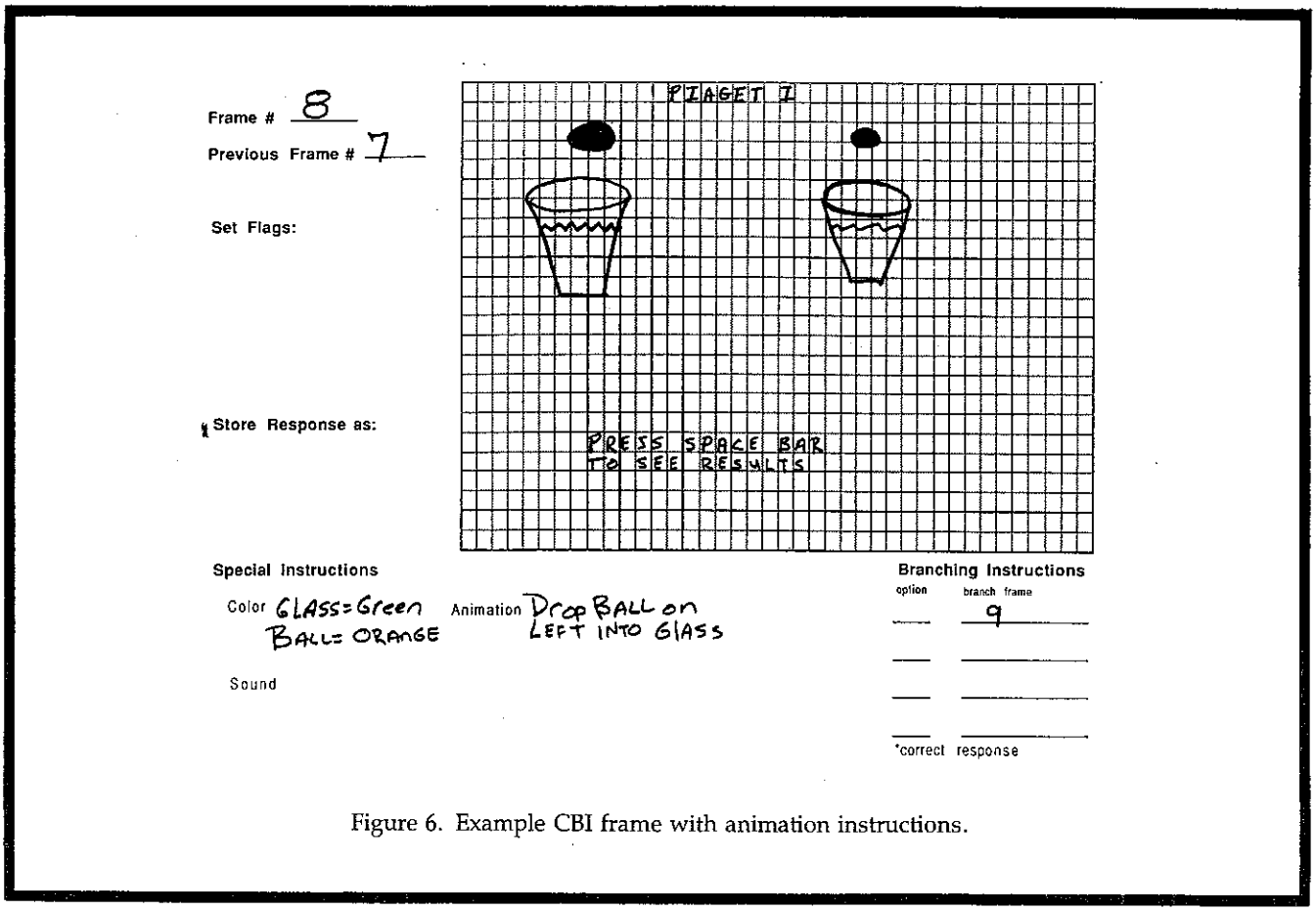

Figure 6. Example CBI frame with animation instructions.

requiring input, including simple keypresses (e.g., space bar, ESCAPE) used for forward and backward paging. Example notation would be of the type, "On A goto frame 26R," "On B goto M1," "On ESCAPE goto END."

Designating Variables. The fourth component is information on what learner response data are to be stored on a disk or to be used as variables within the program. Designers and programmers must identify these variables and the frames at which they are to be established. The model addresses this need by providing two types of prompts. The first prompt, labeled "STORE RESPONSE AS," is used to store either the student's actual answer or information about the answer (e.g., Right or Wrong). The second prompt, "SET FLAGS," is used to increment counter variables for tracking such data as the number of correct and incorrect responses, the number of times a frame is viewed, or the number of attempts required to answer correctly. Conditions under which credit is to be awarded (e.g., first try only) or when branching to a help frame is to occur (e.g., after the third wrong attempt) are also specified as flags.

Special Instructions. The fifth component of the storyboard is "special instructions" regarding uses of features such as color, animation, sound, or inverse or flashing text. This component is also used to specify information such as timing variables, notes on the placement of windows, replacement of text, and other special screen or coding features. Figure 6, for example, shows the special instructions specified for an animated sequence in a program that was developed to simulate a Piagetian interview (Ross, Barnette, & Morrison, 1987).

Detailed Flowchart

The next step in the process involves the design of a detailed flowchart from the information specified in the completed storyboards. The rectangle, diamond, and circle used in the initial flowchart are also adequate here, since the primary purpose is to identify program flow as opposed to the specific programming statements (e.g., READ, PRINT, etc.). In the detailed flowchart, a rectangle roughly corresponds to a single frame. The finished flowchart is used to evaluate the program design, develop code, and evaluate the completed software.

Development. Figure 7 presents parts of a detailed flowchart used to represent a drill-simulation program on "empathetic listening" (Ross, Barnette, & Morrison, 1987). Development of the flowchart starts by plotting the first frame as a rectangle with the frame number inside (notes or titles for reference purposes are often included with the frame numbers). Next, it is determined whether the lesson's logic allows branching to several frames or only to one frame. Where branching options exist, as in question or menu frames, a diamond symbol is used as the decision point and rectangles as the specific options. Again, the associated question frame number is written in each diamond. Circles may also be used to identify the beginning and end of each page of the flowchart and the sequences of pages. Any time a subroutine is used, a circle with the subroutine number is connected to the rectangle or diamond. The next section describes how the storyboards and detailed flowchart are used in the formative evaluation process.

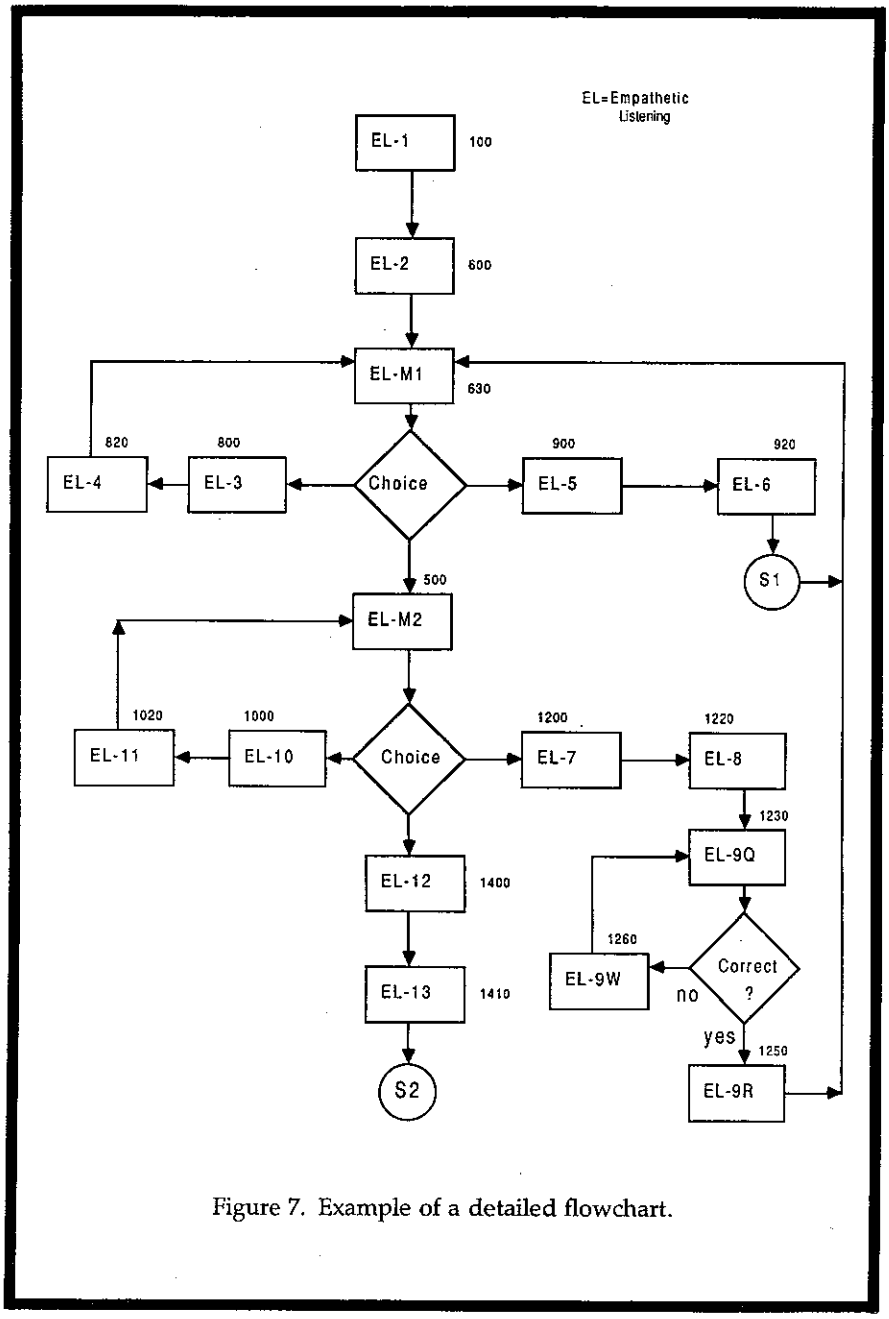

Figure 7. Example of a detailed flowchart.

Evaluations provide the designer with a means of evaluating the lesson design before time and money are invested in the coding process.

Evaluation of Program Design

The completed storyboards and detailed flowchart are used to conduct four formative evaluations of the instructional sequence and the resulting progam. Three of the evaluations are completed before the programming begins. The fourth is an evaluation of the CBI lesson.

Design Evaluation. Both the designers and subject-matter experts participate in the design evaluations. The first evaluation analyzes the sufficiency of branching instructions and verifies their conformity with the storyboards. Corrections in the sequence or the addition of new frames are made during this stage. For example, it may be necessary to insert a review option at the completion of an instructional segment, add an option to allow experienced users to bypass the instruction frames, or decide to introduce a prompt as a window. This initial analysis thus serves to identify weaknesses in the program design that can be corrected before the coding process begins.

The second evaluation identifies situations in which unanticipated responses could occur. If, for example, the user is prompted to enter the date in the form "MM/DD/YY," what will happen if the slashes are omitted, one digit is entered instead of two, or an unrealistic value (13 for month) is entered? This analysis may suggest the need for special help frames that give additional procedural instructions, or for special error trapping routines.

The purpose of the third evaluation is to determine the amount of interactivity in the lesson. Is the activity primarily "page turning," or does it involve answering questions, solving problems, or making content-oriented decisions? A primary focus of this evaluation is interaction that stimulates conscious cognitive processing, as described by Merrill (1983). Jonassen (1985) also provides a taxonomy that can serve as a basis for an instrument to assess the levels and types of interaction in the lesson. Infrequent branching suggests ineffective use of the computer's interactive and adaptive capabilities.

These three evaluations provide the designer with a means of evaluating the lesson design before time and money are invested in the coding process. Additional frames can easily be added at this stage without incurring additional costs for revising computer code.

Product Evaluation. The fourth evaluation uses the storyboards and detailed flowchart to assess the accuracy of the coding and screen designs. During the product evaluation, the designer determinates whether the branching instructions follow the specifications indicated on the storyboards and flowchart, and whether the CRT screen designs are effective with regard to such qualities as (a) liberal use of "white" space around graphics, blocks of text, and between lines, (b) clear visual separation of response prompts and lines of other text, (c) clear labels and prompts, (d) appropriate use of inverse or flashing words, (e) proper grammar and spelling, (f) split words, and (g) overwriting of graphics by the text. Color, inverse words or lines, and general layout may need to be adjusted for aesthetics or impact after the evaluation. Specific criteria for evaluating screen design can be developed from the works of Fleming and Levie (1978), Heines (1984), Hooper and Hannafin (1986), and Schneiderman (1987).

Storyboards and a detailed flowchart provide an accurate and efficient means for evaluating and revising the materials during the design phase and during the coding phase.

Applications and Utilization

This section describes different applications of the CBI planning model. Applications include using the storyboard grid designs to develop screen templates, to improve the human interface, and to evaluate the program design. Some strategies to facilitate programming are also described.

Storyboards and a detailed flowchart provide an accurate and efficient means for evaluating and revising the materials during the design phase and during the coding phase.

Screen Templates

The screen grid (Figure 4) provides a means for designing standard templates for lesson frames. Templates help ensure that the materials produced by different designers and programmers are consistent throughout the project. One of the first steps in designing a standard screen template is to determine the location of standard components—status line, prompt area, and learner response (Heines, 1984). These three components should be displayed in the same relative location and manner in each frame and across related lessons.

Figure 8 presents a sample screen based on a standard template. The status line, shown in the top row, typically identifies the location in the lesson (frame 12 of 23), the title of the module or segment (Piaget I), and type of frame (question, information, etc.). Icons such as a thermometer or clock can also be used to indicate progress or location in a lesson.

The second standard component is the prompt area (bottom of Figure 8) which communicates what action the learner should take, and the acceptable responses to enter. For example, the prompt line might direct the learner to press Q to quit, R to review the content, or A, B, or C to choose an answer.

The third component is the learner response area (middle of Figure 8). Heines (1984) recommends that a standard area be designated for user input, such as space below the multiple-choice alternatives. The designer might also choose an icon or phrase to indicate that area. For example, two dashes and an arrow may be used to prompt the learner for input to questions and menus. For fill-in-the-blank type questions, embedding a familiar prompt/cursor within the sentence itself, rather than at the bottom of the screen, avoids the situation in which the learner must glance back and forth from the sentence to the input area. Most importantly, the status line, prompt line, and input area should always appear in the same screen locations, and use the same symbols across frames to provide consistency both within and between modules.

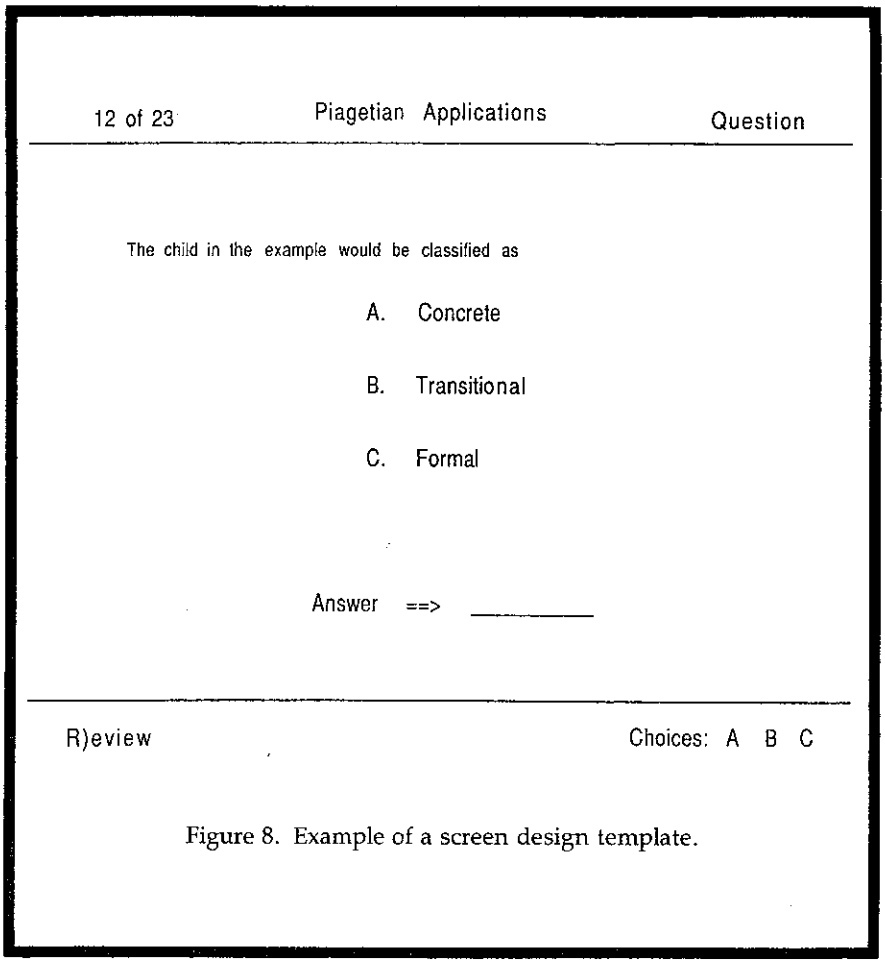

Figure 8. Example of a screen design template.

Interface considerations

An additional area to consider for standardization is the user interface. The designer may want to specify the parameters for menus, multiple-choice questions, "page turning," and exiting the program. Apple Computer (1986) and Schneiderman (1987) have each described various procedures for standardizing program menus. In a similar fashion, standard specifications can be made for identifying multiple-choice alternatives, such as using letters instead of numbers. The keys used to page through a lesson should also be clearly defined and systematized. For example, R may be used to page backwards (review), the space bar to move forward, ESCAPE or Q to exit, and M to return to the main menu. Maintaining consistency of symbols, responses, and format throughout a series of lessons or modules allows the learner to progress through the materials without having to learn a new interface with each new module.

Programming Considerations

The detailed flowchart developed by the lesson designer can also serve as a useful tool for the programmer. After completing the detailed flowchart, line numbers (in BASIC) or labels (in PASCAL or SuperPilot) are assigned to the various frames or segments in the flowchart (see Figure 7). This process leads to the development of a structured program which is easier to revise by grouping related frames into modules. Often a block of frames is assigned a starting number (e.g., 300) with increments large enough to allow adequate coding space between the various blocks. The flowchart has also been used to identify and plan subroutines that can be used for different programs. A library of procedures for creating boxes, borders, prompt lines, help screens, and response checking/editing routines has now been developed. These components are identified on the storyboards and on the detailed flowchart for incorporation into the new program. As different CBI projects are completed, guidelines such as those shown in Table 1 are documented. These quidelines help to ensure that the program can be easily revised if errors are found after the programmer has com-

Table 1

Programming guides

Print statements	Individual lines on the screen are to be written as single (separate) print statements. (This feature facilitates editing of text.)
Menus	Use numbers for menu.
Questions	Use letters (A, B, C) for multiple-choice questions.
Quit	Use ESCAPE to quit.
Repeat current lesson	Use RETURN for option to review current lesson.
Select menu	Use M to return to a menu when given choice of reviewing or exiting to menu.
Input	Check all input to determine if it is in range.
Data input	If the student is required to input a name or exit the unit, ask for verification with Y/N.
Main menu title	The main menu for the disk has the filename of "Menu" and requires learner to use this menu to quit the unit.

pleted the project, or modified for use in a different project.

Summary

This paper has presented a flexible planning process that provides a means of implementing an instructional design on paper as an intermediate step between the design and production of computer-based instruction programs. The model includes a simple syntax for numbering and identifying frames. This syntax can be modified for use with a specific instructional design model (e.g., Merrill, 1983) or for use with an individual designer's own heuristic-based system (Romiszowski, 1981). With this model, the designer has the option of designing screen templates for repetitive frames, or designing individual frames when the content or format changes.

The degree of specificity included in the detailed flowchart is also determined by the individual lesson design. Although the amount of detail included in each step will vary between projects, it is expected that the designer will complete each of the four steps in the model. The purpose of this planning process is to present the designer with a flexible tool that can be used in a variety of CBI projects without requiring unnecessary work.

References

Allen, B. S., & Erickson, D. M. (1986). Training interactive videodisc designers. *Journal of Instructional Development, 9*(2), 19-28.

Apple II Human Interface Guidelines. (1986). Cupertino, CA: Apple Computer, Inc.

Allessi, S., & Trollip, S. (1985). *Computer-based instruction: Methods and development.* Englewood Cliffs, NJ: Prentice-Hall, Inc.

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

Dean, C., & Whitlock, Q. (1983). *A handbook of computer based training.* New York: Nichols Publishing Company.

Fleming, M., & Levie, W. H. (1978). *Instructional message design: Principles from the behavioral sciences.* Englewood Cliffs, NJ: Educational Technology Publications.

Hannum, W. (1986). Techniques for creating computer-based instructional text: Programming languages, authoring languages, and authoring systems. *Educational Psychologist, 21*(4), 293-314.

Heines, J. M. (1984). *Screen design strategies for computer-assisted instruction.* Bedford, MA: Digital Press.

Hooper, S., & Hannafin, M. J. (1986). Variables affecting the legibility of computer generated text. *Journal of Instructional Development, 9*(4), 22-28.

Jonassen, D. (1985). Interactive lesson designs: A taxonomy. *Educational Technology, 25*(6), 7-17.

Jonassen, D. (Ed.) (1988). *Instructional designs for microcomputer courseware.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Merrill, M. D. (1983). Component display theory. In C. M. Reigeluth (Ed.), *Instructional-design theories and models* (pp. 279-335). Hillsdale, NJ: Lawrence Erlbaum Associates.

Richards, B. F., & Salisbury, D. F. (1987). The screen display syntax for CAI. *Performance and Instruction, 25*(10), 10-13.

Romiszowski, A. J. (1981). *Designing instructional systems: Decision making in course planning and curriculum design.* New York: Nichols Publishing.

Ross, S. M., Barnette, J. J., & Morrison, G. R. (1987). *Instructional disk package: Woolfolk's educational psychology.* (2nd ed.) Englewood Cliffs, NJ: Prentice-Hall.

Schneiderman, B. (1987). *Designing the user interface: Strategies for effective human-computer interaction.* Reading, MA: Addison-Wesley Publishing Company.