ED 088 424 IR 000 235

AUTHOR	Schneider, E. W.
TITLE	Course Modularization Applied: The Interface System and Its Implications For Sequence Control and Data Analysis.
INSTITUTION	Human Resources Research Organization, Alexandria, Va.
SPONS AGENCY	National Science Foundation, Washington, D.C.
REPORT NO	HumRRO-PP-10-73
PUB DATE	Nov 73
NOTE	21p.; Paper presented at the Association for the Development of Instructional Systems (Chicago, Illinois, April 1972)
EDRS PRICE	MF-$0.75 HC-$1.50
DESCRIPTORS	Behavioral Objectives; *Computer Assisted Instruction; Computer Programs; *Instructional Design; *Instructional Systems; *Management Systems; Program Descriptions; Systems Analysis; *Systems Approach
IDENTIFIERS	Assembly Language; COURSEWRITER; Instructional Modules; *Interface System

ABSTRACT

The Interface System is a comprehensive method for developing and managing computer-assisted instructional courses or computer-managed instructional courses composed of sets of instructional modules. Each module is defined by one or more behavioral objectives and by a list of prerequisite modules that must be completed successfully before the specific module can be attempted. The system's key components are: 1) a standard general structure for all modules; 2) a consistent method of labeling logic and text elements; and 3) computer programs (presently written in COURSEWRITER with Assembly Language functions) to regulate inter-module student traffic and to execute system-controlled and student-controlled instructional decisions. (PB)

IR

Professional Paper 10-73

HumRRO-PP-10-73

HumRRO

ED 088424

Course Modularization Applied: The Interface System and Its Implications For Sequence Control and Data Analysis

E. W. Schneider

Presented at the meeting of the
Association for the Development of
Instructional Systems (ADIS)
Chicago, Illinois April 1972

HUMAN RESOURCES RESEARCH ORGANIZATION
300 North Washington Street • Alexandria, Virginia 22314

November 1973

National Science Foundation
1800 G Street, N.W.
Washington, D.C. 20550

IR 000 833

Published
November 1973
by
HUMAN RESOURCES RESEARCH ORGANIZATION
300 North Washington Street
Alexandria, Virginia 22314

BIBLIOGRAPHIC DATA SHEET	1. Report No. HumRRO-PP-10-73	2.	3. Recipient's Accession No.

4. Title and Subtitle COURSE MODULARIZATION APPLIED: THE INTERFACE SYSTEM AND ITS IMPLICATIONS FOR SEQUENCE CONTROL AND DATA ANALYSIS	5. Report Date November 1973
	6.
7. Author(s) E.W. Schneider	8. Performing Organization Rept. No. PP 10-73
9. Performing Organization Name and Address Human Resources Research Organization (HumRRO) 300 North Washington Street Alexandria, Virginia 22314	10. Project/Task/Work Unit No.
	11. Contract/Grant No.
12. Sponsoring Organization Name and Address National Science Foundation 1800 G Street, N.W. Washington, D.C.	13. Type of Report & Period Covered Professional Paper
	14.

15. Supplementary Notes
Presented at the meeting of the Association for the Development of Instructional Systems (ADIS), April 1972

16. Abstracts

The Interface System is a comprehensive method for developing and managing CAI or CMI courses composed of sets of instructional modules. Each module is defined by one or more behavioral objectives, and by a list of prerequisite modules that must be completed successfully before the specific module can be attempted. The System's key components are (a) a standard general structure for all modules, (b) a consistent method of labeling logic and text elements, and (c) computer programs (presently written in Coursewriter with Assembly Language functions) to regulate inter-module student traffic, and execute system-controlled, and student-controlled instructional decisions.

17. Key Words and Document Analysis. 17a. Descriptors

Computer programs
Diagnostic routines
Instruction
Systems analysis

17b. Identifiers/Open-Ended Terms

Behavioral objectives
Computer-administered instruction
Course modularization
Instructional decision models

17c. COSATI Field/Group

18. Availability Statement Distribution of this paper is unlimited.	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 18
	20. Security Class (This Page) UNCLASSIFIED	22. Price

Prefatory Note

The work described in this paper was performed while Dr. Schnider was a research scientist with HumRRO Division No. 1 (System Operations), under Dr. Robert J. Seidel as Project Director. The work was performed primarily in connection with a Research Grant from The National Science Foundation, but it also drew upon earlier work performed for the Department of the Army under Work Unit IMPACT.

COURSE MODULARIZATION APPLIED: THE INTERFACE SYSTEM AND ITS IMPLICATIONS FOR SEQUENCE CONTROL AND DATA ANALYSIS

F.W. Schneider

ANALYSIS OF THE INSTRUCTIONAL SITUATION

Systems analysis of the instructional situation[1, 2] usually starts from a representation similar to the one shown in Figure 1. There are two black boxes, one labeled "Instructor," the other labeled "Student," and two lines, one representing the flow of communication from the instructor to the student, and the other representing the flow from the student back to the instructor. The instructor-student path carries information to be learned, questions to be answered, and so forth. The student-instructor link carries questions for the instructor to answer and answers to the instructor's questions.

The Instructional Situation in Simplest Terms

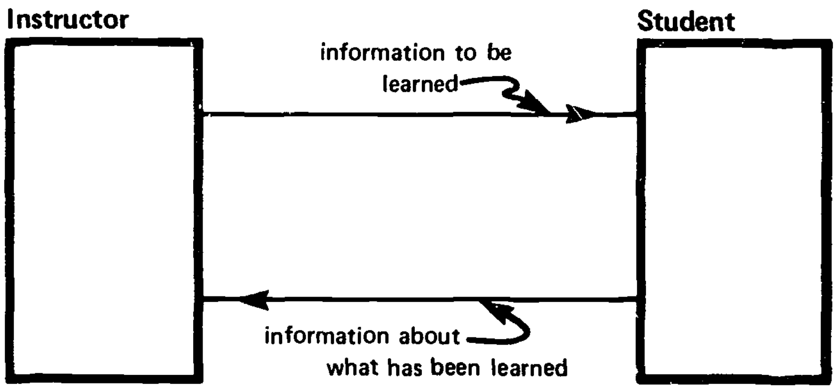

Instructor
Student
information to be learned
information about what has been learned

Figure 1

Now suppose that we sketch in some of the contents of the Instructor box. One essential element is represented by the amorphous blob in Figure 2. This represents the body of knowledge to be learned. The structure of this body of knowledge can be represented by a graph[3] where each node is a concept and each arc is an association. The

[1] Robert J. Seidel, Judy G. Compton, Felix F. Kopstein, Richard D. Rosenblatt, and Sally See. *Project Impact: Description of Learning and Prescription for Instruction*, HumRRO Professional Paper 22-69, June 1969.

[2] Felix F. Kopstein and Robert J. Seidel. *The Computer as Adaptive Instructional Decision Maker*, HumRRO Professional Paper 1-70, January, 1970.

[3] Edward Kingsley, Felix F. Kopstein, and Robert J. Seidel. *Graph Theory as a Metalanguage of Communicable Knowledge*, HumRRO Professional Paper 29-69, September 1969.

number of nodes is finite and the number of arcs is generally much larger than the number of nodes. The complete graph is essentially time-invariant; that is, the concepts and their relationships remain constant throughout the instructional interaction. Now, consider these properties of the body of knowledge in relation to the instructor's task, which is to transfer the informational content of the body of knowledge to the student. All the information in the body of knowledge exists in parallel, simultaneously, but the instructor-student link is not time-invariant. It transfers information in a serial fashion. This task of parallel to serial conversion is represented by the box in Figure 2 labeled Instructional Decision Maker. The term is deliberately vague as the content of this box is, in effect, the terra incognita of scientific pedagogics. At one time, the box was labeled simply Sequencer, but there is a falacious implication of automaticity in that term. The actual sequence depends upon what the instructor knows about the student (represented in Figure 2 by the Student Image box), and by the general rules for the instruction situation (represented by the Instructional Rules box). By instructional rules I mean, among other things, the protocol for the use of the communication link.

The Instructor Converts Parallel Information to Serial Information

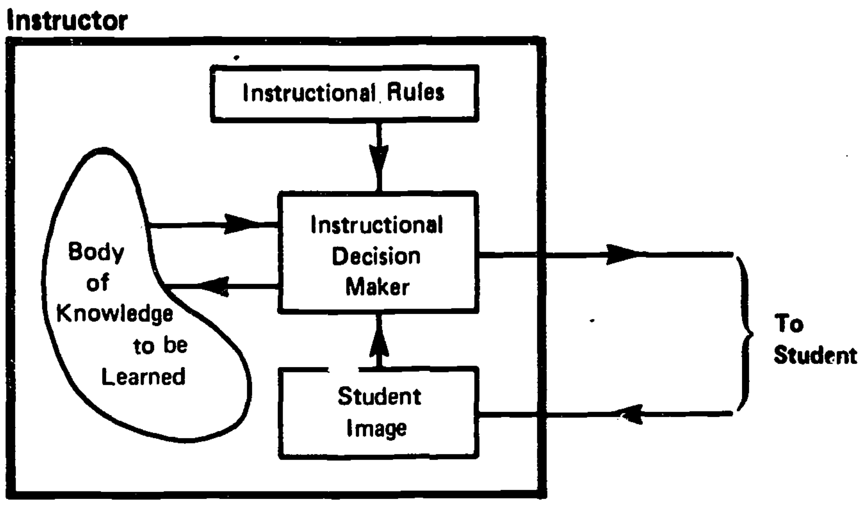


Figure 2

In a nutshell, the instructor-process consists of converting the parallel structure of the body of knowledge into a serial flow of information based upon the structure of the body of knowledge itself, what is known about the student, and the rules for the instructional situation.

Figure 3 diagrams the student process. Here the body of knowledge is not time-invariant. It is growing as the serial information from the instructor is stored in an associative, parallel form. The student uses the instructional rules and his knowledge of the already-constructed body of knowledge to decide whether a new piece of information can be added, and, if so, how it should be added. If the student is unable to encode a given piece of information, he can use the return path to the instructor to question him. Similarly, the instructor can ask questions of the student to confirm that the student can, in fact, correctly retrieve information from his newly constructed body of knowledge.

The Student Converts Serial Information to Parallel Information

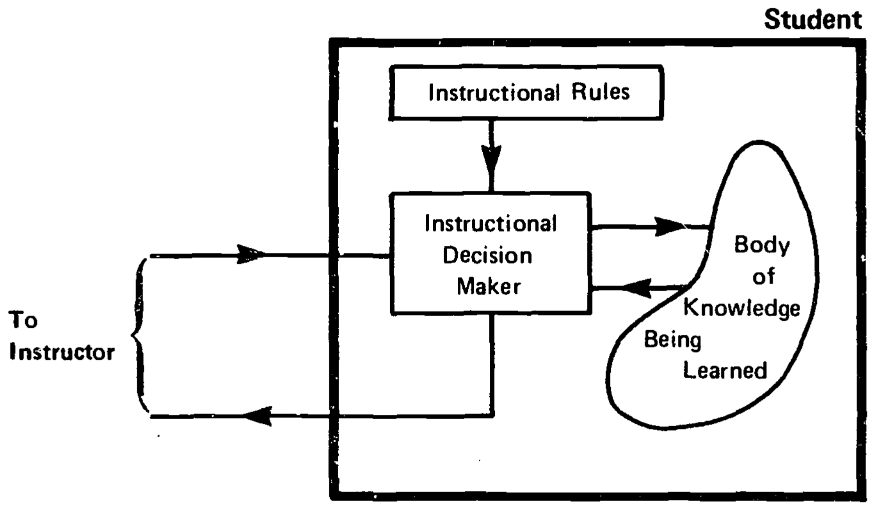


Figure 3

Surely, the reader can elaborate and refine the contents of both the instructor and the student processes, but the above models are sufficiently detailed to demonstrate (below) the effect of introducing a computer into this instructional situation.

INSERTING A COMPUTER INTO THE INSTRUCTIONAL SITUATION

Figure 4 shows our instructor conversing with a computer instead of a student. We have already discussed the instructor's task, but what is the computer's task? The computer has to extract from the instructor all the information that it will need to teach a large number of students with different abilities and interests. To do this, the computer needs to know a good deal more than what any one student would know. On the other hand, the computer isn't another instructor, a peer of our human instructor, because the computer knows nothing about the topic to be taught.

One alternative is to have the computer assume the collective role of a class of students, systematically constructing and presenting possible student responses to the human instructor, and then storing his rejoinders for use with future students. Another approach is to have the computer play two *distinct* roles—that of student and that of editor. One machine with two roles may confuse authors, but it should take less time to collect the needed information with this approach than with the first method.

In any case, the most immediate problem is not what role the computer should play, as opposed to the instructor, but what the computer should do with the information that it gleans from the instructor-computer interaction. Obviously, the information has to be stored until a student needs it. But how? At the present state of the art, in both instructional and computer technology, it is not feasible to store a veridical representation of the instructor's body of knowledge into the computer. There are too many relationships between concepts (too many arcs between nodes), and probably too many concepts to be stored in a reasonable amount of the computer's memory. But it is possible to store a greatly simplified version of the knowledge graph, if we develop behavioral objectives for each concept or set of concepts and replace the associative arcs

The Computer as an Instructional Interface:
Instructor Link

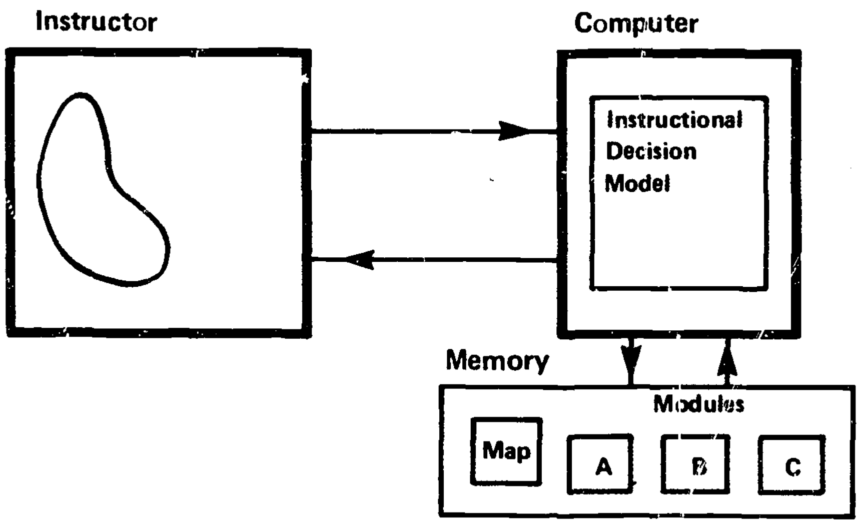


Figure 4

of the original graph with arcs representing only the precedence (or prerequisite) relationship. These arcs are determined when the question: "What objectives must the student meet before he can be expected to master this objective?" is answered.

These arcs can be stored in the computer as a set of bit masks and the nodes, the behavioral objectives, can be the teaching objectives of an instructional module and tested for mastery when the student completes the module. So, our computer memory in Figure 4 contains a map showing the prerequisite structure of a set of instructional modules and the modules themselves, each with a quiz to test mastery of its behavioral objectives.

WHAT DOES AN INSTRUCTIONAL MODULE LOOK LIKE?

We have talked about the use of the prerequisite relationship to produce a module structure. Now, let's look at the internal structure of a module (Figure 5). Each module consists of a set of instructional sections called T/P Sections, and a set of testing sections called Q Sections. The Telling subsection explains the basic concept or concepts, and the Practice subsection allows the student to apply the concepts to tasks similar to those in the Q Sections, receiving feedback and remediation, as necessary.

Why the alternate versions for the different sections? Alternate versions can serve a number of different purposes. Having alternate versions of the same material, which differ, for example, in reading level or reliance on graphical exposition, allows the system to better match the course material to each student's abilities. Some versions can be written for remedial use by students who fail the quiz on their first attempt. These versions can be keyed to the specific types of error made in the Q Sections. Another use for alternate versions is in writing the divergent experimental materials necessary for research in computer-administered instruction.

Similarly, alternate forms of Q Sections provide for alternate test forms and alternate scoring methods. When a student first enters a module, he is assigned one version of the T/P Section and one version of the Q Section. When he completes those, a new assignment is made and so on until he successfully completes the module.

Internal Structure of a Module

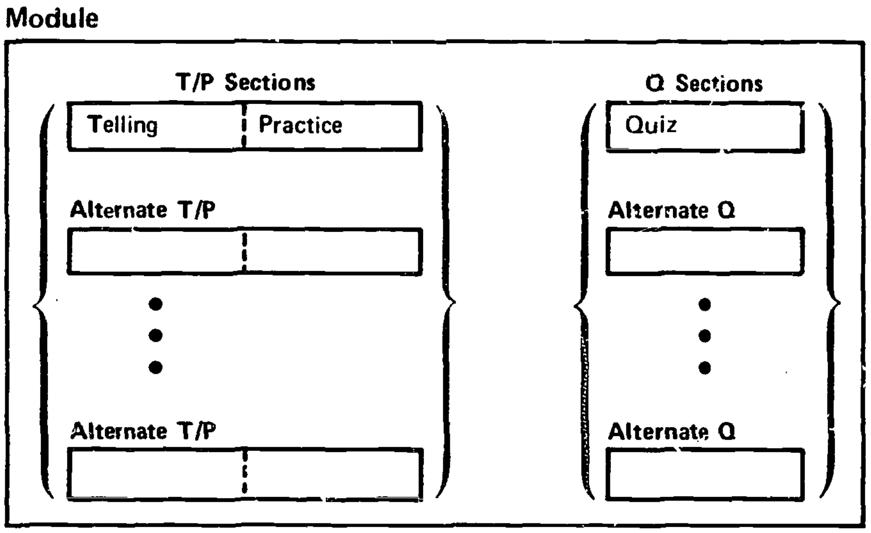
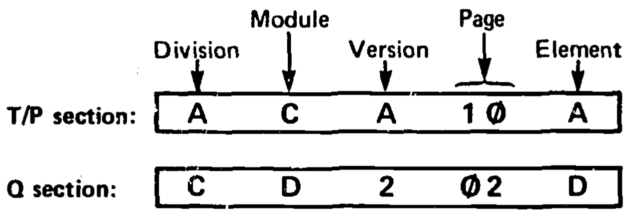


Figure 5

HOW TO LABEL INSTRUCTIONAL ELEMENTS

In order to keep track of these bits and pieces of instruction, there must be a consistent labeling scheme to identify and access them. Figure 6 shows the Interface labeling scheme. Each element of a course is assigned a 6-character Coursewriter[4] label structured in the following manner:

Coursewriter Labels as Access Keys to a Structured Course



I. Every T/P section starts with element A of page ØØ.
II. Every Q section starts with element A of page Ø2.
III. The sequence of labels conforms to the standard E B C D I C collating sequence.

Figure 6

[4] Coursewriter as referred to in this paper is a modification of the standard IBM computer-administered instruction program Coursewriter.

The first character is a letter representing the course division. A division is contained within a Coursewriter course segment and represents a major part of the overall course objectives. The next character position contains a letter referring to the module. Our present system can support 24 modules in each division. As mentioned before, a module teaches one or more closely related behavioral objectives. The next character position denotes the version within the module. T/P Section versions are denoted by alphabetic characters, Q Section versions are denoted by numerical characters. The fourth and fifth positions denote the page within the version and the sixth position denotes an element on the page.

There are a few other necessary codicils to the labeling protocol. For example, every T/P Section starts with element A of page 00. Similarly, every Q Section starts with element A of page 02. These two conventions allow the Interface System to synthesize the entry point to any division/module/version/section that may be required.

Another convention that greatly facilitates data analysis is that the sequence of labels within the course follows the standard EBCDIC collating sequence.

STRUCTURE OF THE INTERFACE SYSTEM ITSELF

Now that we've seen how the Interface can access the instructional elements, we can look at the relationship of the system to Coursewriter. Figure 7 shows the block diagram of a multi-segment Coursewriter course using the Interface System. The first segment, Segment 0, contains the Interface System and the next N segments contain the course divisions. To implement this course, the Interface System is copied into Segment 0, and then the course structure (i.e., the module map) is either copied into auxiliary storage blocks, one block in each segment being reserved for the purpose, or else the course administrator constructs the map by signing onto Segment 0 with a given student's number, and interacting with a map building routine.

Structure of the Interface System

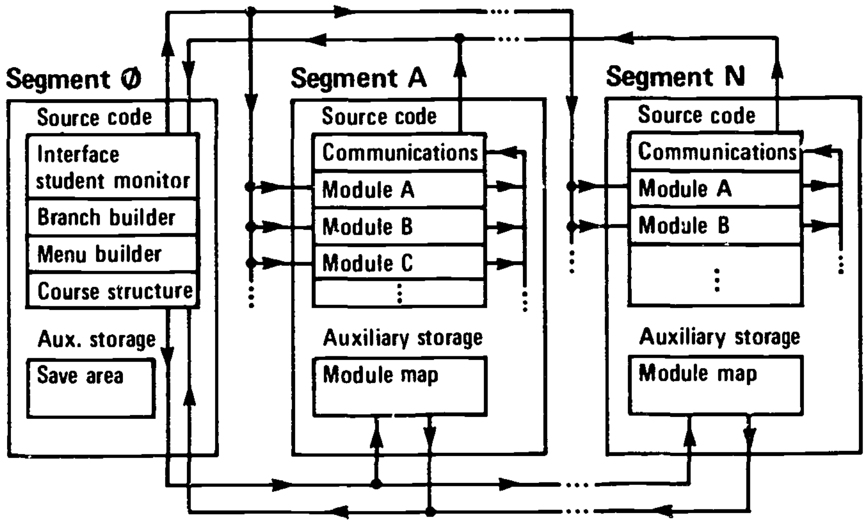


Figure 7

When an actual student signs on to the course, he starts out in Segment A. If the course has several "starting points," he is shown a menu and given a choice among them. If there is only one starting point, he automatically branches to the first display of that module. When the student completes his first module, the Coursewriter code branches him to a communications routine that, in turn, branches him back to Segment 0. The system then marks that module as completed in the module map. By comparing the prerequisite masks for each new module with the student's completion mask, Interface is then able to find all the modules that the student is now eligible to take, and it builds a menu showing the names of these modules and allows him to choose one. The System then builds an appropriate label and branches him to that module. When he has completed all the assigned modules in Segment A, he goes on to the next segment, and so forth, each segment communicating back to Segment 0 at the end of each module, or when the student invokes a student control option.

ENTER THE STUDENT

Let us now look at the corresponding computer-student link shown in Figure 8. The computer is now interacting with the student, but it still has an identity problem in that it knows more than the student, but less than the instructor. It knows less than the instructor, not only because of its inability to process "natural" language, but also because it has only the relatively crude representation of the body of knowledge that we discussed before. Fortunately, this problem has a better solution than the one caused by the instructor-computer link: the student can be encouraged to assume functions that are normally the prerogative of the instructor and to view the computer as part instructor, part access tool to the stored representation of the body of knowledge.

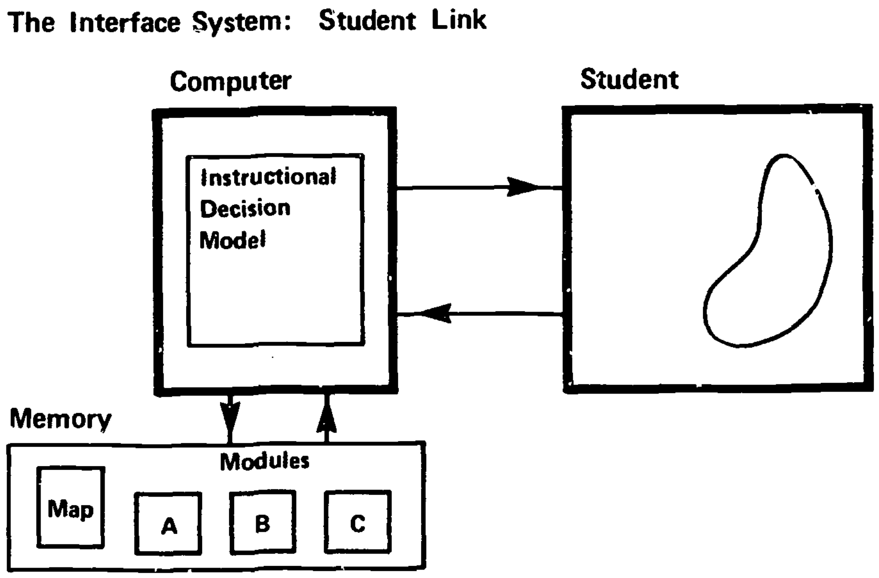

The Interface System: Student Link
Computer
Student
Instructional Decision Model
Memory
Modules
Map
A
B
C

Figure 8

Figure 9 shows the student control options and their relationship to the systems control. The central blocks in the figure depict the sections of a module. If the student is anywhere within the T/P Section, there is a 40-character block at the bottom of his CRT display. (Our CAI system uses CRT terminals.) The first five positions of this block contain the characters: INFO-. If the student types a word after the dash, followed by an EOB (EOB = end-of-block character), a definition is retrieved from the on-line glossary and displayed directly above this block.

Student Control Options

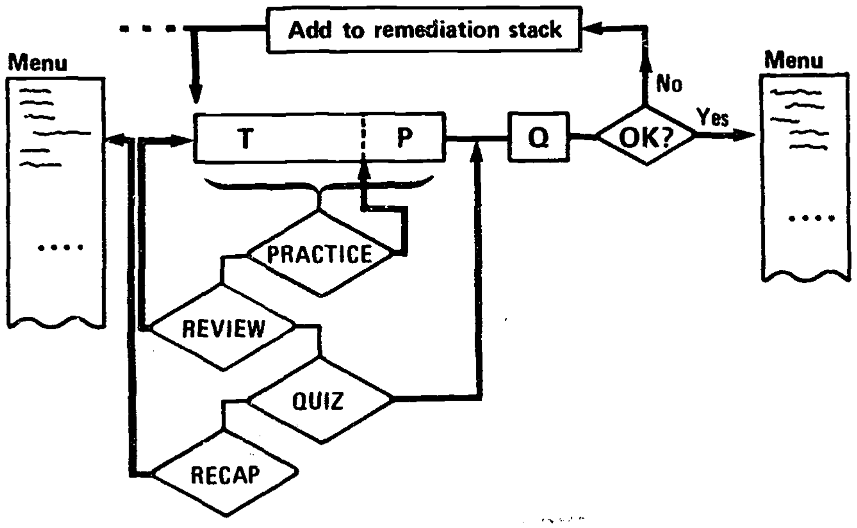


Figure 9

STUDENT CONTROL OPTIONS

Some words do not produce definitions. They have special meanings for sequence control. For example, if he types the word: PRACTICE, the system will position him at the beginning of the practice section. Similarly, if he types REVIEW, he will be positioned at the beginning of the module; if he types QUIZ, he will be positioned at the beginning of the quiz. If the student types RECAP, Interface constructs a menu listing all the modules that he has completed in that division, and positions him at the beginning of the T Section of the module he selects. If he has completed one or more divisions, the System first asks him which division he wishes to recap, and then builds a menu for just that division.

When a student is in a Q Section, all student-control options are turned off. Another system feature that can be especially useful with the recap option is the ability to mark certain modules as already completed when they are intiially registered as part of the structure. This provides an easy method of advanced placement. Students who already know some of the course material can have the corresponding modules marked as completed, and yet have them available for review by using the recap option. Using the same technique, division summary or overview modules can be written and registered as completed. The student is not required to go through these modules, but they are available when he uses the RECAP option.

If a student does not use the control options, but just keeps slogging along, the system will automatically take him from the end of the P Section to the beginning of the Q Section, and from the end of the Q Section to the Instructional Decision Model (IDM) in the Interface System. Generally, if the student's performance is satisfactory, Interface will build a menu of the modules that are presently available, or if only one module is available, it will just branch the student to that module automatically. If the student's performance on the quiz is not satisfactory, the system can stack any desired modules, including the one just completed, in the remediation stack. The remediation stack must be emptied before students can go on to new material. So, we see that the student has control over sequence within his present module and modules that he has already completed, while the system has control over access to new material.

IMPROVED DATA ANALYSIS

Another major advantage of the structured modular course is the potential for more compact, comprehensive, and systematic data analysis. In our system, we do not use the regular Coursewriter recording options. Instead, we have written a function, function RECORD, which writes the data indicated in Figure 10 each time that it is invoked. The 196 bytes of information tell us who, where, when, and what happened. One such recording is made for every student response processed by Coursewriter.

Our recordings have three special features: the recording time is recorded in tenths of seconds to provide unambiguous time sequences; a two-byte occasion code is added to the record to indicate the nature of the student's response (i.e., answer to question, glossary request, comment, etc.); and the record labels are of the structured type discussed earlier. We rely heavily upon the standard IBM SORT utility program, so it should be evident why the structured record labels must follow the EBCDIC collating sequence. Although they are a mixture of letters and numbers, the SORT utility puts them in the correct order as if they were entirely numeric.

Layout of Student Recordings

(Every student response produces a recording.)

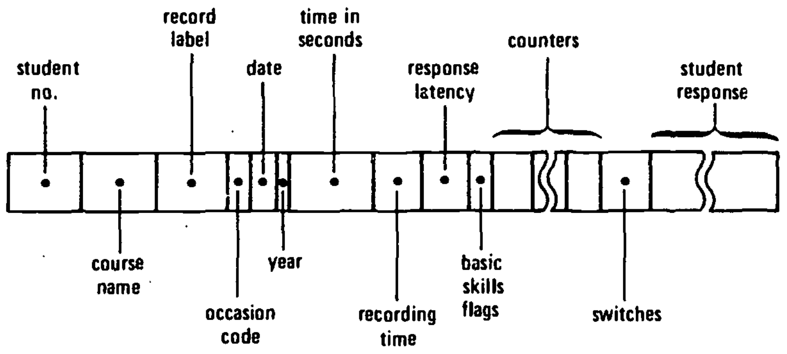


Figure 10

EXAMPLES OF DATA ANALYSIS OUTPUT

Here are a few examples of data analysis using these recordings. Figure 11 shows a summary student response matrix. Each column represents a student, each row represents a question that the student is expected to answer. A 0 in a cell indicates that the student attempted the question and answered it correctly. A 1 indicates that the student attempted the question and answered it incorrectly. A 9 means that the student sent a blank line as his answer. If the cell is empty, it means the question was not attempted. In our system, this is perfectly normal, because the students can skip to the quiz whenever they wish. A column with a preponderance of 1s or 9s indicates a student in trouble. A row with many 1s or 9s indicates the question in trouble. For example, the 4th column, student 131, contains five 1s and one 9. The lower margin shows that his percent correct for the T/P Section was 71% and his percentage for the Q Section was also 71%. The last row of the matrix, question AC124A, contains seven 1s, giving a percent correct of 36% as indicated in the right-hand margin. This type of summary matrix gives a quick reading of the module's performance, the student's performance, and their interaction.

Summary Student Response Matrix Group—DE05 Topic AC

		S128	S129	S130	S131	S132	S133	S134	S135	S136	S137	S138	S139	% Correct
	ACA11A	0	0	1	0	0	0	0	0		0	0	0	90.9
	ACA14A	0	0	0	0	0	0	0	0		0	0	0	100.0
	ACA16A	0	9	0	0	0	0	0	0		0	0	0	90.9
	ACA18A	0	0	0	0	0	0	0	0		0	9	0	90.9
	ACA20A	0	0	0	0	0	0	0	0		0	0	0	100.0
	ACA36A				1		9		1		0			25.0
	ACA38A				0		1		1		1			25.0
	ACA42A				0		0				0			100.0
	ACA44A				0		1				0			66.7
	ACA46A				0		0				0			100.0
	ACA50A				0		0				0			100.0
	ACA50B				1		0				0			66.7
	ACA50C				1		0				0			66.7
	ACA50D				1		0				0			66.7
	AC108A	0	0	0	0	0	0	0	0		0	0	0	100.0
	AC112A	0	0	0	0	0	0	0	0		0	0	0	100.0
	AC114A	0	0	0	0	0	0	0	0		0	0	0	100.0
	AC116A	0	0	0	0	0	0	0	0		0	0	0	100.0
	AC118A	0	0	0	0	0	0	0	0		0	0	0	100.0
	AC120A	0	0	0	9	0	0	0	0		0	0	0	90.9
	AC124A	0	1	1	1	1	1	0	1		0	1	0	36.4
T-SECT.	% Correct	100	80	80	71	100	79	100	71	0	93	80	100	
Q-SECT.	% Correct	100	86	86	71	86	86	100	86	–0	100	86	100	

Figure 11

Because our students can control their path through the course, it becomes more difficult and more important to keep track of where they went and when they went there. Figure 12 shows another computer-produced matrix, derived from the state-transition matrixes used to define stochastic processes. Each column represents the Nth label recorded for a student, and each row represents the (N+1) label. Each cell records the number of times that the students went from the corresponding column label to the corresponding row label. This particular matrix represents about half the T/P Section for one fairly large module. If the internal structure of this T/P Section were completely linear (i.e., every student saw everything that every other student saw in exactly the same order), then all the transitions would occur along the diagonal, one cell below the major diagonal of the matrix. Entries above this diagonal represent the branch-back to earlier material, and entries below the diagonal represent the branch-forward. Where students are branched to different materials as a consequence of their performance, the transitions appear in a column below this diagonal. For example, AEA16A is a question. Eighty attempts at this question were made. (Eighty people were branched to this label.) Of these 80 attempts, 39 resulted in a presentation of feedback AEA16B, 31 resulted in the message associated with the label AEA16G, and 10 attempts saw label AEA16H. Each of these feedbacks deals with a specific type of error. It is very simple in this case to see the relative prevalence of various types of error. When a student uses the REVIEW option, that means he will be branched to label AEA00A, so the corresponding row of the matrix indicates the use of this option and where it occurred. In like manner, the QUIZ option will produce a branch to label AE102A, and the corresponding row of the matrix indicates the use of this option. The data on this one page represent the combined paths of 43 students through this piece of instruction.

Summary Intra-Modular Path Matrix

	AEA00A	AEA01A	AEA04A	AEA10A	AEA12A	AEA14A	AEA16A	AEA16B	AEA16C	AEA16D	AEA16G
ADA00A	1	0	0	0	0	0	0	0	0	0	0
AEA00A	9	0	1	0	1	0	0	1	0	0	0
AEA01A	94	1	0	0	0	0	0	0	0	0	0
AEA04A	0	91	1	0	0	0	0	0	0	0	0
AEA10A	0	0	84	0	0	0	0	0	0	0	0
AEA12A	0	0	0	84	1	0	0	0	0	0	0
AEA14A	0	0	0	0	81	1	0	0	0	0	0
AEA16A	0	0	0	0	0	80	0	0	0	0	0
AEA16B	0	0	0	0	0	0	39	2	0	0	0
AEA16C	0	0	0	0	0	0	0	38	0	0	0
AEA16D	0	0	0	0	0	0	0	0	38	0	0
AEA16G	0	0	0	0	0	0	31	0	0	0	0
AEA16H	0	0	0	0	0	0	10	0	0	0	0
AEA18A	0	0	0	0	0	0	0	0	0	38	30
AEA18G	0	0	0	0	0	0	0	0	0	0	0
AEA18H	0	0	0	0	0	0	0	0	0	0	0
AEA19A	0	0	0	0	0	0	0	0	0	0	0
AEA19B	0	0	0	0	0	0	0	0	0	0	0
AEA20A	0	0	0	0	0	0	0	0	0	0	0
AEA22A	0	0	0	0	0	0	0	0	0	0	0
AEA24A	1	0	4	0	0	0	0	0	0	0	0
AEA26A	0	0	0	0	0	0	0	0	0	0	0
AEA26G	0	0	0	0	0	0	0	0	0	0	0
AEA26H	0	0	0	0	0	0	0	0	0	0	0
AEA28A	0	0	0	0	0	0	0	0	0	0	0
AEA28G	0	0	0	0	0	0	0	0	0	0	0
AEA28H	0	0	0	0	0	0	0	0	0	0	0
AEA30A	0	0	0	0	0	0	0	0	0	0	0
AEA30G	0	0	0	0	0	0	0	0	0	0	0
AEA30H	0	0	0	0	0	0	0	0	0	0	0
AEA32A	0	0	0	0	0	0	0	0	0	0	0
AE102A	12	3	2	0	2	1	0	0	0	0	1
BGA00A	0	0	0	0	0	0	0	0	0	0	0

Figure 12

The above examples were developed to answer research questions about the instruction. There are operational questions that are just as important, such as: "How many students showed up?" or "Was student 146 absent again?" or "Which one is the slowest student?" These kinds of questions are answered by our course management system (CMS). It produces daily status reports as well as module reports and class reports.

USING THE INTERFACE SYSTEM

Suppose that a course of instruction is being "systems-engineered" using the procedures described in CONARC Regulation 350-100-1, *Systems Engineering of Training.*[5] Further assume that the first three steps, through the training analysis, have been completed. Referring to Section IV, "Course Structure" of the training analysis, we can "package" each of the training objectives as an Interface instructional module. Each cluster of training objectives can be treated as an Interface division, which can contain up to 24 modules in the present implementation.

The training objectives can then be sequenced to produce the usual hierarachical structure. In a paper-and-pencil course, the hierarchy would have to be further reduced to a linear sequence. With the Interface System, the hierarchy can be preserved, allowing the instructor or the student the prerogative of selecting an individual path through the cluster.

For example, suppose a cluster in our course has the structure shown in Figure 13. If each module has one T Section and one Q Section version, the commands that the course registrar will use to build this structure in the computer are:

```
ADD  A/A1//
ADD  B/A1//
ADD  C/A1//
ADD  D/A1//
ADD  E/A1/A/
ADD  F/A1/ABC/
ADD  G/A1/DEF/
```

The first field after the verb ADD is the module designator. The next field specifies the order of T Section and Q Section versions. Because there is only one version of each for each module in the example, all the T Sections are designated version A and all Q sections are version 1. The third field lists prerequisite modules. Modules A through D have no prerequisites, so the field is blank.

A more elaborate example is shown in Appendix A of this paper. It shows the structure and the registration from HumRRO's COBOL2 CAI course.

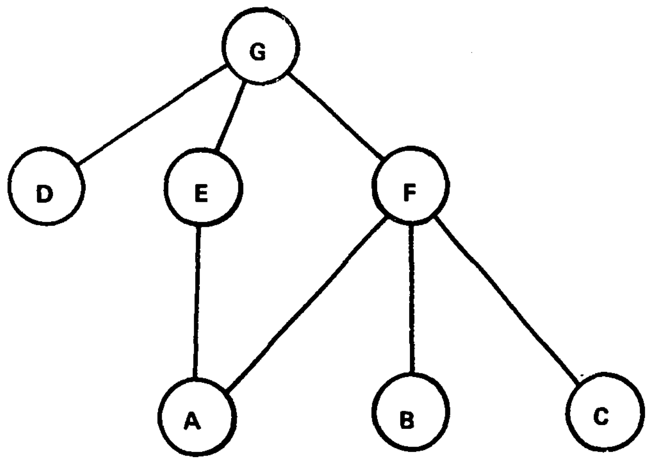
In the existing implementation, Interface supports only a linear sequence of clusters, because the initial application did not require anything more refined. A nominal amount of additional development could extend the method of hierarchically sequencing modules to hierarchical sequencing of divisions (clusters).

Section V of CONARC Regulation 350-100-1 discusses the identification of evaluation points in the course sequence. Interface provides for an evaluation point (Q Section) at the end of every module. If some of these quizzes are not needed, a trivial Q Section consisting of just a string of labels in the course logic can be used.

When the training analysis is complete, the next major step in the systems engineering of our course is the preparation for training (Appendix D, CONARC Regulation

[5] *Systems Engineering of Training (Course Design).* Headquarters, U.S. Continental Army Command, Fort Monroe, Va., Regulation 350-100-1, February 1968.

A Cluster of Training Objectives



Each objective is an Interface Module.

Figure 13

350-100-1). The availability of a computer adds a dimension to the choice of methods of instruction. Any given module could be any mixture of the following:

- A lecture or a demonstration, followed by a computer-administered and scored test. (Use a trivial T Section on-line.)
- A programmed-instruction booklet with an integral quiz, scored by the instructor. (In this case, the instructor inputs the student's score and the computer draws the appropriate conclusion.)
- A tutorial CAI lesson with an off-line GO—NO GO practical exercise as the quiz. (Here, the on-line Q Section is a loop with a code word, typed in by the instructor, that will branch the student forward. The code word is typed when the exercise is completed to the instructor's satisfication.)
- A training pamphlet, followed by an on-line quiz. If the student fails the quiz, he takes a CAI lesson to pick up what he missed the first time. Then he takes the quiz again.

A large number of similar permutations and variations exist. The basic point is that the Interface System does not require all the instructional material on-line to manage individualized instruction. When developing a new course, *all* the instruction might be off-line initially, essentially using the Interface System to support computer-*managed* instruction (CMI). As student needs and development resources dictate, on-line instruction can be developed and used module by module.

The fifth step of the system-engineering process is testing. The computer offers two significant advantages: immediately available scoring and feedback from the test, and sequential, diagnostic testing. The use of sequential test methods in CAI has been demonstrated by Ferguson[6]. The idea is to ask just enough questions to decide whether the student knows the material or not, and no more. The diagnostic testing commences

[6] R.L. Ferguson. *Computer Assistance for Individualizing Measurement*, Technical Report, Learning Research and Development Center, University of Pittsburgh, March 1971.

when a negative decision is reached, and continues until what he doesn't know and why he doesn't know it have been satisfactorily explicated. This type of testing may sound like a lot of trouble, but it is worthwhile to the bright student who isn't slowed down by unnecessary testing, and worthwhile to the unsuccessful student whose attention can be focused on specific weaknesses.

The sixth step in systems engineering, quality control, is a tailor-made computer application. The data matrixes discussed above demonstrate some of the possibilities in this area. Especially appealing is the possibility of a Course Management Information System (CMIS) that produces summary and exception reports upon completion of each instructional module.

In summary, the Interface System is a package of software routines and course-element labeling methods that provides computer support for the development and operation of modularized courses of instruction. The actual instructional material may be presented by the computer (on-line instruction—CAI) or may use traditional media (off-line instruction—CMI). In either case (or for any mixture), the Interface System provides:

(1) Individualized, self-paced instruction (where individual learning materials are provided).
(2) Records of student performance and speed, stored in the computer, and ready to analyze. These data can be used to evaluate student learning capabilities, instructional module teaching abilities, and the utility of various instructional strategies.
(3) Compatibility with *Systems Engineering of Training* regulations.
(4) Student control of path through the course, to the extent desired by the instructor.
(5) Flexibility in improving or expanding a course without inconveniencing the existing group of students.
(6) Generality in that the system makes no assumptions about the instructional content of the course.

Appendix A

STRUCTURE AND REGISTRATION NOTATION FOR HumRRO COBOL2 CAI COURSE

Cobol Course

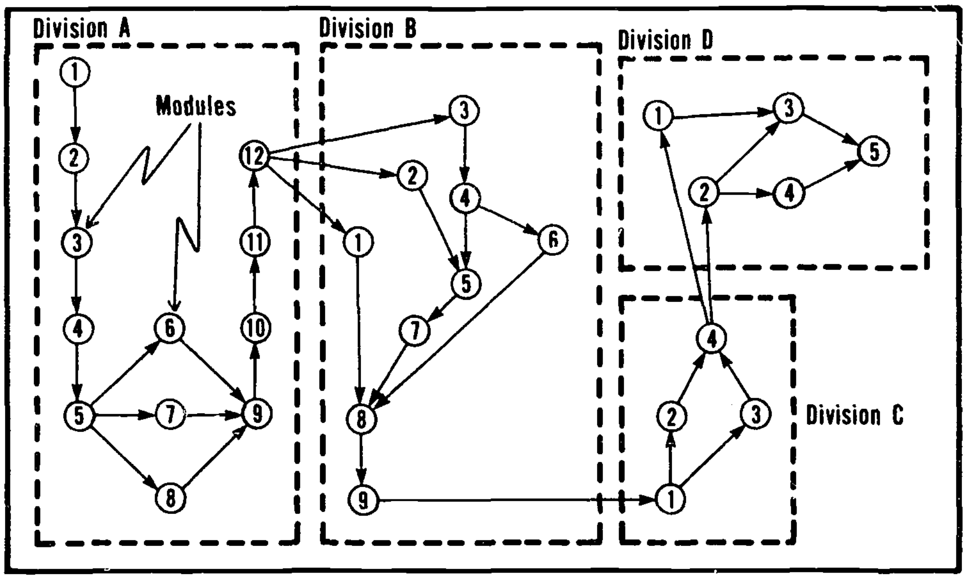


Figure A-1

```
DIVISION -I- STRUCTURE FOR STUDENT          999

        A/A1//
        B/A1/A/
        C/A1/B/
        D/A1/C/
        G/A1/F/
        H/A1/G/
        I/A1/H/
        J/A1/I/
        K/A1/J/
        L/A1/K/
        M/A1/L/
        F/A1/D/
        P/A1/M/

DIVISION -A- STRUCTURE FOR STUDENT          999

        B/B2B2XX/A/
        C/A1A1XX/B/
        D/A1A1XX/C/
        E/A1A1XX/D/
        H/A1A1XX/E/
        G/A1A1XX/E/
        F/A1A1XX/E/
        I/A1A1XX/FGH/
        J/A1A1XX/I/
        K/A1A1XX/J/
        L/A1A1XX/K/
        M/A1//
        A/A1B1/M/

DIVISION -B- STRUCTURE FOR STUDENT          999

        C/A1A1XX//
        B/A1A1XX//
        A/A1A1XX//
        D/A1A1XX/C/
        E/A1A1XX/BD/
        F/A1A1XX/D/
        G/A1A1XX/E/
        H/A1A1XX/AFG/
        I/A1A1XX/H/

ENTER REGISTRAR COMMAND—DIVISION B

DIVISION -D- STRUCTURE FOR STUDENT          999

        A/A1A1XX//
        C/A1A1XX/A/
        B/A1A1XX/A/
        D/A1A1XX/BC/
        E/A1A1XX/D/
        F/A1A1XX/E/
```

```
ENTER REGISTRAR COMMAND—DIVISION C

DIVISION -D- STRUCTURE FOR STUDENT          999

    B/A1A1XX//
    A/A1A1XX//
    C/A1A1XX/AB/
    D/A1A1XX/B/
    E/A1A1XX/CD/
    F/A1A1XX/E/
    C/A1A1XX/F/
```