DOCUMENT RESUME

ED 126 924 IR 003 839

AUTHOR	Seidel, Robert J.
TITLE	Research on Instructional Decision Models. Final Report.
INSTITUTION	Human Resources Research Organization, Alexandria, Va.
SPONS AGENCY	National Science Foundation, Washington, D.C. Office of Experimental Projects and Programs.
REPORT NO	HumRRO-FR-DI-73-6
PUB DATE	Dec 73
GRANT	NSF-GJ-774
NOTE	71p.
EDRS PRICE	MF-$0.83 HC-$3.50 Plus Postage.
DESCRIPTORS	Autoinstructional Aids; *Computer Assisted Instruction; Computer Oriented Programs; Computer Programs; Computers; *Computer Science Education; Educational Objectives; Educational Research; Experimental Programs; Higher Education; *Individualized Programs; *Instructional Systems; Learning Characteristics; Models; On Line Systems; Program Descriptions
IDENTIFIERS	Aptitude Treatment Interaction; COBOL; *Instructional Decision Models; Interactive Computer Systems; Learner Controlled Instruction

ABSTRACT

Optimization procedures for a computer-assisted instruction (CAI) system were developed using iterative development and tests of a series of instructional decision models (IDM). The result was a total systems effort in which the instruction was carried on by a dialogue between a computerized tutor and the student. A profile of the student, student characteristics at the time the course began, a map of the structure of the subject matter, and within-the-course history of information exchanges between the student and the CAI program were the factors examined. As factors and decision rules were increased, the IDM's were extended and retested to isolate the most appropriate combination of elements for the next version. The resulting IDM is adaptive to the different needs of individual students. (CH)

Final Report

Research on Instructional Decision Models

Prepared for

National Science Foundation
Office, Experimental Projects and Programs
Washington, D.C. 20550

Under Grant GJ 774

Robert J Seidel
Principal Investigator

December 1973

HumRRO Division No. 1 (System Operations)
HUMAN RESOURCES RESEARCH ORGANIZATION
Alexandria, Virginia 22314

FOREWORD

The research described in this final report was conducted by the Human Resources Research Organization. Sponsorship has been from Grant GJ774 from the National Science Foundation. Additional support for the activities was provided by the Department of the Army (Work Unit IMPACT and CATALIST).

The work was performed at HumRRO Division No. 1, Alexandria, Virginia. Dr. J. Daniel Lyons is Director of the Division.

In addition to Dr. Seidel, principal contributors to the research activities have been Mr. Richard Rosenblatt, Dr. Edward Schneider, Mrs. Judy Compton, Dr. John Stelzer, Mr. Michael Hillelsohn, and Mr. William Underhill. The report was prepared by Dr. Seidel, Mr. Rosenblatt, Mr. Hillelsohn, and Dr. Stelzer.

CONTENTS

Tables

Section 1

SUMMARY

The purpose of this National Science Foundation grant was to conduct programmatic research in the area of instructional decision models (IDM). This was accomplished over a three-year period in a computer-administered instructional (CAI) environment. The principal sponsor of HumRRO's CAI research and development efforts during that time was the Department of the Army (Army Research Office), under an overall program entitled Project IMPACT. The research conducted under this NSF grant augmented the IMPACT efforts.

The research strategy followed cyclical or iterative development and testing of successive versions of instructional decision models. This permitted the improvement of decision rules by which the subject matter, COBOL, was taught in an individualized CAI environment. Two iterations of the COBOL course were designed and developed. The latest IDM made possible a more individualized, adaptive, much improved course of instruction. It should be noted that the findings herein reported were gathered using an operational course vehicle with real-world application. The average time to complete the instruction was approximately 60 hours.

Also developed as a result of this research strategy were a number of software support capabilities that could be generalized to other individually adaptive instructional environments. A total systems approach was used in which the strategies, software, and content, as well as hardware capabilities, were revised from cycle to cycle during the course of the research.

Pre-course histories of individual students were incorporated as potential predictor variables, as were the students' in-course histories. Factors included in this exploration were Structure of Intellect tests developed by Guilford and his associates (said tests being relevant to cognitive kinds of tasks such as the COBOL course represented) and the use of Level of Aspiration or expectations of the students concerning their potential, as well as some exploration of the use of some confidence measures as additional in-depth assessments of the students' state of understanding.

Parallel to the efforts at improving the understanding of the students' attribute structure, with regard to entry characteristics and within-course changes in states of understanding, was the continued empirical and formal development of subject-matter structures. Techniques for modularizing course content were developed and implemented on-line within the HumRRO environment. General subject-matter mapping rules were also formalized during the conduct of this research, and this work continues beyond the tenure of the grant.

The IDM research results verified, in general, the value of Structure of Intellect test batteries in differential prediction of student performance with differing task requirements in the class of course content exemplified by COBOL (e.g., problem solving tasks of a high cognitive and memorial variety).

The value of student control options in a total tutorial CAI environment was also explored as part of this research effort. Essentially, the data showed that there are differences in the use of control options by students who tend to be high performers in contrast to those who tend to be low performers. The characterization of high performers was shown to be unique to the particular tasks or levels of the course and not a general

trait characteristic for a student across all portions of the course. Indeed, these characteristics were shown to be relevant and describable by the entry characteristics tests used during the research. Generally speaking, the high performers tended to use the student control options to a lesser degree and with greater efficiency than did the low performers.

A number of important course development tools were also accomplished as a result of this research effort. They provide major by-products to the National Science Foundation and can be generalized to other computer-instructional environments. Some of the techniques developed have already been adopted elsewhere (e.g., data management and text-logic separation techniques taken on by the TICCIT project at Brigham Young University). These techniques are described within this document and in the supplementary appendices. They include such things as preformatting capabilities for authors to use with minimal requirements for computer programming expertise, automated trouble reports that can be used on-line by proctors, and improved Coursewriter recording and analysis functions, as well as the general approach to modularization of course content.

Dissemination of information on the progress of this research effort has been accomplished through presentations at various professional and scientific meetings, as well as in publication form. These products are listed in the final section of the report. Additional dissemination has been accomplished in response to requests for the operational COBOL2 course, now available to any interested users. This use of the research results is discussed in Section 2, under Accomplishments.

The remainder of this report is separated into five sections relevant to the deliverable products promised as a result of the research activity. Section 2 is an introduction which deals with the objectives and accomplishments; next is a section on the details of the IDM research, followed by a section on the COBOL course implementation, along with the course development procedures and tools. Section 5 provides an overview of the software developments designed to support IDM research and operational implementation of courses of instruction, along with various formative evaluation techniques drawing upon the computer for efficient iterative development. Included here are data analysis techniques, Coursewriter function descriptions, and the various reporting capabilities developed at HumRRO. Lastly, the listing of and brief description of the publications accomplished during the conduct of this research effort is provided. The appendices include brief summaries of the more relevant supporting documents, which are available in separate form.

Section 2

INTRODUCTION

OBJECTIVES

The research proposal prepared by the Human Resources Research Organization addressed the problem of developing a comprehensive instructional model through an iterative process. Our approach distinguished between two avenues toward achieving this goal. *Empirical* tests were used to identify the variables of importance for prediction in the learning environment. In parallel, *theoretical* design was used to provide the formal framework within which the instructional model can be formulated.

The empirical work therefore was concerned with testing versions of the IDMs. Through extensive testing, measurement, and evaluation in a learning environment, two successive versions of the IDM were implemented, evaluated, and modified. This process led to the refinement of the IDM. At the same time, a theoretical framework in which the results of the IDM research could be coherently described was developed.

The general goal of the effort under the National Science Foundation grant was to provide a model of the human learning process. The model must be *descriptive* in that it must provide a precise description of the learning process. The model must also be *prescriptive* in that it must provide for specific solutions with respect to optimizing the learning process in the management of instruction. Conceptually, when the prescriptive portions of the learning model are implemented as rules or procedures for controlling an instructional process, then an Instructional Decision Model (IDM) results. The descriptive portion of the model yields constraints on the IDM. In this sense, the descriptive and prescriptive portions of the model merge in both the operating IDM and the course vehicle.

Since the initial course vehicle (COBOL1) and the initial IDM seemed unduly restrictive,[1] a rapid and massive effort at revision was made. The results (COBOL2) and the Interface schema[2] provided at the end of two years some developmental products originally promised for delivery to the Foundation at the end of the third year. Empirical research results, in turn, were provided at the end of the third year instead of the second year as originally planned.

The 1970-71 Technical Progress Report reviewed the approach taken and reported on the fact that less empirical research work had been done than originally planned during the first year and a half. The research was principally devoted to establishing systems reliability and implementing the course vehicle. While this action resulted in less data collection and experimental manipulation of pertinent variables, the available time permitted a substantial amount of (a) conceptual development relevant to a model for individualized instruction, and (b) empirical testing and refinement of course development products. Both of these efforts, described in separate documents (see Appendices A and B), have provided powerful tools for more efficient research on the descriptive/prescriptive problem of relating the learning model to the instructional decision process.

[1] See Technical Progress Report, 1970-71.

[2] See Proposal for Continuing Support, January 1972.

This work was performed during the third year of the research grant in order to answer the questions of how student and system control could best be implemented to facilitate management of instruction in an adaptive, tutorial CAI environment.

In summary, the proposed *objective* of this research was to:

(1) Arrive at an appropriate model of the instructional decision process for a given class of content.
(2) Implement the IDMs as computer programs.
(3) Ultimately determine optimization procedures for a meaningful CAI program.

ACCOMPLISHMENTS

Instructional Decision Model (IDM)

The implemented version of the first IDM has been described in the 1970-71 Technical Progress Report (pp. 6, 7, and 14-19). Initial results of the Iteration 1 testing of the model were also described in that document (pp. 17-18).

In general, firm conclusions concerning the adequacy of the initial IDM are not possible, for several reasons. First, the confidence measure as a diagnostic tool could not be evaluated because of faulty implementation. As has been indicated previously (Technical Progress Report, 1970-71), students tended, over time, to operate with the principle of least effort, using stereotyped confidence estimates (in the main, 0 or 100%). Secondly, because of the limited number of students used (a total of 42) and because the course proved to be very difficult (mean percent correct in final criterion, 59%), the data analyses could reveal only tentative suggestions for further work. Further analyses have been conducted since the last report was submitted. These analyses used better clustering techniques which eliminated bias in obtaining centroids in the previous analyses and provided additional step-wise multiple regression results.

Given the above limitations, the suggestions from the data obtained in the first two years were in essential agreement with the previous work of Guilford, Bunderson, and associates (Guilford, 1967; Bunderson, 1967; Dunham and Bunderson, 1968; Dunham, Guilford, and Hoepfner, 1968) concerning the applicability of Structure of Intellect concepts and of differential performance predictors useful in tasks of different levels of complexity. The research during the third year verified these findings and suggested that specific student profiles were associated with ability to self-manage instruction. Moreover, the student characteristics correlated with high performance were unique to specific divisions of the revised COBOL course, again supporting the concept of differential performance transfer based upon a match between instructional tasks and student descriptions. These findings and recommendations for future research are discussed in detail in Section 3 of this report, IDM Research.

The requirement for extensive course revision and formative evaluation came at a point in time which was simultaneous with a reduction in overall funding from the Army, a principal sponsor of the HumRRO CAI research. The result was a constrained implementation of new decision rules as they pertained to self-assessment (expectancy operators and the use of improved confidence measures). However, a redesign of these rules was completed and they are considered to be useful guidelines for instructional strategies wherever self-assessment and use of motivational indices are desirable. These decision rules are discussed for the use of Level of Aspiration (LOA) and confidence measures in Section 3 of this report.

The limitation on equipment and personnel resources permitted only the first experiment, on student versus system control, to be completed. The comparison of

figural versus semantic presentation of materials is currently being designed for implementation during the latter part of FY 1974 and the beginning of FY 1975.[3]

The formal theoretical model under development is at present primarily descriptive in nature and concerns essentially the mapping of subject-matter structure with the example provided so far, principally Boolean algebra. In addition to this, Appendix A presents a preliminary formulation of a model to represent student attribute structures. Prescriptive guidelines are currently being developed for the management of an individualized instructional environment, vis-a-vis unique instructional strategies matching student attribute structure with the structure of the subject matter. During the course of the formal theoretical work, the reduction in funding from the Army sponsor necessitated seeking additional support to accomplish this portion of the research. To this end funds were sought and obtained from the Air Force, and some empirical tests of subject-matter structuring were applied to a short Air Force course. The work governing the progress in formalizing the instructional process is detailed in Section 3, IDM Research.

In brief, the Iteration 1 of the Instructional Decision Model suggested that self-assessment could be a useful tool for prescribing management of the instructional process; however, more work needs to be done in this area. The Level of Aspiration (expectations on the part of the student) can be quite useful as a predictor for student performance. However, the use of confidence measures was not implemented adequately enough to permit final judgment on its utility to augment the use of expectations with respect to self-assessment. The confidence implementation was redesigned in some off-line initial testing with staff members, and guidelines have been developed for its inclusion in a subsequent instructional strategy investigation. It can be used, it is felt, as an aid to assessing progress in instruction and as a prescriptive index for remediation or acceleration. However, no firm judgments could be made from our existing research.

The second iteration of our IDM included the modularization of the course materials in accordance with specified subsets of behavioral objectives (see Appendix C as an added detailed discussion of this concept). Also, the Interface IDM framework was introduced and used quite effectively, along with the testing of various student or system controls over the options for remediation and acceleration. As indicated, these findings are discussed in more detail in Section 3, along with recommended use of our IDM findings.

Course Development

During the course of development of instructional decision models, we also developed an operational COBOL course which is described in Appendix B attached to this final report. The total number of students used to validate the COBOL course—that is, in terms of development, revision, and retesting leading to an operational product—was 205. COBOL1 involved 42 students, and COBOL2 included 83 students testing the revision of the instructional decision model, plus an additional 80 students who constituted the experimental sample for the research on student versus system control options.

Compared to COBOL1, COBOL2 reorganized the materials, that is, introduced some gaming and some restructuring, particularly the modularization already described. Secondly, the performance was much improved over COBOL1, especially with the last 80 students who served as the experimental subjects. Again, this will be discussed in more

[3]This will be accomplished on a new terminal system, Plato IV, sponsored by the U.S. Army Research Institute for Behavioral and Social Sciences (ARI). HumRRO lost its system capability during the latter part of FY 73. An additional hindrance has been the delay in delivery of required new hardware; thus experimentation has had to be postponed.

detail in Sections 3 and 4 (the latter concerning course development and implementation procedures). Suffice it to say here that in the revised COBOL2 course, for the introductory part of the course (Director), the first quartile score was 90% of objectives attained on first attempt, the median score was 80%, and the third quartile was 75%. For Division B, the second part of the course, the experimental students again scored an average of 80% correct in terms of number of objectives passed the first time (the median); first quartile in Division B was 85% of objectives passed the first time, and the third quartile was 75%. These data are in stark contrast to the initial criterion level of 59% for COBOL1. Thus, the course was much improved after revision.

In addition, the accomplishments under course development consisted of upgraded development and evaluation procedures with an automated trouble report capability (VOYEUR Program for proctors), modularization techniques in the form of SUPER-EDITOR preformatting capability, as well as the techniqtie for dividing the materials into subsets of behavioral course objectives. Next, the data management capabilities described in this report in Section 5 (particularly IDES-2) were upgraded to be much more efficient and relevant to the classic storage and retrieval capabilities of standard data processing systems. Finally, there were improved recording and analysis functions developed to interface with our CAI language, a modified version of Coursewriter III of the IBM processing system.

A copy of the operational course is designated as Appendix D. The complete course package includes a microfiche of text, student manual, course logic, and the introductory course.

Dissemination

With the completion of the operational COBOL2 course, a number of requests have been received and are in the process of being fulfilled for documentation and course listing in order that the course can be given elsewhere. To date, the Rochester Technical Institute for the Deaf has requested and been sold, at cost, a complete copy of the COBOL course. Other requests in process have come from the U.S. Census Bureau, Ohio State University, and Simon Fraser University. A complete package of all course listings and documentation has also been forwarded to U.S. Continental Army Command (now called TRADOC). Ballou High School of the District of Columbia has implemented a computer-managed (CMI) version of the course.

Additional dissemination of our materials has been accomplished through the request and delivery of software techniques and some listings to the Brigham Young University, Instructional Technology Group. (A letter of appreciation from Dr. C. Victor Bunderson is attached as Appendix E.)

All of the course products are available from Divison No. 1, HumRRO, at cost.

Section 3

IDM RESEARCH

INTRODUCTION

The second version of the Instructional Decision Model was implemented in the COBOL course concurrently with the large-scale revision, described elsewhere, of that material.

Principal among the features of the new IDM is the ability to afford the student a specified degree of control over the sequencing of instructional material. Of the four "learner-control" options used in the study, three—REVIEW, RECAP, and QUIZ—affect remedial activity and acceleration; the fourth, ROUTE, controls topic presentation sequence at specified points in the course.

Specifically, should a student type QUIZ he is branched from his current location in a topic to the beginning of the "Quiz" section of that topic (attempted use of this control option, or any of the other three for that matter, while in the Quiz section would be ignored by the IDM). If the student types REVIEW, he is returned to the beginning of the topic; if he types RECAP, he is shown a list of the topics he has already taken and allowed to review as many as he wishes without having to retake the quizzes. The student may exercise the first three options at his discretion. The fourth option, ROUTE, is enabled by the IDM at any point where the course prerequisite structure permits selection from equivalent topics. At such points the student is shown a "menu" of topics presently available, one of which he may choose to enter next; if he declines to choose, the system will randomly pick one of the topics for him.

These four learner control options comprise the independent variables in the present experiment.

During the conduct of the experiment, three types of measures were taken on each student: entry characteristics, including aptitude, affective, and biographical data; learner strategies, including type and frequency of control usage and the circumstances of their use; and achievement and other performance-related measures including quiz scores, transit times, programming errors, opinions of topics, and Level of Aspiration prior to the quiz of each topic.

It was the purpose, then, of the present experiment to assess the relative contributions and interactions among the learner-control and entry characteristic variables with respect to instructional effectiveness and efficiency as represented by the dependent measures.

EXPERIMENTAL DESIGN

Prior to the start of the course, each student was randomly assigned to one of the 16 possible combinations of the learner-control options; thus some students had all options available, some had access to none, some had only one of the four, while others had some combination of two or three control options.

Every student saw the same topics, although not always in the same order since those who had ROUTE were able to modify the sequence to some extent. Further, the

particular option combination to which students were assigned and the extent to which they exercised those options would cause the exposure to a given topic to vary among students.

Students who failed the quiz for a given topic were branched back to the beginning of that topic to restudy the material and retake those quiz items previously missed; the amount of time spent in restudy depended on the student's own discretion and on his option combination (e.g., a student with the QUIZ option could jump to it immediately, whereas a student without QUIZ would perforce see all instruction in the topic before his second attempt at the quiz items). Should the student fail the quiz a second time, the same procedure would be followed except that a staff member would monitor his third attempt at the quiz and clear up any misconceptions evidenced by the answers.

At the end of each division of the course, the students were required to code, run, and debug a COBOL program to demonstrate mastery of the skills taught in the topics of that division; successful completion of this task was a prerequisite to starting the topics of the next division.

The revised COBOL course originally contained four such divisions with a total of 32 topics comprising about 60 hours of instruction. A reduction in resources coupled with the impending loss of HumRRO's in-house computing capability necessitated shortening the course in order to guarantee a sufficient number of subjects for the experiment; consequently, for the present study the first two divisions, comprising 21 modules and about 30 hours of instruction, were used.

SUBJECTS

Ninety percent of the sample (N=80) were paid volunteers recruited through advertisements in the local newspapers; the remaining subjects were military personnel, also volunteers, supplied by the Army's Project Transition—an activity designed to assist separating soldiers in developing job-related skills for civilian life.

Our experience in the first COBOL course demonstrated that students who were severely deficient in programming aptitude were generally incapable of acquiring even the basic skills taught in the course. Since such students provided little usable data while placing an additional strain on our already limited resources, we screened out any prospective student whose score on the IBM Programmer Aptitude Test fell below a raw score of 46 (a "low C") by more than one standard deviation.

To make our experimental findings relevant to real world training, we intended that our subjects reflect the characteristics of programming trainees generally—young, with a minimum of some high school, and naive with respect to programming. The data indicate that these requirements were met. Table 1 summarizes the relevant biographical characteristics of the sample.

Eighty-two percent of the sample were 30 years old or younger; 52% were 25 or younger and approximately 27% were under 21. Ninety-six percent had no programming experience or training.

Nearly 99% of the sample had at least some high school education. Seventy-six percent had at least a high school education, and 55% had at least some college, while 30% had a college degree or beyond.

While we consider the age spread of the sample to be appropriate, they were, if anything, somewhat overeducated for our purposes; the effect of their schooling on course performance will be shown in the results section.

While over half the sample had vision difficulties (generally a need for glasses), in response to an exit questionnaire almost no one reported any problems reading the course materials from the display devices.

Table 1

Personal Data for Experimental Subjects

Characteristic	Percent of Responses in Category	Percent of Total Group
Sex		
Male	45	56.3
Female	35	43.7
Age		
Less than 21 years	22	27.5
21 - 25 years	28	35.0
26 - 30 years	16	20.0
31 - 35 years	3	3.8
36 - 40 years	4	5.0
More than 40 years	7	8.8
Vision Difficulties		
Yes	46	57.5
No	34	42.5
Typing Speed		
Non-typist	24	30.0
1 - 20 WPM	7	8.8
21 - 40 WPM	32	40.0
41 - 60 WPM	10	12.5
61 - 80 WPM	1	1.2
Greater than 80 WPM	2	2.5
No response	4	5.0
Educational Level		
Eighth grade (8 years)	1	1.2
Some high school (9-11 years)	14	17.5
Completed high school (12 years)	19	23.8
Some college (13-15 years)	21	26.2
Completed college (16 years)	14	17.5
Graduate (more than 16 years)	10	12.5
Other	1	1.2
Training and Experience		
No training or experience	77	96.2
Some training and/or experience	3	3.8
Current Occupational Status		
Student	27	33.8
Military	8	10.0
Employed—other	20	25.0
Unemployed	25	31.2

DESCRIPTION OF ENTRY CHARACTERISTICS MEASURES

Student entry characteristics were measured in the following four areas: Structure of Intellect Factors, Motivation, Computer Programming Aptitude, and Reading Comprehension.

The Entry Characteristics Test (ECT) battery consists of 27 instruments which yield 35 distinct scores; all but one are time tests. They range in length of testing time from 2 minutes to 70 minutes, with a majority of the tests involving less than 20 minutes testing time. In general, the tests are of the paper-and-pencil variety and designed for administration in group testing sessions.

Structure of Intellect Factor Tests

These tests measure ten factors and were tests used most recently by Bunderson (1967). There are 27 tests used for factor measurement. The factors and tests used to measure these factors are described below.

(1) General Reasoning (2 tests). This factor has been described as "the ability to solve a broad range of reasoning problems, including those of a mathematical nature" (French, Ekstrom, and Price, 1963). These tests selected to define this factor are the Ship Destination Test (Sheridan Psychological Services, Inc.) and Necessary Arithmetic Operations (Educational Testing Service).

(2) Induction (3 tests). This factor has been described as "associated abilities involved in the finding of general concepts that will fit sets of data, the forming and trying out of hypotheses" (French *et al.*, 1963). The Letter Sets Test (Educational Testing Service), Locations Test (Educational Testing Service), and Figure Classification Test (Educational Testing Service) define this factor.

(3) Figural Adaptive Flexibility (5 tests). French *et al.* describe this factor as "the ability to change set in order to meet new requirements imposed by figural problems." The following five tests define this factor:

(a) Match Problems IV (Part 1 and Part 2) (Aptitudes Research Project, University of Southern California).

(b) Match Problems V (Educational Testing Service).

(c) Word Coding Test (designed by Lennart Sjoberg, John Frederiksen, and Victor Bunderson).

(d) Decoding Test (designed by Lennart Sjoberg, John Frederiksen, and Victor Bunderson).

(4) Verbal Reasoning (3 tests). This factor has been given a number of different names, including "Deduction" by Thurstone, "Logical Reasoning" by Guilford, and "Syllogistic Reasoning" by French *et al.* (1963). French *et al.* describe it as "ability to reason from stated premises to their necessary conclusions." The Nonsense Syllogisms Test (Educational Testing Service), Logical Reasoning Test (Sheridan Psychological Services, Inc.), and Inference Test (Educational Testing Service) were selected to define this factor.

(5) Symbol Substitution (1 test). Guilford and Hoepfner (1966) classify this factor as a factor of convergent production and define it as "the ability to produce a completely determined, symbolic deduction from given symbolic information, where such an implication has not been practiced, as such." One test defines this factor: Sign Changes (Aptitudes Research Project, University of Southern California).

(6) Chunking Memory (1 test). This is a new factor postulated by Bunderson. He designed two tests; in the present study one of them, the Binary Digit Span Test, measures this factor. He also developed a control test, the Number Span Test-VS, which is "similar in content but not in opportunity for chunking" (Bunderson, 1967).

(7) Memory Span (2 tests). This factor has been described as "the ability to recall perfectly for immediate reproduction a series of items after only one presentation of the series" (French *et al.*, 1963). The marker tests for this factor are the Auditory Number Span Test (Educational Testing Service) and the Auditory Letter Span Test (Educational Testing Service).

(8) Associative Memory (3 tests). This factor is defined as the ability to remember paired associates. The three tests which define this factor are: the Picture-Number Test (Educational Testing Service), the Object-Number Test (Educational Testing Service), and the First and Last Names Test (Educational Testing Service).

(9) Perceptual Speed (1 test). This factor is described as the ability to make comparisons and find figures fast and accurately. The test which defines this factor is the Number Comparison Test (Educational Testing Service).

(10) Spatial Scanning (1 test). This factor is defined as "speed in visually exploring a wide or complicated spatial field" (French *et al.*, 1963). The Maze Tracing Speed Test (Educational Testing Service) measures it.

Motivation

Included in the Entry Characteristics Test battery are four tests which were selected to measure anxiety and achievement motivation. The psychological literature is replete with studies showing relationships between anxiety and learning in laboratory situations (Spence and Taylor Spence, 1967). Recently studies by Hansen and associates (1969) have also shown some value in the study of anxiety as it relates to performance in CAI. These three tests are the IPAT Anxiety Scale Questionnaire (Institute for Personality and Ability Testing), the Sarason Task Anxiety Questionnaire (adapted from Mandler and Sarason, 1952), and the Sentence Completion Test of Achievement Values (Mukherjee, 1964).

Computer Programming Aptitude

A survey of the literature revealed that by far the most widely used test of aptitude for programming has been IBM's Programmer Aptitude Test (PAT) and Revised Programmer Aptitude Test (RPAT). A large body of reliability and validity data is associated with these tests. Recently, the PAT and RPAT have been replaced by the Aptitude Test for Programmer Personnel (ATPP), which is included in the battery. This test correlates highly with both PAT and RPAT.

A second test, Primary Mental Abilities (PMA), will also be used to measure programmer aptitude. The Primary Mental Abilities Test has for years been used by the RAND Corporation and System Development Corporation as a programmer selection device (Perry and Cantley, 1965; Rowen, 1957). The Army uses a programming aptitude test developed by a civilian company and very similar to the ATPP. While the test is not included in our battery, the scores for military subjects were obtained for research purposes.

Reading Comprehension

Finally, the Entry Characteristics Test Battery includes one test of reading comprehension, the Reading Comprehension Cooperative English Test (Cooperative Test Division of Educational Testing Service). This instrument provides four scores: vocabulary, level of reading comprehension, speed of reading comprehension, and total reading comprehension (level + speed/2).

RESULTS

The dependent measures gathered in the experiment were initially analyzed by means of the analysis of variance; Figure 1 contains the underlying statistical model with annotation of each variable. Because it is a mixed model with nested factors, expected mean squares were calculated; these and the F ratios appropriate to testing each effect are shown in Figure 2.

The analyses of variance were computed on quiz scores for the first try only and on transit times for the first and second tries; beyond these limits, most of the cells of the design would be either empty or of unequal size.

The results of the analysis are presented in Table 2. Notice that the only statistically significant main effects ($p \leq .05$) occur for the units of instruction. The very large F ratios obtained for all transit times, and the smaller but significant ones for quiz performance, compilation errors, and LOA simply indicate that the instructional units varied in their difficulty and in their length. The few significant interactions that involve the learner control options are due to the fact, reported in the experimental design section, that the extent to which students make use of the options affects the amount of time spent in a topic.

Table 2

Analysis of Variance Results

Dependent Measure	Significant Main Effects & Interactions	Ratio F	Degrees of Freedom
Transit Time Telling/Practice Section (1st try)	Topics	105.88	20;1280
	Route X Quiz X Topics	1.86	20;1280
Transit Time Quiz Section (1st try)	Topics	254.40	20;1280
	Route X Topics	2.07	20;1280
	Review X Topics	1.60	20;1280
	Quiz X Topics	1.68	20;1280
	Route X Quiz X Topics	2.07	20;1280
Total Transit Time All Sections (1st & 2nd tries)	Topics	51.64	20;1280
	Recap X Review	4.01	1;64
	Review X Quiz	4.84	1;64
Transit Time Divisions	Divisions	41.41	1;64
	Recap X Review	4.01	1;64
	Review X Quiz	4.84	1;64
Quiz Score	Topics	1.75	25;1600
Compile Errors	Divisions	12.06	1;64
LOA	Objective	7.85	21;1008
	Route X Recap	5.41	1;48
	Route X Objective	2.48	21;1008
	Recap X Quiz	3.26	1;48
	Route X Review X Quiz	6.75	1;48
	Review X Quiz X Objective	1.55	21;1008

Statistical Model Underlying a 2^4 x n Repeated Measures Factorial Design

$$Y_{ijklmno} = \mu + a_i + b_j + c_k + d_l + s_{m(ijkl)} + f_n + ab_{ij} + ac_{ik} + ad_{il}$$
$$+ af_{in} + bc_{jk} + bd_{jl} + bf_{jn} + cd_{kl} + cf_{kn} + sf_{m(ijkl)n}$$
$$+ df_{ln}$$
$$+ abc_{ijk} + abd_{ijl} + abf_{ijn} + acd_{ikl} + acf_{ikn}$$
$$+ adf_{iln} + bcd_{jkl} + bcf_{jkn} + cdf_{kln}$$
$$+ abcd_{ijkl} + abcf_{ijkn} + acdf_{ikln} + bcdf_{jkln} + abcdf_{ijkln}$$
$$+ \varepsilon_{o(ijklmn)}$$

a_i = route option, where i = 1,2

b_j = recap option, where j = 1,2

c_k = review option, where k = 1,2

d_l = quiz option, where l = 1,2

$S_{m(ijkl)}$ = students, where m = 1,2,3,...,5

f_n = instruction segments, where n = 1,2 (course divisions)

n' = 1,2,3,...21 (division modules)

n'' = 1,2,3,...,26 (module objectives)

$\varepsilon_{o(ijklmn)}$ = experimental error, where o = 1,2,3,...,80

Figure 1

Expected Mean Squares for the Statistical Model Underlying a $2^4 \times n$ Repeated Measures Factorial Design (where n = 1,2,3, . . . ,26)

	Mean Squares for Computing Observed "F" Ratios	
a	$\sigma_\varepsilon^2 + n8\emptyset\sigma_s^2 + n32\emptyset\emptyset\sigma_a^2$	MS_a/MS_s
b	$\sigma_\varepsilon^2 + n8\emptyset\sigma_s^2 + n32\emptyset\emptyset\sigma_b^2$	MS_b/MS_s
c	$\sigma_\varepsilon^2 + n8\emptyset\sigma_s^2 + n32\emptyset\emptyset\sigma_c^2$	MS_c/MS_s
d	$\sigma_\varepsilon^2 + n8\emptyset\sigma_s^2 + n32\emptyset\emptyset\sigma_d^2$	MS_d/MS_s
s	$\sigma_\varepsilon^2 + n8\emptyset\sigma_s^2$	Not Testable
f	$\sigma_\varepsilon^2 + 8\emptyset\sigma_{sf}^2 + 64\emptyset\emptyset\sigma_f^2$	MS_f/MS_{sf}
ab	$\sigma_\varepsilon^2 + n8\emptyset\sigma_s^2 + n16\emptyset\emptyset\sigma_{ab}^2$	MS_{ab}/MS_s
ac	$\sigma_\varepsilon^2 + n8\emptyset\sigma_s^2 + n16\emptyset\emptyset\sigma_{ac}^2$	MS_{ac}/MS_s
ad	$\sigma_\varepsilon^2 + n8\emptyset\sigma_s^2 + n16\emptyset\emptyset\sigma_{ad}^2$	MS_{ad}/MS_s
af	$\sigma_\varepsilon^2 + 8\emptyset\sigma_{sf}^2 + 32\emptyset\emptyset\sigma_{af}^2$	MS_{af}/MS_{sf}
bc	$\sigma_\varepsilon^2 + n8\emptyset\sigma_s^2 + n16\emptyset\emptyset\sigma_{bc}^2$	MS_{bc}/MS_s
bd	$\sigma_\varepsilon^2 + n8\emptyset\sigma_s^2 + n16\emptyset\emptyset\sigma_{bd}^2$	MS_{bd}/MS_s
bf	$\sigma_\varepsilon^2 + 8\emptyset\sigma_{sf}^2 + 32\emptyset\emptyset\sigma_{bf}^2$	MS_{bf}/MS_{sf}
cd	$\sigma_\varepsilon^2 + n8\emptyset\sigma_s^2 + n16\emptyset\emptyset\sigma_{cd}^2$	MS_{cd}/MS_s
cf	$\sigma_\varepsilon^2 + 8\emptyset\sigma_{sf}^2 + 32\emptyset\emptyset\sigma_{cf}^2$	MS_{cf}/MS_{sf}
sf	$\sigma_\varepsilon^2 + 8\emptyset\sigma_{sf}^2$	Not Testable
df	$\sigma_\varepsilon^2 + 8\emptyset\sigma_{sf}^2 + 32\emptyset\emptyset\sigma_{df}^2$	MS_{df}/MS_{sf}

(Continued)

Expected Mean Squares for the Statistical Model Underlying a 2^4 x n Repeated Measures Factorial Design (where n = 1,2,3, . . . ,26) *(Continued)*

abc	$\sigma^2_\varepsilon + n8\emptyset\sigma^2_s + n8\emptyset\emptyset\sigma^2_{abc}$	MS_{abc}/MS_s
abd	$\sigma^2_\varepsilon + n8\emptyset\sigma^2_s + n8\emptyset\emptyset\sigma^2_{abd}$	MS_{abd}/MS_s
abf	$\sigma^2_\varepsilon + 8\emptyset\sigma^2_{sf} + 16\emptyset\emptyset\sigma^2_{abf}$	MS_{abf}/MS_{sf}
acd	$\sigma^2_\varepsilon + n8\emptyset\sigma^2_s + n8\emptyset\emptyset\sigma^2_{acd}$	MS_{acd}/MS_s
acf	$\sigma^2_\varepsilon + 8\emptyset\sigma^2_{sf} + 16\emptyset\emptyset\sigma^2_{acf}$	MS_{acf}/MS_{sf}
adf	$\sigma^2_\varepsilon + 8\emptyset\sigma^2_{sf} + 16\emptyset\emptyset\sigma^2_{adf}$	MS_{adf}/MS_{sf}
bcd	$\sigma^2_\varepsilon + n8\emptyset\sigma^2_s + n8\emptyset\emptyset\sigma^2_{bcd}$	MS_{bcd}/MS_s
bcf	$\sigma^2_\varepsilon + 8\emptyset\sigma^2_{sf} + 16\emptyset\emptyset\sigma^2_{bcf}$	MS_{bcf}/MS_{sf}
cdf	$\sigma^2_\varepsilon + 8\emptyset\sigma^2_{sf} + 16\emptyset\emptyset\sigma^2_{cdf}$	MS_{cdf}/MS_{sf}
abcd	$\sigma^2_\varepsilon + n8\emptyset\sigma^2_s + n4\emptyset\emptyset\sigma^2_{abcd}$	MS_{abcd}/MS_s
abcf	$\sigma^2_\varepsilon + 8\emptyset\sigma^2_{sf} + 8\emptyset\emptyset\sigma^2_{abcf}$	MS_{abcf}/MS_{sf}
acdf	$\sigma^2_\varepsilon + 8\emptyset\sigma^2_{sf} + 8\emptyset\emptyset\sigma^2_{acdf}$	MS_{acdf}/MS_{sf}
bcdf	$\sigma^2_\varepsilon + 8\emptyset\sigma^2_{sf} + 8\emptyset\emptyset\sigma^2_{bcdf}$	MS_{bcdf}/MS_{sf}
abcdf	$\sigma^2_\varepsilon + 8\emptyset\sigma^2_{sf} + 4\emptyset\emptyset\sigma^2_{abcdf}$	MS_{abcdf}/MS_{sf}
ε	σ^2_ε	

Figure 2

Absence of significant effects of any option on quiz performance implies that the independent variable made no difference in the learning. Examination of the overall performance by the students suggests why these results occurred. Figure 3 indicates that the great majority of the students did quite well in the course. Notice that the frequency distributions of objectives passed on first try for both Divisions A and B are severely truncated with a marked positive skew. Seventy-five percent of the students passed at

Frequency Distribution of Subjects, by Percentage of Learning Objectives Passed on First Try

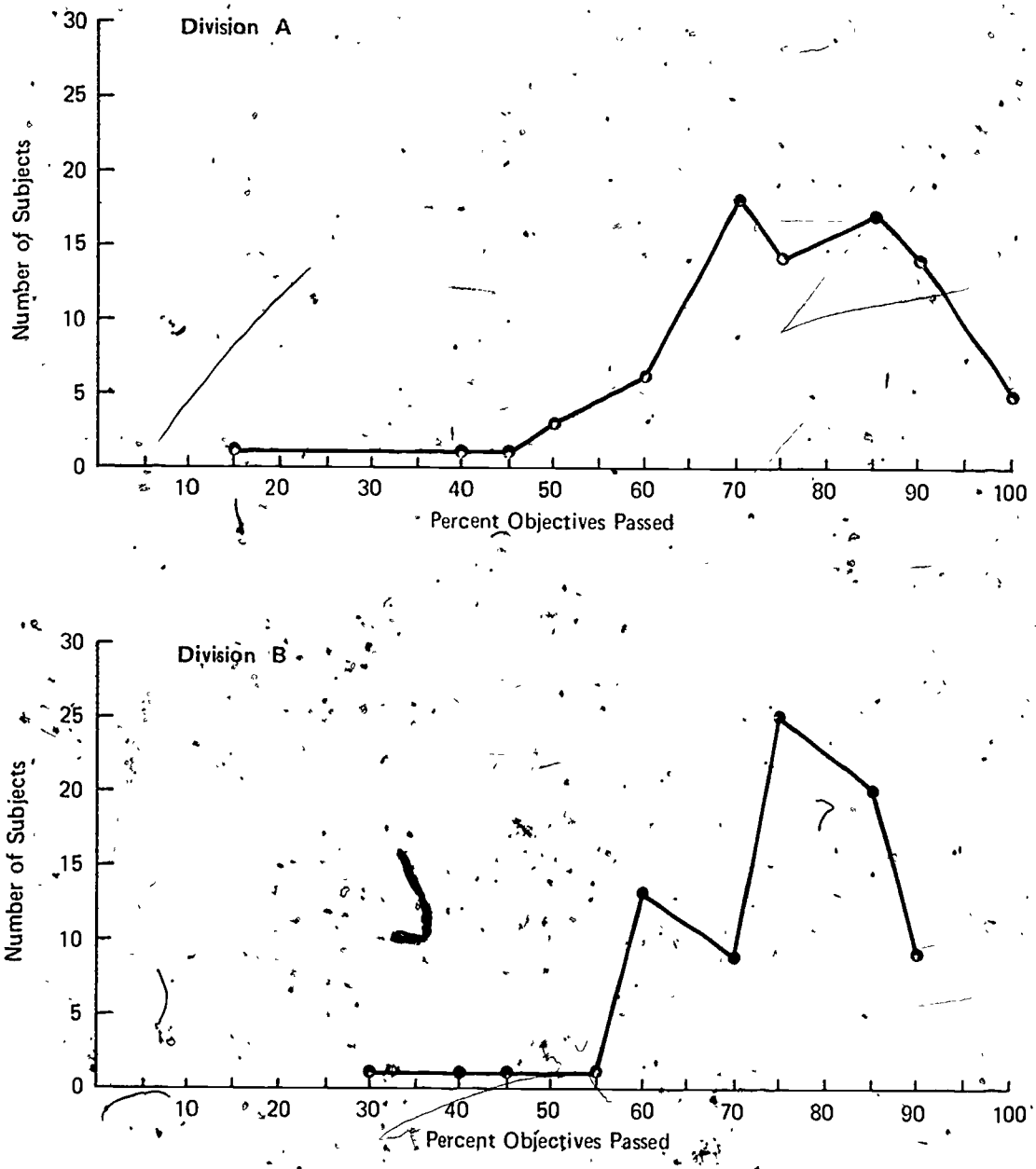


Figure 3

least 70% of the objectives on the first try for Division A; fully half the students passed 75% of the objectives on the first try. The results for Division B are comparable. First quartile scores for Divisions A and B are 90% and 85% respectively. Third quartile scores are identical, 70%.

These data strongly suggest that the limited variability of student performance, coupled with a restricted range and severely skewed distribution rendered a conventional technique such as the analysis of variance inappropriate as a means of analyzing the effects of the independent variables. Such an occurrence is not uncommon in educational research where the instructional material used as vehicle for experimentation must also meet the requirement that it teach well.

HIGH AND LOW PERFORMERS

It was decided that, under the circumstances, a potentially fruitful way of investigating the effects of learner control would be through a comparison of the best and the worst students in the sample for any treatment differences (i.e., in the use of the options). The "high/low" performer technique used in the development of psychological inventories (Brennan, 1970) was applied. The individual items are analyzed for their capacity to discriminate between those whose overall test scores are high and those whose scores are low. In the present study we wish to analyze the way learner control options are used, as discriminators of high and low performers in the COBOL course.

The following questions were posed for this aspect of the analysis:

(1) Who are the high performers in Division A of the course? Who are the low performers?

(2) Who are the high performers in Division B? Who are the low performers?

(3) Do they differ in the way or manner in which they make use of the options?

(4) Is there any difference in the attributes of the high and low performers which might help explain or predict differences in option use?

Selection

In order to have a sufficient number of observations for the analysis, it was decided that the 20 highest and lowest performers in each division would be identified for further study. A combination of absolute and relative performance criteria were used to select them; the specific criteria for high and low performers in each division are shown in Table 3.

Manner of Option Use

The frequency with which the high and low performers used the options is presented in Table 4. The values have been adjusted to equate option availability across groups. This was necessary because, in some instances, the high and low performers did not come from treatment conditions having the same degree of access to options. (Because second attempts occurred with relatively low frequency, these data on option usage are not reported here.) Data on the ROUTE option usage are presented in Table 5.

Table 4 shows that the low performers consistently use the options more frequently than the high performers in both divisions. This suggests that, if the options aid learning at all, the gain is due not to how often they are used but rather to where and when they are used.

Table 3

Selection Criteria for High and Low Performers

(Divisions A and B of the COBOL Course)

High Performers on Division A

Criterion 1: Passed ⩾ 84.6% of first-attempt objectives.

Criterion 2: Of first-attempt passed objectives, 72.7% were ⩾ 100 percentile.

Criterion 3: Top 20 ranked according to score

$$\left(\text{score} = \Sigma\left(\frac{\text{obtained } S_i}{\text{max obtained } S_i}\right)\right)$$

Low Performers on Division A

Criterion 1: Failed ⩾ 30.8% of first-attempt objectives.

Criterion 2: Of failed first-attempt objectives, 75.0% were ⩽ 23.75 percentile.

Criterion 3: Bottom 20 ranked according to score

$$\left(\text{score} = \Sigma\left(\frac{\text{obtained } S_i}{\text{max obtained } S_i}\right)\right)$$

High Performers on Division B

Criterion 1: Passed ⩾ 84.6% of first-attempt objectives.

Criterion 2: Of first-attempt passed objectives, 66.6% were ⩾ 98.75 percentile.

Criterion 3: Top 20 ranked according to score

$$\left(\text{score} = \Sigma\left(\frac{\text{obtained } S_i}{\text{max obtained } S_i}\right)\right)$$

Low Performers on Division B

Criterion 1: Failed ⩾ 23.1% of first-attempt objectives.

Criterion 2: Of first-attempt failed objectives, ⩾ 75.0% had scores ⩽ 26.25 percentile.

Criterion 3: Bottom 20 ranked according to score

$$\left(\text{score} = \Sigma\left(\frac{\text{obtained } S_i}{\text{max obtained } S_i}\right)\right)$$

Table 4

Frequency of First Attempt Option Usage

(Adjusted)

Performers	Options		
	RECAP	REVIEW	QUIZ
Division A			
High	6.2	12.0	15.0
Low	35.0	65.0	25.0
Division B			
High	15.0	4.0	8.8
Low	20.0	52.5	11.1

Table 5

Use of the ROUTE Option by High and Low Performers

Performers	ACTIVE	PASSIVE	TOTAL
Division A			
High	14	5	19
Low	10	10	20
Division B			
High	21	11	32
Low	18	19	37

Students who had the ROUTE option could, when presented with a "menu" of available topics, choose to pick their own or defer to the system to pick one at random. Table 5 shows that the ROUTE option was made available by the IDM far more often in Division B than in Division A; this occurs because the prerequisite structure in Division B is less ordered.[1] In both divisions, the high and low performers saw menus about the same number of times. However, the proportion of occurrences in which subjects made an active selection of the next topic differs markedly between the highs and lows. In Division A the high performers made their own choice nearly three times more often than not; the low performers actively chose only half the time. In Division B the high performers chose nearly twice as often as they deferred to the system; here, the low performers also chose about half the time.

Personal Attributes

In an effort to identify the cognitive and affective characteristics on which high and low performers differ, the Entry Characteristics Test scores of these groups were subjected to multiple stepwise discriminant analysis. The results are summarized in Table 6.

The overall discrimination for both divisions was highly significant (for A the overall F was 10.154; for B the F was 9.609; $p<.01$). In Division A three of the 35 entry characteristics scores were selected by the analysis: the Primary Mental Abilities Test, a measure of general verbal aptitude; Match Problems, Test V, a measure of figural adaptive flexibility; and the IPAT Anxiety Test—Score B, a measure of the extent to which an individual reports anxiety-related feelings or behaviors. Examination of the standardized coefficients for these variables shows that the greater part of the discrimination is due to the verbal abilities test. The positive sign of the coefficients indicates that the high performers possess these attributes to a greater degree than the low performers.

For Division B the analysis identified programming aptitude, the Aptitude Test for Programming Personnel, and vocabulary, measured by the Cooperative English Test, as the principal discriminators between the high and low performers. Two tests—the PMA Spatial Relations and the Letter Sets Test (a measure of inductive reasoning ability)—are not easily interpreted in the present context. The negative sign of the coefficients means the high performers possess less of these attributes than the low performers, although one would expect a positive relation with success in the course.

[1] See Figures 8 and 9.

Table 6

Summary of Stepwise Multiple Discriminant Analysis of High and Low Performers on 35 Entry Characteristics Test Scores

Division	Test Selected	Standardized Discriminant Weight
A	PMA–Verbal	7.738
	IPAT Anxiety–Score "B"	1.560
	Match Problems V	2.506
	Significance of overall discrimination: F (3,36) = 10.154; $p << .01$	
B	Cooperative English–Vocabulary	19.742
	ATPP	13.314
	PMA–Spatial Relations	-6.57
	Letter Sets	-2.693
	First and Last Names	3.007
	Significance of overall discrimination: F (6,33) = 9.609; $p << .01$	

SELF-ASSESSMENT

Two techniques of self-assessment as potential decision factors were studied in this research effort: expectancy (LOA) and confidence measures.

Expectancy Measure

During the first iteration of IDM investigation, LOA (Level of Aspiration) was studied as a correlational variable. LOA correlated significantly with criterion performance in both levels of complexity in COBOL1. The correlations were +.42 and +.53 ($p<.01$) respectively. These findings were consistent with previous results in programmed instruction (see Seidel and Hunter, 1970).

It was therefore decided to make LOA part of the decision-making strategy in the revised IDM for COBOL2. Extensive course revision coupled with curtailed resources prevented implementation of LOA as the newly developed Expectancy Operator, but LOA was used again as a correlational variable. The results were consistent with the previous data, indicating significance of student expectations as predictors of achievement.

Because of the modularization of COBOL2, it was possible to perform a finer-grain analysis than previously. The LOA data were analyzed at the level of specific objectives. The basic hypothesis tested was that high performers would be more realistic than low performers. Operationally this would take the form of (a) a smaller positive discrepancy score for the high performers, and (b) a smaller absolute value of discrepancy between LOA and objective score for the high performers.

A ceiling effect because of the excellent performance by the high performers prevented an analysis of the *signed* differences. However, the test of absolute value differences as revealed in Figure 4 clearly supported the hypothesis that high performers would be more realistic than the low performers. Coupled with the previous LOA findings these results substantiate the value of providing an Expectancy Operator for remediational purposes as part of an improved IDM.

The fact that the LOA measures indicated a high degree of value to self-assessment and the fact that other studies (e.g., Shuford, Albert, and Massengill, 1966) supported the value of confidence measures as an aid to learning led us to re-evaluate the ways in which we would implement confidence measures in COBOL2 (rather than eliminating confidence as a sensitive index of state of understanding).

The goal in COBOL2 was to make the implementation easier for the student to use, make all input responses equivalent in effort and difficulty, lessen the frequency with which the confidence measures were used to avoid interrupting and interfering with the learning process, and lastly to increase the value of providing confidence measures by making associative materials attached to the various states of understanding more meaningful and positive than they had been for the student in COBOL1.

The redesign was accomplished and initial off-line preliminary testing was achieved with staff members of the research project. However, because of the limited resources and other difficulties cited earlier, the re-implementation of confidence measures was not accomplished during this research project. We feel, nevertheless, that, in combination with the Expectancy Operator as discussed above, the confidence measures should provide a very sensitive component to revised decision-making rules taking into account student motivation. The suggested implementation of confidence measures is provided as follows.

Confidence testing would be part of the Q-sections of the course, COBOL2, and handled in the following manner.

For the constructed response type of question, the student, after making sure that his answer is the one he wishes to have recorded and checked, will input his response. His display (CRT, hard copy, etc.) will be cleared and a confidence question will be displayed. This confidence question will summarize the task asked of the student and ask him to place his confidence in a prescribed location on the display.

The student's confidence will be indicated by his selection of one of 11 characters from his keyboard. The characters are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, and T-, where T stands for 10. A computer program computes the number of points the student receives by multiplying the number of points a question is worth (10-99), as determined by the author, by a three-place decimal associated with the student's confidence (see Table 7). For a correct response, the student receives that number of points. If he is incorrect, he

Table 7

Scoring System for Confidence Measure

Student Inputs for Correct Alternative	Payoff
0	0.000
1	0.500
2	0.650
3	0.739
4	0.801
5	0.850
6	0.889
7	0.923
8	0.952
9	0.977
T	1.000

The rationale behind the Expectancy Operator is based upon the relationship between a student's performance and the relative reality of his Expectancies. If a student is judged to be unrealistic—too great a discrepancy between LOA and performance—a Probe path analysis and remediation are advocated. Recommended instructional guidelines for initial use of the Expectancy Operator and probe path are as follows.

It is suggested that implementation of the Expectancy Operator (LOA) take the form of providing for LOA measurement prior to a Q-section, criterion test on specific subsets of objectives, and measurements of student performance for each module (subset of objectives). The IDM takes specific action based on a comparison of these measurements. The action to be taken by the IDM is based upon a decision of a problem in *reality* estimation or conceptual understanding, or some combination. In essence, the discrepancy score relationship between the LOA difference score in combination with percent correct (and when refined, confidence) is to be used for determining whether or not a Probe path should be followed. A Probe path is initiated by the system in order to gain more information concerning the student's problem and in order to take appropriate remedial action.

For example, if, following the pretest, the alignment of realism and percent correct represents "appropriate" behavior, then the student is to follow the "normal" path for him. (Normality in this case is to be defined idiographically with a continually refined entry battery.) If discrepancies occur in the student's estimations of reality, then the Probe path will be followed for additional diagnosis and action (by noting the relationship between the measures of percent correct and the measures of the LOA).

The specific plan and guideline for initial implementation of an Expectancy Operator in a tutorial environment follows.

1. Objective for using the Expectancy Operator: To lessen the relative distance between level of aspiration and performance by raising the level of the lower (LOA or performance) to meet the higher value.
2. Plan for measuring LOA and providing solutions:
   a. Measure LOA prior to entering the Quiz.
   b. Measure performance in the Quiz.
   c. Feedback to student on LOA versus performance, verbal plus numerical comparison.
   d. Provide solution.
      (1) Solution to raise performances to LOA: Follow-up alternate strategy with alternate quiz (check discrepancies).
      (2) Solution on subsequent modules to bring confidence level up to performance: Check discrepancy scores on subsequent modules.
3. State DIAGNOSES: Eighteen possible states are derivable from the three sets of characteristics listed below. For purposes of initial implementation it is suggested that A3a, A3b, C1a, C1b will be most useful as additions to an adaptive IDM.

LOA	PERFORMANCE	ABILITIES
A High	1 High	a Above Average
B Average	2 Average	b Below Average.
C Low	3 Low	

4. Set of Alternative ACTIONS:
   a. Remedial modules geared to specific content failures concentrating on variation of practice exercises.
   b. Conference with proctor/instructor.

c. Fun Option (on-line games).
d. Skip ahead (practice).
e. Leaving early.
f. Alternate media module (CMI type, cassette, PI text).
g. Compliment.
h. Do nothing.

5. Recommended ACTIONS for DIAGNOSES (A3a, A3b, C1a, C1b):
   c, d, e, g: Confidence Building (C1a, C1b)
   a, b, f: Performance Building (A3a, A3b)
6. Actions to be implemented initially:
   A3a: a
   A3c: b and/or f
   C1a: d, e, f, and/or g
   C1c: g or h

General illustrations of how the Probe path would operate are as follows. First, let E stand for the student's estimate of his performance after the fact; let L stand for his level of aspiration or expectancy before the fact; and let A stand for his actual performance. The definition of i is trial number of a referent for the particular measurement number of LOA or EST. Applied to COBOL2, it would reference module number. Generally, the values for L, E, and A would be derived from the degree of criterion attainment determined for a particular application in a given computer-based environment.

A: DIAGNOSIS: If $|L_i - A_i| > |E_i - A_i|$ & $(E_i^{disk} = +)$ then student state is defined as REALISTIC to the IDM.

ACTION: Diagnosis proceeds to next stage in IDM. Score on Test of Objectives, percent correct, defines CONCEPTUAL state and confidence value defines REASSURANCE state. Given A, and passage of criteria here, student continues on "normal" module path available (based on current options—eventually to be redefined by our improved Entry Battery and better within-course historical predictors). Subject possesses the three R's—Realistic, Reassured, and Right (see Figure 5).

B. DIAGNOSIS: If $|L_i - A_i| < |E_i - A_i|$ $(E_i^{disk} = -)$.

ACTION: Go to PROBE path.

PROBE path: Here IDM can be thought of in the following way.

(1) It can query student directly to determine the nature of the problem as perceived by the student; that is, the student says, "No problem," or "I don't think I can hack it;" or "I think I understand this stuff, but I'm not sure" (or some variation on this theme).

(2) Diagnosis of problem is defined as the intersection of three orthogonal binary dimensions. The resulting state is estimated as follows (verbally below and pictorially in Figure 5):

(a) Given B above (diagnosis of UNREALISTIC) and Low Confidence and High Objectives score, then two dimensional motivational problems exist—REASSURANCE and REALISM.

(b) Given B, and Low Confidence and Low Objectives score, then problem is diagnosed as both overall motivational and CONCEPTUAL.

(c) Given B, High Confidence and High Objectives score, then problem is uniquely one of REALISM (e.g., the pessimist even though confident at time answering).

(d) Given B, High Confidence and Low Objectives score, then also REALISM, but probably of different type (e.g., delusions of grandeur).

(e), (f), and (g): Cases currently handled by the IDM where the Expectancy Operator would be in the zero condition.

Illustration of Diagnostic States

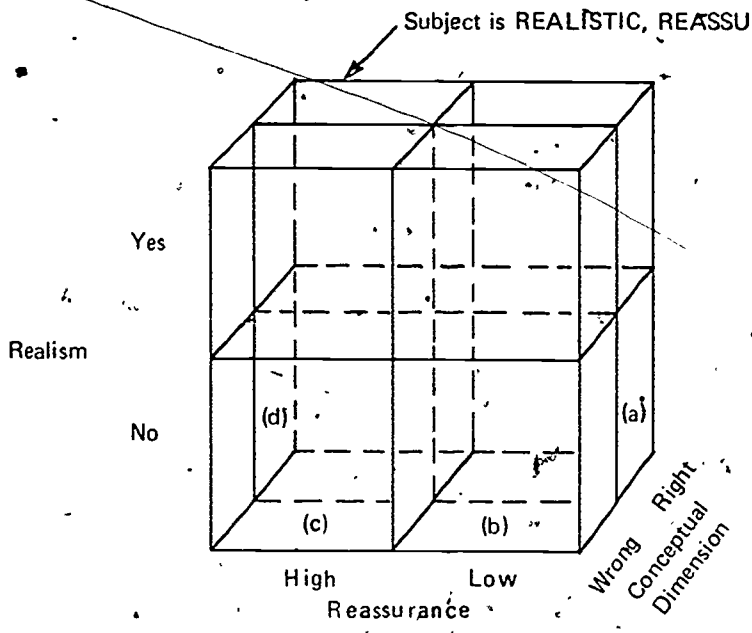


Figure 5

Mathematically (and for IDM use) the state diagnosis can be described by ordered triples where 1 = a problem condition, a remedial operator is called for, and 0 = no problem. Thus, reading Realism, Reassurance, and Conceptual dimensions from left to right:

Case (a) is described as	<1,1,0>		
Case (b)	<1,1,1>		
Case (c)	<1,0,1>		
Case (d)	<1,0,0>		
Cases (e), (f), and (g)	<0,1,0>	<0,1,1>	<0,0,1>

Confidence Measure

A second measure of assessment used in our IDM research effort was confidence responding by the students. In COBOL1, unlike the implementation of LOA measures, confidence responding was part of every student response. The student gave an answer to a question and immediately thereafter distributed his confidence with respect to the answer over a series of alternatives if the alternatives were available; or he attributed a degree of confidence to the correctness of the answer he provided in a completion type format.

The results, as reported in the Technical Progress Report, 1970-71, indicated a lowering of the correlation between confidence measures and correct responding as the student progressed through the 18 modules of COBOL1. The implementation scheme apparently was not a useful one for the students. They were required to give a percentage value between 0 and 100% using two digits as appropriate (e.g., 45, 55) and eventually adopted a principle of least effort. That is, the students either used a 0 or 100% confidence choice eventually, and the result was a lessening of the value of the confidence measure as an indicator of a state of understanding on the part of the student.

The fact that the LOA measures indicated a high degree of value to self-assessment and the fact that other studies (e.g., Shuford, Albert, and Massengill, 1966) supported the value of confidence measures as an aid to learning led us to re-evaluate the ways in which we would implement confidence measures in COBOL2 (rather than eliminating confidence as a sensitive index of state of understanding).

The goal in COBOL2 was to make the implementation easier for the student to use, make all input responses equivalent in effort and difficulty, lessen the frequency with which the confidence measures were used to avoid interrupting and interfering with the learning process, and lastly to increase the value of providing confidence measures by making associative materials attached to the various states of understanding more meaningful and positive than they had been for the student in COBOL1.

The redesign was accomplished and initial off-line preliminary testing was achieved with staff members of the research project. However, because of the limited resources and other difficulties cited earlier, the re-implementation of confidence measures was not accomplished during this research project. We feel, nevertheless, that, in combination with the Expectancy Operator as discussed above, the confidence measures should provide a very sensitive component to revised decision-making rules taking into account student motivation. The suggested implementation of confidence measures is provided as follows.

Confidence testing would be part of the Q-sections of the course, COBOL2, and handled in the following manner.

For the constructed response type of question, the student, after making sure that his answer is the one he wishes to have recorded and checked, will input his response. His display (CRT, hard copy, etc.) will be cleared and a confidence question will be displayed. This confidence question will summarize the task asked of the student and ask him to place his confidence in a prescribed location on the display.

The student's confidence will be indicated by his selection of one of 11 characters from his keyboard. The characters are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, and T-, where T stands for 10. A computer program computes the number of points the student receives by multiplying the number of points a question is worth (10-99), as determined by the author, by a three-place decimal associated with the student's confidence (see Table 7). For a correct response, the student receives that number of points. If he is incorrect, he

Table 7

Scoring System for Confidence Measure

Student Inputs for Correct Alternative	Payoff
0	0.000
1	0.500
2	0.650
3	0.739
4	0.801
5	0.850
6	0.889
7	0.923
8	0.952
9	0.977
T	1.000

receives the number of points found by multiplying the three-place decimal associated with the ten's complement of the student's confidence by the author's point value for the question.

For example, if the student places a 7 in his confidence block, then the ten's complement is taken as his "no confidence" response, in this case, 3. Suppose the author states that the question is worth 60 points; then, if the student is correct he receives 60 x 0.923 = 55.3 rounded to 55 points, and if he is incorrect he receives 60 x 0.739 = 44.3 rounded to 44 points.

Each of the computer point values rolls onto the display in the proper location. If the student is dissatisfied with the number of points he will receive, he will change his confidence. The computation will be done again. It can be done as many times as the student wishes until he is satisfied with his potential number of points. He will then signal his completion by proper key press, and the appropriate number of points will be credited to him.

The student will receive a feedback message on the display if his confidence is not one of the 11 characters named above.

For multiple-choice questions, the student will distribute his confidence over all alternatives (as in COBOL1). His confidence must add to T (ten) and he must strike a character for each alternative. Once again, the weight of the question supplied by the author will be multiplied by the three-place decimal associated with the student's response. These products will be rolled onto the display in payoff fields next to each alternative. If the student is satisfied, he just presses the appropriate key. If he is not satisfied, he will change his confidences until he is happy with the payoff involved. When it is found that the student has pressed his key without changing his confidence assignments, he will be awarded the number of points he has assigned next to the correct alternative.

For example, if a question is worth 50 points, and the student distributes his confidence as follows on a four-alternative question whose second alternative is the correct one, he will receive 33 points:

C		Payoff	
2	A.	33	Alternative
2	B.	33	Alternative
6	C.	44	Alternative
0	D.	0	Alternative

Here too, should the student type a character into his confidence that is not one of the 11 characters mentioned, or if his confidences do not add to 10, he will receive a feedback requesting him to correct his error in assigning his confidence to each of the alternatives.

In sum, it is felt that a combination of reality testing, confidence measuring, and conceptual responding would form a useful baseline of a next generation instructional strategy (IDM). Research on refinement of decision rules using these dimensions should aid further development of useful computer-based instructional materials in meaningful tutorial environments.

IMPLICATIONS OF EXPERIMENTAL FINDINGS

Interpretation of the data can best be presented in terms of guidelines for prescribing instructional management.

With respect to option control by students, the data suggest that we:

(1) Establish high and low performer initial predictions for sets of similar instructional tasks.
(2) Provide maximal student-control for the predicted high performers.
(3) Design maximal *system* control for predicted low performers.
(4) Track the performance of *all*.
(5) Adjust the degree of student-control, based upon the changes in performance according to an empirical model that maintains the maximum percent of high performers. This model would have to be derived for each subject-matter application since tasks and instructional materials would have unique complexities and forms.

Related to the above is the ability to be able to discriminate, using some entry test battery, the high and low performers on an initial basis. To the degree that this can be done, the designated student or system control options will be more or less appropriate.

As noted from the above ECT analyses in the current study with a primarily verbal course like COBOL2, the single best predictors were verbal entry tests such as the Primary Mental Abilities Test used for programmer aptitude selection, as well as the Cooperative English and, in another case, a uniquely suited Programmer Aptitude test called the ATPP. The significance of the ATPP, as well as the other structure of intellect tests which showed up as significant in our discriminant analysis, emphasizes that there will be other unique characteristics of any given instruction which will also aid in discriminating the predicted high from the low performers. These factors would have to be discerned from a structural analysis of the subject matter and its related tasks. (In the current instance, we are still doing analysis of this subject matter by factor structure using multiple raters to arrive at a reliable index, vis-a-vis the Guilford Structure of Intellect characteristics.)

Again, from the current study, supportive evidence for differential task transfer and the contribution of unique task and subject-matter characteristics comes from the comparison of Division A and Division B predictors from the ECT batteries. For example, in the introductory part of the course, Division A, the discriminant function was characterized by the most general and smallest number of predictor tests. Whereas in Division B, which was more heavily loaded with unique characteristics of COBOL programming and specific technological tasks, there were more variables present, and we found that the characteristics of these ECT predictors were more unique to the programming and specific tasks related to factor structure. Specifically, we refer to the fact that the ATPP, a uniquely oriented programmer aptitude test, was a heavily weighted factor under Division B prediction. Moreover, there was the appearance of structure of intellect factors involving associative memory and logical reasoning. In like manner, the anxiety test, IPAT B, was a predictor of performance in the introductory part of the course but dropped out as people became more familiar, comfortable, and sophisticated in the COBOL programming tasks.

Carrying this logic further, it would have been very interesting to test this hypothesis of specific task transfer with even more unique and specific, sophisticated COBOL course materials like the other two divisions which were not available for our experimental subjects during the conduct of this research. It would be predicted that probably more specific factors like logical reasoning and figural adaptive flexibility (the ability to change set with new materials) would take on even greater importance, all relative of course to the specifc nature of the task the individual would be encountering.

Combining the above noted discussion of option availability and relevant ECT for prediction of high or low performers with the previous description of the self-assessment results provides an indication of a workable instructional decision model to be tested in future research. It would take something like the following form.

Given that the high performer predicted by specific entry tests is more efficient in his use of options and is more realistic about his own performance, the self-assessment via the use of LOA could be used as a tracking device or technique for adjusting the degree of student control over the available remediational or accelerating options within the course of instruction. When a predicted low performer, for example, begins to fall within the realistic range of predicted high performers, that individual would then be allowed more control over the use of available options.

From the other side of the coin, when an individual falls outside the range of reality testing and is predicted to be a low performer, then the adjustment would take the form of eliminating student control over available options until such time as the individual performance begins to come more in line with reality and, indeed, until the predicted performance jumps back up to what a high performer would be. This model, however, does require continued research in order to verify its appropriateness to various applications of instructional tasks.

As an adjunct to this model, it may also be relevant to add other parameters which describe in a more sensitive way the high or low performer's state of understanding. This might be done by use of the revised confidence measures discussed earlier under self-assessment. It may well be that a previously designated high performer who yields some unrealistic estimates of performance in some novel material might be signaling that he is getting into deep water and can no longer handle the instructional tasks required of him. In this case, supplementing the probing of that individual's understanding by the use of confidence techniques might provide additional indices for the kind of specific remediation unique to his requirements. All of the above awaits further verification in real-world instructional environment, similar to that used within the current study.

FORMAL THEORETICAL MODEL FOR INDIVIDUALIZED INSTRUCTION

The formal theoretical research has concentrated on the development of a mathematical model for individualized instruction. To this end we have concentrated on the first four components of Pask's (1969) five requisite ingredients for a model for instruction:

(1) A representation of the subject matter to be taught.
(2) A representation of the educational goal—both terminal behavior and cognitive components as appropriate.
(3) A representation of the initial student-state or the state of the student upon entering the system.
(4) A representation of the current student-state, or the state of the student at any time.
(5) A representation of the teaching system, including its teaching strategies.

The model developed for individualized instruction is at present primarily descriptive, with extensions under way to prescriptive guidelines for developing an individualized instructional environment. The work to date can be summarized under four headings:

(1) Structuring Subject Matter
(2) Representing Student States
(3) Representing the Goal of Instruction
(4) Identifying Paths Through the Subject Matter

Consider the first problem: Structuring Subject Matter. Historically, attempts to characterize subject-matter structures seem to fall into two classes. On the one hand, researchers such as Gagne (e.g., 1965, 1969) have focused on task structures. Gagne's formula, "What must one be able to do before. . .?" leads to structures that focus on the

tasks a student must learn how to perform. Researchers such as Pask (1969), Guilford (1967), and Gagne (1971), on the other hand, have focused on the content of subject matter. Pask (1969) structures subject matter along the lines of the formula: "What must a person know before he can learn. . .?"

Thus, an adequate model for subject-matter structure must include both task- and content-oriented components. Prescriptively, our model asserts that before specifying tasks, behavioral objectives must be known. Before specifying subject-matter content, the body of knowledge to be studied must be defined. Neither of these steps can be made in a vacuum. The context in which instruction is to be offered must be specified clearly before either behavioral objectives or the details of the body of knowledge can be specified. The model asserts that the target population determines the context for the instruction. Furthermore, the general body of knowledge mediated by the context of instruction yields the requisite detailed description of the body of knowledge. Space prohibits more than this very brief summary of the prescriptive aspects of the model for subject-matter structure.

Descriptively, the model yields two primary components: The Task in Context Structure (TICS) and the General Cognitive Net (GCN) for the context-conditioned communicable body of knowledge. Together, these components and their relationships make up the subject-matter structure. Both the TICS and the GCN are graphs. Graphically, they can be represented as a set of nodes with connecting, directed arcs. The term graph is used to indicate that there are no restrictions concerning the number of in-directed and out-directed arcs from any node such as would be imposed by the mathematical concept of a function. The descriptive aspects of the model for structuring subject matter are included in a separate report (see Appendix A). The document includes a statement of the theoretical axioms for the GCN and TICS, development of the notion of dependency, and a comprehensive application of the model.

The last three theoretical topics: Representing Student States, Representing the Goal of Instruction, and Identifying Paths through the Subject Matter will be dealt with in a later paper (in preparation). Briefly, the methodology of Inductive Logic (Carnap, etc.) has been adopted to achieve a representation of initial and current states. A set theoretic representation of the goal of instruction is then developed from the student state descriptors. Finally, the notion of dependency developed within the subject-matter structure portion of the model is applied to eliminate some student states as being impossible. The dependency relation then can be shown to constrain the sequence of the remaining possible student states which the individual can progress through. Finally, an algorithm is developed that leads to the identification of all possible paths (successive student states) through the subject matter.

Section 4

COURSE DEVELOPMENT AND IMPLEMENTATION

HISTORY OF COBOL DEVELOPMENT

COBOL1 was administered first to high school students for debugging and then to Army students for evaluation. Data were collected and analyzed on a total of 42 students. The evaluative data showed that subsets of instruction needed to be smaller, for both learning and research purposes. Thus arose the modular concept employed in COBOL2, the second iteration COBOL course developed by HumRRO. Also, more opportunities for program writing and training in debugging procedures (through gaming) were provided for the student.

The IDM and associated support software also underwent changes at this point, with a major element being the separation of the logic of the IDM from the instruction. The ramifications of this development were enormous because the IDM could be dynamic and unconstrained by the course logic.

The COBOL2 course and associated logic were debugged and evaluated with 83 students of various civilian and military backgrounds. Add to this the 80 experimental subjects and the number of students who have taken the COBOL materials totals to 205. During the course of running all these students and the concomitant effort made to perfect the COBOL materials, a complete instructional system evolved, including managerial, administrative, production, and evaluative tools and procedures.

The topics addressed in the remainder of this section are: course design, author support, components of the courseware subsystem, production process, operation and administration, and revision process.

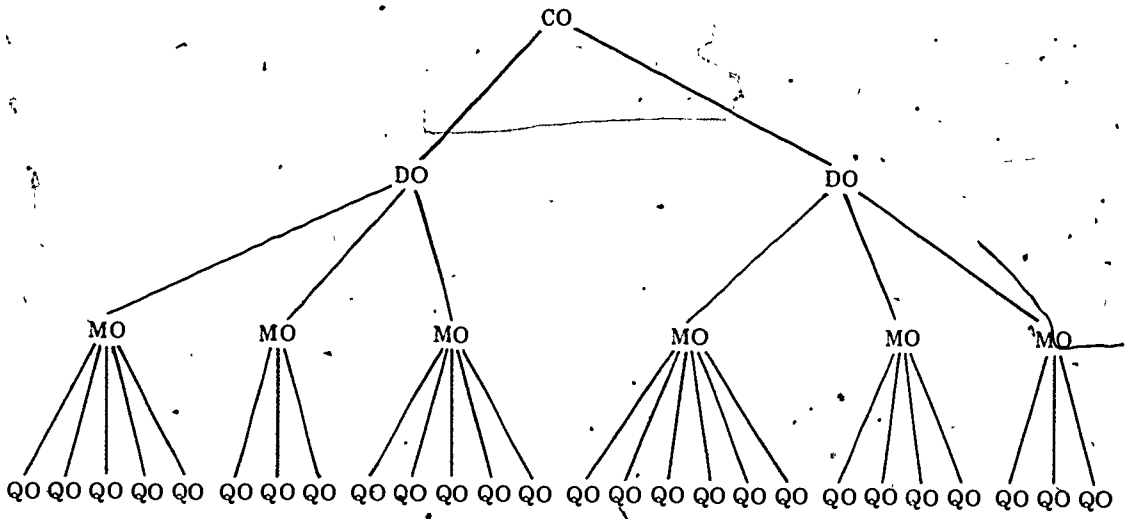
COURSE DESIGN

Within the IDM the rules for presentation of course material are defined and the software interface operationalizes these rules. Since the whole instructional system is based on rules and the practical hardware/software limitations relevant to their execution, the course author must structure his material so that it is compatible with both the rules, IDM, course structure, and limits of execution.

Since the course is based on a well-defined schema of objectives, the first constraint on an author is that his material must be structured so that it fits the design of the course objectives—that is, that there is a singular terminal behavior defined by the single course objective, and the question objectives are single enabling objectives. The module and division objectives are of an intermediate type, whereby they are enabling to whatever is above them (see Figure 6) and terminal to whatever is below them. Because the module objective is the lowest-order terminal objective, it becomes the logical unit of instruction for an author.

The course structure (Figure 7) reflects the essential elements of the objective schema. The set of behavioral objectives for a Division is sufficiently large and complex to require further partitioning from subsets to sub-subsets. This produces a Module.

Hierarchy of Objectives



Where: CO = Course objective
DO = Division objective
MO = Module objective
QO = Question objective

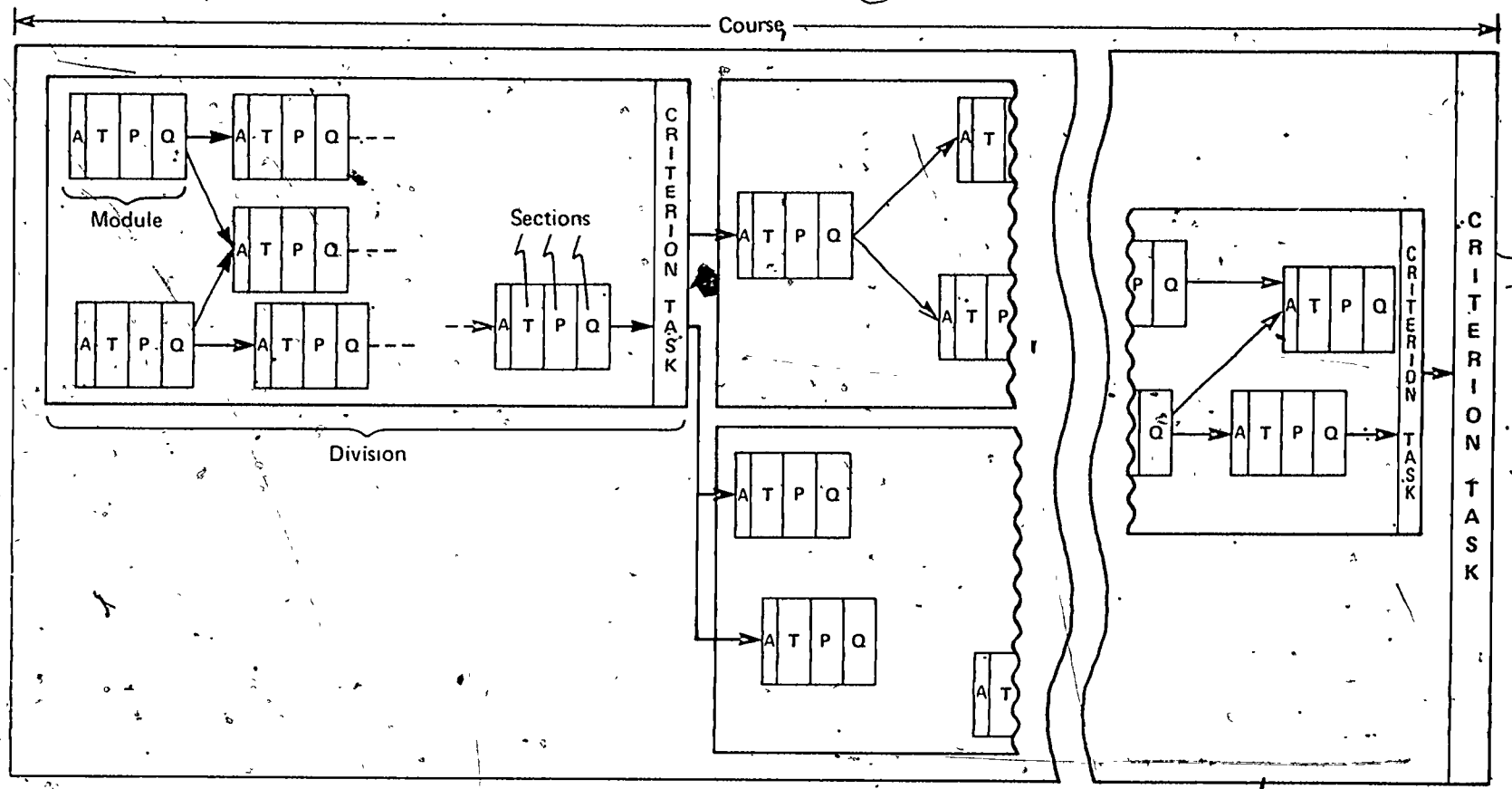
Figure 6

Modules are partitioned into sections. The first is the A (administrative) section, which contains a variety of administrative documentation that the student does not see, for the most part, but which is used to provide course management information. Second is the T (telling) section in which relevant subject-matter information is presented to the student. Third is the P (practice) section which permits the student to practice objective related behavior. The P-section is followed by the Q (quiz) section which tests for achievement of the behavioral objective(s). A module can have several versions of each section. The limitation is, of course, that each version covers the same basic material and therefore teaches to the same objective. The differences between versions are differences of form and/or thinking rather than content. For example: Version 1 of a T-section (T_1) may require extensive reading, whereas version 2 of the T-section (T_2) for the same module may be high pictorial or contain smaller chunks of information per display.

An author must write a module so that it instructs to one objective as that objective is represented in the prerequisite structure (Figure 8). That is, if a module (objective) does not have a linearly prerequisite module, the module being written must be independent of other modules. For example, in Division A, as represented in Figure 8, notice that modules F, G and H must each be written in a manner such that the author of any of these modules assumes only mastery of the objective taught by Module E when the student is taking Modules F *or* G *or* H. The author of Module I, however, can assume that the student has mastery on the objectives taught in Modules F *and* G *and* H. The prerequisite structure shown was established for the COBOL2 course during its behavioral design phase and is based on an analysis of the relative nestings of behavioral objectives within others.

The implications of this design for the way a course author must create his material is evident in Figure 9, which is a partial diagram of the path structure through Division B

Course Structure—Divisions, Modules, and Sections



A = Administrative Section

T = Telling Section

P = Practice Section

Q = Quiz Section

Adapted from Felix F. Kopstein and the Instuctional Programming Staff of Project IMPACT, "COBOL Programming. Readers' guide to the Instructional Content for Computer Administration," HumRRO Research Product, June 1972.

Figure 7

COBOL2 Prerequisite Structure

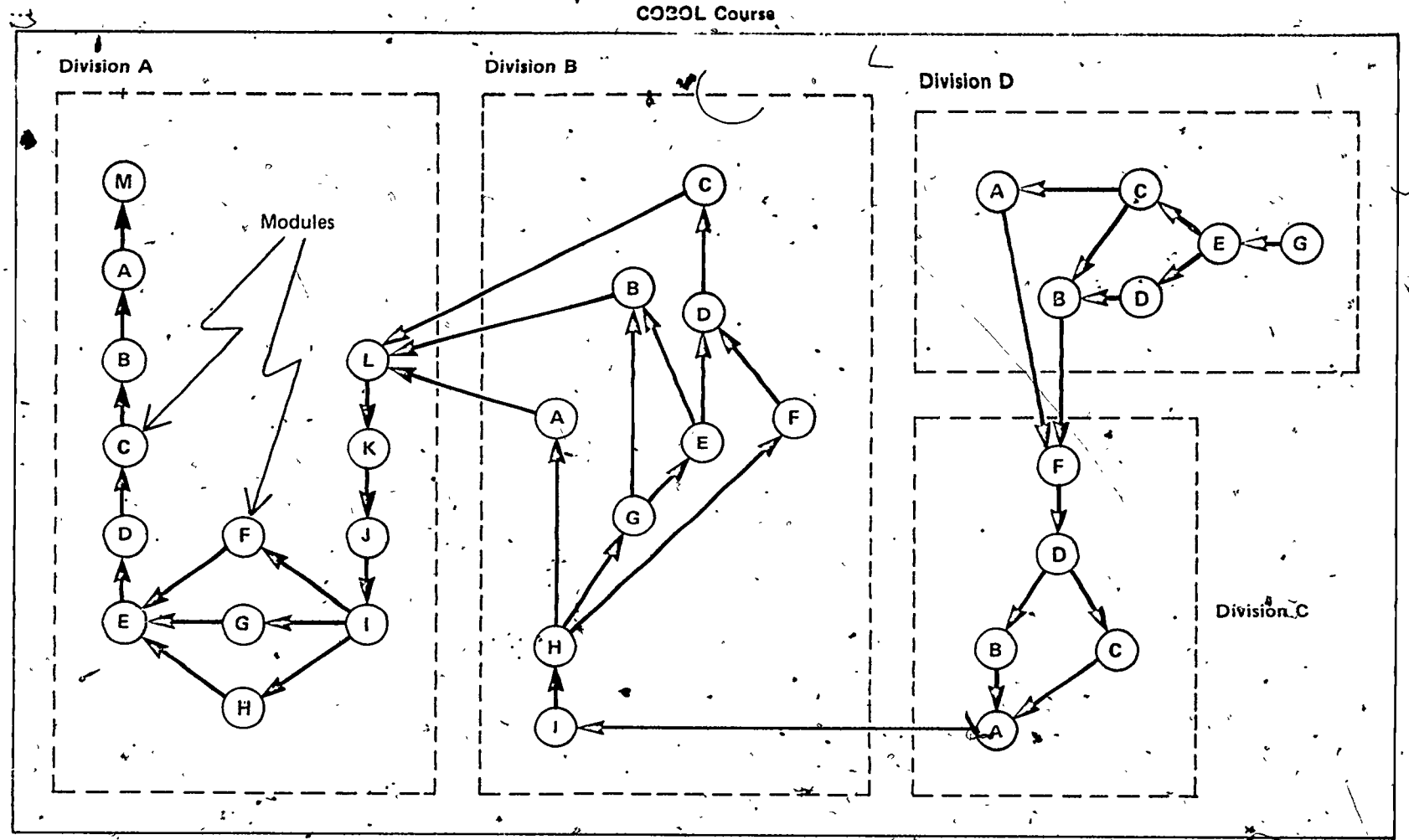
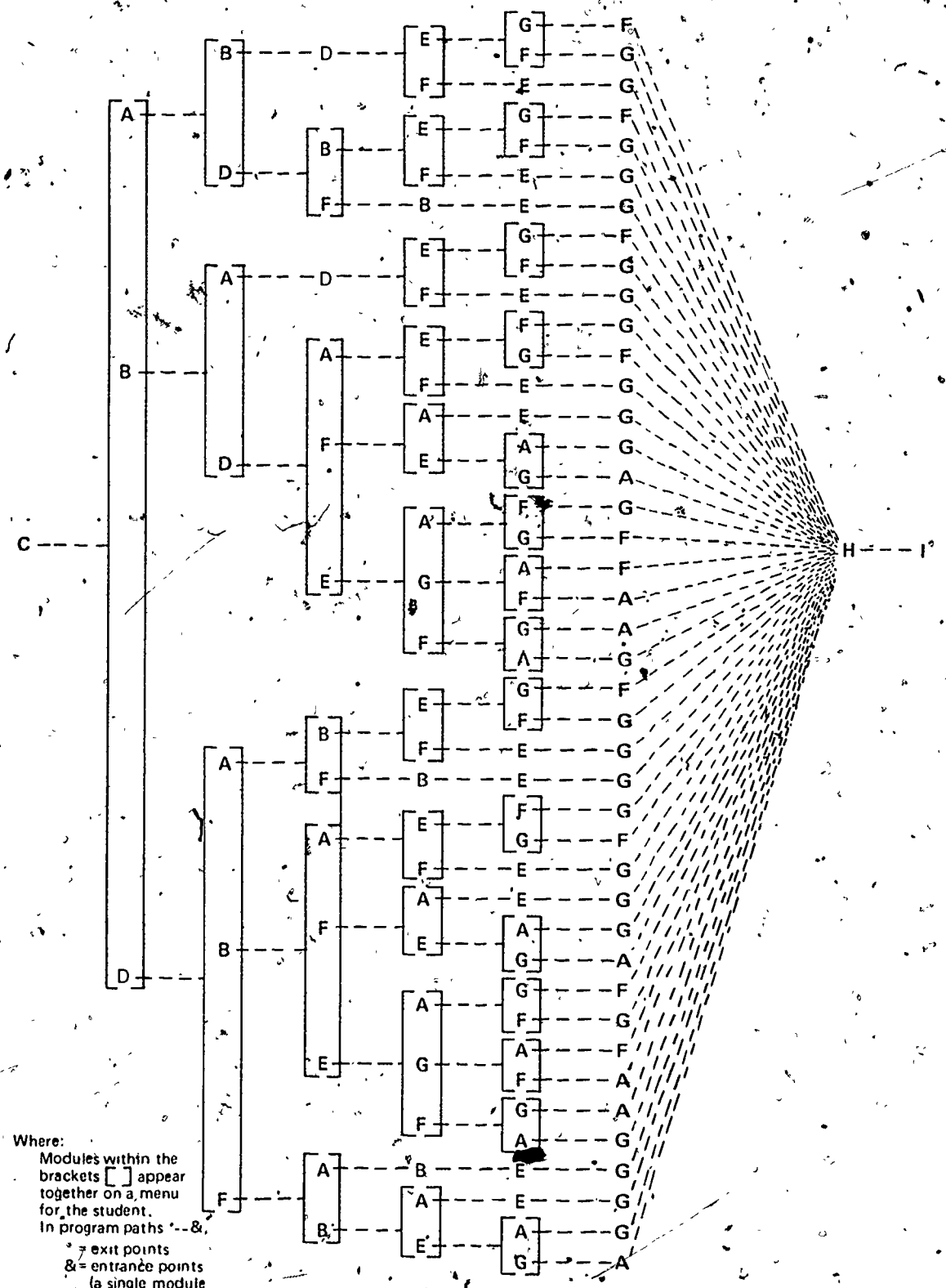


Figure 8

Partial Path Diagram for Division B



Where:
Modules within the brackets [ ] appear together on a menu for the student.
In program paths '--&'
' = exit points
& = entrance points (a single module or a menu).

Figure 9

(if the student chooses Module C at his first choice point). Keeping in mind that the COBOL2 course is a self-contained tutorial (no non-system instructional support) the author must create the instruction so that all students, no matter what path they took to get to the module, can comprehend and thereby master any module in the Division.

This structure was an outgrowth of the IDM design permitting learner control over his path through the subject matter. The other learner control features of the IDM had different effects on the authoring of modules. T-sections are written so that general concepts are introduced first and, as the student goes on, the specifics are explained in detail. This technique enables the student to discern, as early as possible, whether he thinks he has sufficient mastery to skip to either the P- or Q-section. It also enables the student who is in REVIEW or RECAP mode to find the information he needs as early in the module as possible.

The structure of a P-section should resemble that of the T-Section in that exercises dealing with the general concepts should be first, followed by questions on specific enabling objectives (for the module). The P-section provides drill and practice, and detailed explanation (in the form of associative displays) to complement and expand on the information presented in the T-section, and should also provide content and format preparation for the Quiz.

The Q-section must be a discrete, self-contained section including explicit directions to the student on the mechanics of answering the question(s) that test the module objective. It is important that the quiz be an accurate test of the objective and not be affected by the idiosyncracies of question presentation or methods of student response.

Authors found the course design and structure to be beneficial during the creation of materials, because it imposed enough constraints on the instruction to make different modules, written by different authors, compatible in form and general order of content presentation. It allowed, however, much individual freedom to authors in the creation of within-module strategies. This is exemplified by the fact that the modules which used gaming as an instructional method met the IDM and course structure requirements, as did the more conventional instructional techniques employed in some of the other modules.

AUTHOR SUPPORT

CAI as developed by the project (IMPACT and GRANT GJ-774) was characterized by separate storage and retrieval of instructional content and instructional logic. This innovative technique, proven successful under operational conditions, gave rise to unique on-line authoring aids. One of these, being able to retrieve and display a piece of text throughout the course while storing only a single copy in the instructional text files, resulted in very economical use of file space. Economy in the form of increased cost-effectiveness was the result of allowing several people to work on the same piece of instruction (module, division) simultaneously, thereby amortizing computer costs over more users and reducing total time on-line needed to create a given segment of instruction.

Part of the time-sharing monitor and data-base management executive subsystem, called ZEUS, was a simple author language called EDITOR. EDITOR could be mastered quickly since the author need know only five commands to manipulate the text files adroitly. (See Section 5, Software Support). These five commands are CREATE, MODIFY, DELETE, COPY, and DISPLAY. Their meanings are self-explanatory. All an author need do is enter a command and an object file location identifier, and the next thing he sees is the requested file element. In all cases except DISPLAY, the element is automatically updated, according to the command issued, after the author signals that he is finished with the element.

The separation of text and logic; in combination with EDITOR, made possible the development of an even more sophisticated and powerful authoring aid—SUPEREDITOR. The role of SUPEREDITOR in the courseware subsystem is to facilitate the manipulation of the text files for naive, as well as sophisticated, authors and editors when either authoring or editing is done interactively, on-line. To use SUPEREDITOR an author need only know the operational definitions of the SUPEREDITOR modes as shown in Table 8. Unlike EDITOR, SUPEREDITOR allows the author to work on a string of disk elements and he has some new capabilities which make string manipulation even easier. For example, PRE-FORMAT mode allows the author to select an existing file element (or create a new one) to function as a template. Thereby the new element being created already has a structure when it appears on-line and the author need only fill in the content. This feature is especially useful for standardized pages such as documentation pages.

The bulk of the course-specific documentation is resident on the instructional text files. For the most part, the student does not see this documentation, but it can be accessed by course authors for modification and/or viewing in the same manner as any other element. It is not always desirable, in terms of time and cost-effectiveness, to work with and inspect either instructional or documentation items on-line. Therefore, a hard-copy, off-line reproduction of the instructional text files was implemented via the File Activity Control System (FACS). (See Section 5, Software, and Appendix F.)

Table 8

SUPEREDITOR Modes

Symbol	Mode	Operation
0	ANTIDOTE	Retrieves a deleted element if no other operation has intervened between DELETE and ANTIDOTE.
1	DELETE	Inactivates a disk location.
2	DISPLAY	Retrieves the contents of a disk location on the CRT screen for viewing, but does not permit any permanent alteration of the element.
3	CREATE	Activates a disk location and permits the user to store text on the element.
4	MODIFY	Accesses an active disk location and permits the user to change the content and/or structure of the resident element.
5	PRE FORMAT	Copies a previously stored template into a newly activated disk location and permits the user to input content into the structure.
6	DISK DISPLAY	Retrieves the disk location number of an element being viewed via DISPLAY and places it in the upper left corner of the element.
7	EDITOR	Allows the user to write his own Editor commands, two at a time. Adds these to the Editor page.
8	SIGN OFF	Allows the user to exit from SUPRED
9	DISK TITLES	Executes the list command which shows the label of all specified active disk locations and the name of the last person who worked on each.
A	JOB ENTRY	Allows the user to retrieve and execute any program in the D-file.
B	TITLE MODIFY	Allows the user to make changes in the label of a disk location and to change the content and/or structure of the element, if desired.

Separate TITLE listings are also produced by FACS. It contains such information as the entry's location on the text file, its identifier, whether it is seen by the student, and what type of student response it calls for. An author can follow the flow of a module just by perusing the TITLE listing. This check also insures that the module flow does not violate the IDM rules for module structure.

In order for the IDM to make decisions based on quiz scores, the scores have to be standardized, but different criterion tasks are not always amenable to similar scoring. Therefore, a special computation element, Figure 10, was developed to help authors assign raw scores to quiz questions/tasks, and then transform them to standardized IDM scores. The FACS copy of the computation element is a useful tool for the author to have when he is debugging a quiz.

During course administration and debug, the FACS copies of the standardized documentation pages (especially the A-section) enable any author or administrator to understand what should happen during the instruction, even if he is unfamiliar with the module as a whole. All authors must provide documentation with any module they create, and this practice has proved to be beneficial both when the material is being prepared for on-line operation and after it is operational.

Overall, the author support facilities have enabled authors with little or no programming skill to create and put on-line effective textual materials for CAI.

COMPONENTS OF THE COURSEWARE SUBSYSTEM

The set of instructional materials for computerized training by this project consists of the following products:

- Text[1]
- Auxiliary Visuals[2]
- Glossary[1]
- Student Reference Manual

The text was developed for on-line, interactive application in a total tutorial CAI mode. The text files contain both the instructional materials for student viewing and on-line documentation.

Two main courses were developed. The Introduction to CAI Course (INTRO) is used to introduce the student to the features of the CAI environment. It is independent of any subsequent substantive course and serves only to familiarize the student with the parameters and mechanics of the CAI system in which he will learn. There are 13 modules in the INTRO course, and it takes the students from two to five hours to complete all of them. An alternate version of INTRO was developed which replaces nine of the modules with an off-line lecture and retains on-line only the modules covering keyboard practice and IDM features. Completion time for this version is a maximum of three hours.

The COBOL2 course teaches the student some of the basic concepts of data processing using the COmmon Business Oriented Language (COBOL) as a vehicle. The course consists of four divisions which contain 33 modules with an estimated average completion time of 45 hours. By "cutting and pasting" the FACS representation of the on-line text, a CMI (quizzes and prescription remained on-line) version of portions of the COBOL2 course was developed. The CMI students did not do quite as well as their CAI counterparts in terms of transit time and performance but the results were still acceptable. The differences could be explained by realizing that little attempt was made to adapt the now off-line materials for the new media.

[1] Available in microfiche form.
[2] Available in either 35mm or 16mm.

Standard Page for Quiz Score Computations

```
--------1---------2---------3---------4---------5---------6---------7---------8----

DIV: B   MOD: A  VER: 1   OBJ: 1   DATE: 9/22/71  INIT:FFK

OBJ DES:  CODE AN IDENTIFICATION DIVISION

SCORABLE EVENTS:

A.  CORRECT A-B MARGIN          D.  VALID P.S.C.N.        G.

B.  CORRECT PUNCTUATION         E.  CORRECT LINE NO.      H.

C.  CORRECT CRW                 F.  CORRECT SENTENCE      I.

EVENTS      A   B   C   D   E   F   G   H   I   MAX   Q-SCORE    IDM

POINTS      1   1   1   3   2   1               PTS      0         0
                                                        1-5       1
C  BA110A    1   1   2       1   1                7       6-10      2
   BA110B    1   2   1   1   1                    9      11-14      3
W  BA110C    1   1   1   1                        6      15-17      4
   BA110D    1   1   1  -3   4                    3      18-20      5
I                                                       21         6
                                                        22         7
P                                                       23         8
                                                        24         9
S                                                       25        10

TOTALS      4   5   5   3   2   1               25

OTHER INSTR:  SEE BA1018

--------1---------2---------3---------4---------5---------6---------7---------8----
```

Figure 10

Because of the lack of an efficient graphics capability (simple graphics could be done using alphanumeric and special characters) on the CRT, an auxiliary visual device was included in the student terminal configuration. This device was a random-access, 16mm projector which is entirely under computer control (author specified). The device was used when color, graphics, animation, or illustration was required, or if the CRT was already in use and the author wanted to present additional or supplemental text, or if an author wanted a dynamic pictorial presentation of the subject matter rather than a dynamic semantic presentation (via the CRT).

The glossary or inquiry capability was presented to the student as an IDM feature. For the most part the student used it as he would a dictionary, but in the "debugging" modules he inquired as to the meaning of various COBOL diagnostic codes for program error messages, using the same technique. Terms with their definitions were input into the glossary file by one of two methods: (a) an author, when creating material, specified any terms that should be included; (b) if three different students requested a term, its definition was input.

The student manual was organized like a standard reference book, rather than by course structure. This was done to familiarize the student with using a programmer's manual, since programming normally requires considerable facility in the use of reference manuals. Since the student has the manual with him almost all the time, the author must be careful that he does not negate quiz results either in a current module or a future one, by putting too much course-specific information in the manual.

The courseware components then represented a diverse set of media and capabilities available to an author, so that each could be employed optimally in the CAI-learning environment.

PRODUCTION PROCESS

The approach taken to the development of instructional materials has been to use a multidisciplinary team to design, develop, and implement course materials. The team capabilities include instructional design, subject-matter expertise, instructional programming, editorial skills, hardware/software expertise, media selection and utilization, and production and management skills.

The instructional designer establishes behavioral requirements, encodes them into instructional objectives, and designs the IDM that will best serve the needs of the students and the subject matter. The subject-matter expert provides "raw" content to the instructional designer and instructional programmer and serves as a consultant when the programmer structures the content for implementation in a CAI mode. The instructional programmer is a specialist in using computers effectively and efficiently for instruction.

The editor reviews the draft instruction, which is usually delivered by the instructional programmer, on a paper representation of the CRT screen containing all the information for a particular element—text, characteristics (to be entered in the title listings); and so forth. The editor is examining the material for accuracy, consistency (inter- and intra-modular), and style. Auxiliary material (visuals, manual, glossary) is reviewed along with the future on-line materials, at this time.

The material next goes to the technicians who input the instructional text and program the instructional logic.

After the material is implemented and before students are allowed to take it, the course must go through a debugging phase. Material moves down the production chain modularly and the first debug is done a module at a time. The debug includes on-line checks for proper linkage among text elements, samplings of all possible answers to questions to insure proper analysis and branching, verification of quiz scoring using *all*

possible permutations of correct and partially correct answers, and an off-line check, using FACS output, for textual errors (spelling, syntax, etc.). After this step is satisfactorily completed, several naive students go through the material to further "shake down" any bugs.

Only when the *entire* production process, from design to on-line test and debugging, is satisfactorily completed is the course deemed ready for students to see in a full operational mode.

OPERATION AND ADMINISTRATION

After the INTRO and COBOL2 were thoroughly debugged, students were permitted to take the courses. Before the students went on-line, they had to take the Entry Characteristics Test (ECT) battery. At this time they were given an overview of the project, told the function of the ECT, and given an explanation of what they could expect to gain from taking the COBOL course. ECT lasted for three half days and screening took place after the first session. COBOL2 was written for students who exhibited a minimum aptitude (equivalent to a low C grade) on the programmer aptitude test, the ATPP, and this was the test upon which students were screened.

To take the course proper, as part of the first instructional session the student keypunches some data cards and is taken to the computer center to see the program run, using the cards he has just punched. The student then enters his carrel and starts the INTRO course. From here on, the students are scheduled for sessions of three hours a day, five days a week until they are finished with the course. They arrive in the beginning of a session, pick up their folders containing supplies from the proctor, and spend the remainder of the session in their carrels. If they perceive problems during the instruction, they call the proctor to their carrel.

The proctor is part of the administrative staff responsible for the operation of the course. The members of the administrative staff are:

The *Director for Daily Operations*, who is responsible for efficient day-to-day "running" of the course. He serves as liaison to the directors for other components of the total system. It is his responsibility to insure that required system (any component) changes are accomplished in such a manner that student progress is minimally affected. In effect, he is a traffic manager for problem solutions and computer utilization while students are on-line. His duties also include supervision of the proctoring staff.

The *Operations Monitor* (OM), who is responsible for the smooth running of a particular student period/shift and supervising the proctors on that shift. He serves as a resource for the proctors and is the only person assigned to the shift who can give the student relevant subject-matter information.

The *proctor*, whose function is purely procedural. The students should be informed of this. The proctor is responsible for insuring that the student has all required materials, correcting minor hardware difficulties (auxiliary visual display does not align properly with primary display, etc.), answering system-generated proctor calls, diagnosing a student's problem, and, if it is subject-matter specific, referring the problem to the OM.

Most proctor calls are student initiated, and the proctor goes to the student, diagnoses the student's problem, and effects a solution. In our tutorial CAI application it is crucial that the course administrator and/or evaluator be able to determine what transpired during the proctor-student interaction. Therefore, a proctor trouble report (TR) was designed and developed to document all proctor calls, both student and system generated (Figure 11). The TR was part of the VOYEUR course. VOYEUR was created as a course, although it was purely for administrative purposes, so that the

VOYEUR Element for Trouble Reports

CWIP... STUDENT NO. ... CUBICLE.. PROC. ... OM ... ID. 1008 DATE 9 JAN 1973 TR.14
TIME: 18:57

TROUBLE ..

IP	CW	XX IN QUIZ HRDWRE	MISC
A1-TYPO CRT	G1-RESPONSE ANAL.	M1-LOST CUE COUNT	S1-LOCKED KBD
A2-TYPO P-SCOPE	G2-CURSOR POS. NG	M2- P-SCOPE NG	S2- P-S OFF CENTER
A3-TYPO MANUAL	G3-CRT SCRAMBLED	M3-SYSTEM CRASH	S3-MULT. CURSORS
A4-CRT TEST NG	G4- P-SCOPE CUE NG	M4-LIGHT PEN NG	S4-MISSING MATLS.
A5-PSCOPE TEST NG	G5-WRONG BRANCH	M5-CRT NG	S5-IDM PROBLEM
A6-MANUAL TEST NG	G6-CRT BLANKED	M6-LINE TROUBLE	S6-S ERROR
A7-DIRECTIONS UNCLR	G7-WRONG FDBK	M7-BULB BURN-OUT	S7-CMI CUB BUSY
A8-SUBJ MATTER ?	G8-WRONG POINTS	M8-VIEWLEX NG	S8-IP INTERNAL
A9-XX IN P SECTION	G9-TP I/O ERROR		
B1-OPTION PROB.	H1-EXCESS PROC.		
B2-GLOSSARY PROB			
B3-PROGRAM ?			

50-OTHER.....

Figure 11

proctor-generated records were compatible with all student records, so as to allow standardized manipulation and analysis.

Once a day the data in the TR file are reproduced in hard copy and are disseminated to an appropriate referee who takes action to eliminate the cause that generated the proctor call. If the referee feels that the problem was unique and does not have evidence that it could reoccur for other students, he need not take any action. The TR record is updated after referee action takes place. At the end of a student group, all non-updated records are identified and the Director for Daily Operations insures that some referee action is taken. These procedures provide for the quality control of the instruction while in an operational mode.

During course administration the off-line author support facilities (FACS, TITLE listings, etc.) are used for administrator support. Because of the extensive documentation in the FACS, proctors and OMs can accurately analyze and solve most student problems without referring back to the original author or having to know how to read the instructional logic printouts.

Course operation functions as if it were in an applied operational setting. It is an independent system administered by para-professionals which would, in a school setting for example, free the professional subject-matter expert from the relatively menial tasks associated with course administration.

REVISION PROCESS

The COBOL2 course was evaluated for revision requirements after each group of students completed the course. The revision and review process drew upon computer-generated data management reports (daily and module summaries, Table 9), TRs, and the expertise of behavioral scientists, instructional technologists, subject matter experts, and software support staff.

A systematic methodology for diagnosing revision requirements was developed. It is being written up now as part of a paper on formative evaluation. When a "repair" has to be made, the new material goes back through the production process, entering at an appropriate point and going through a debug phase before it becomes part of the course. Application of this algorithm led to no modules being rejected by the time the experimental subjects took the COBOL2 course.

Table 9

Summary of Quality Control Programs for CAI Course

Program	Output
Acceptance/Rejection	Flags substandard instruction units by statistical criteria, as soon as possible, so revision can be initiated.
Student attendance	Reports attendance record (daily and cumulating), time of signon and signoff for the day per student, furthest place in course, and cumulative time spent on course.
Student comments	Reports student opinion ratings and all comments referenced by the student number and the place where it occurred.
Student Glossary Request Found	Reports word requested, student number, location at time of request, data and time request was made.
Student Glossary Request Not Found	Same as above.
Programming Errors	Questions that were incorrectly graded.
Programming Errors II	Records inconsistencies in data compiled for specific items.
Performance Summary (one table/objective)	Lists raw and transformed quiz score for each student and for each attempt.
Summary Student Response Matrix (1 matrix/instructional unit)	Each row of the matrix represents a question. Each column represents a student. The cells of the matrix indicate correct answers, incorrect answers, and skipped questions. Marginal percentages show the percent correct by student and by question.
Student Opinions	Prints a value from 0 to 9 for each student.
Student Options	Indicates number and type of student control options exercised by each student, and where they were invoked.
Student comments	Verbatim, with student and course location identifier.
Student Response Listing	For each question (except quizzes) lists actual response typed by student, as well as certain counters and switches.
Answers for Quiz Attempts	Lists quiz responses by student. Distinguishes first, second, and subsequent attempts.
Most Frequently Incorrect Answers	Lists the five questions in the instructional unit with the highest frequency of incorrect answers. For each question, it prints: number of attempts; number of incorrect answers; number of attempts with response greater than 10 minutes; number of unanticipated responses; number of answers with spelling margin, punctuation, or subject matter specific errors; and the cumulative time in seconds spent answering the question.
Most Frequently Correct Answers	Provides same information as above.
Questions with Longest Cumulative Response Time	Same as above.

Section 5

SOFTWARE SUPPORT

Software research and development has resulted in the development of a comprehensive, operational software subsystem that performs all software tasks required in a computer-administered instruction environment. Appendix G provides an overview of the software subsystem.

The software subsystem can be divided into real-time and off-line components. Real-time subsystem components include the following.

OS/3700	IBM 370 Operating System
Zeus	The on-line monitor developed at HumRRO used to interface student terminals with Coursewriter. Zeus includes three distinct subsystems.
	Editor: for interactive authoring development.
	Director: for locating and retrieving disk stored information.
	RJE: for inputting, initiating and executing jobs remotely.
Coursewriter III	IBM's Coursewriter CAI software subsystem with two important extensions.
	Interface: maintains on-line updated description of student attributes, course component prerequisites, and in general controls all intermodule branching as well as some intramodule branching.
	Functions: specially developed functions to extend Coursewriter response analysis, data recording, branching, etc., capabilities.

Figure 12 summarizes this on-line software subsystem and also depicts terminals, random access storage, and printed output.

Of particular importance in this subsystem is the Interface component. The Interface assumes that a CAI course is structured in a particular manner. A course is composed of one or more divisions with each division having one or more modules. A module contains a Telling (T), a Practice (P), and a Question (Q) section. One of the primary features of the Interface is that it permits an author to design an individual course for *each* student with respect to division and module definitions and structure. That is, each student can have his own course structured for him by an author, using the pool of available modules and module components (T, P, Q sections). Interface permits the author to establish individual module prerequisites for each student and controls intermodule and intramodule transfers for each student. Appendix H provides more details concerning Interface logic, operation, and use.

Real-Time Software Components

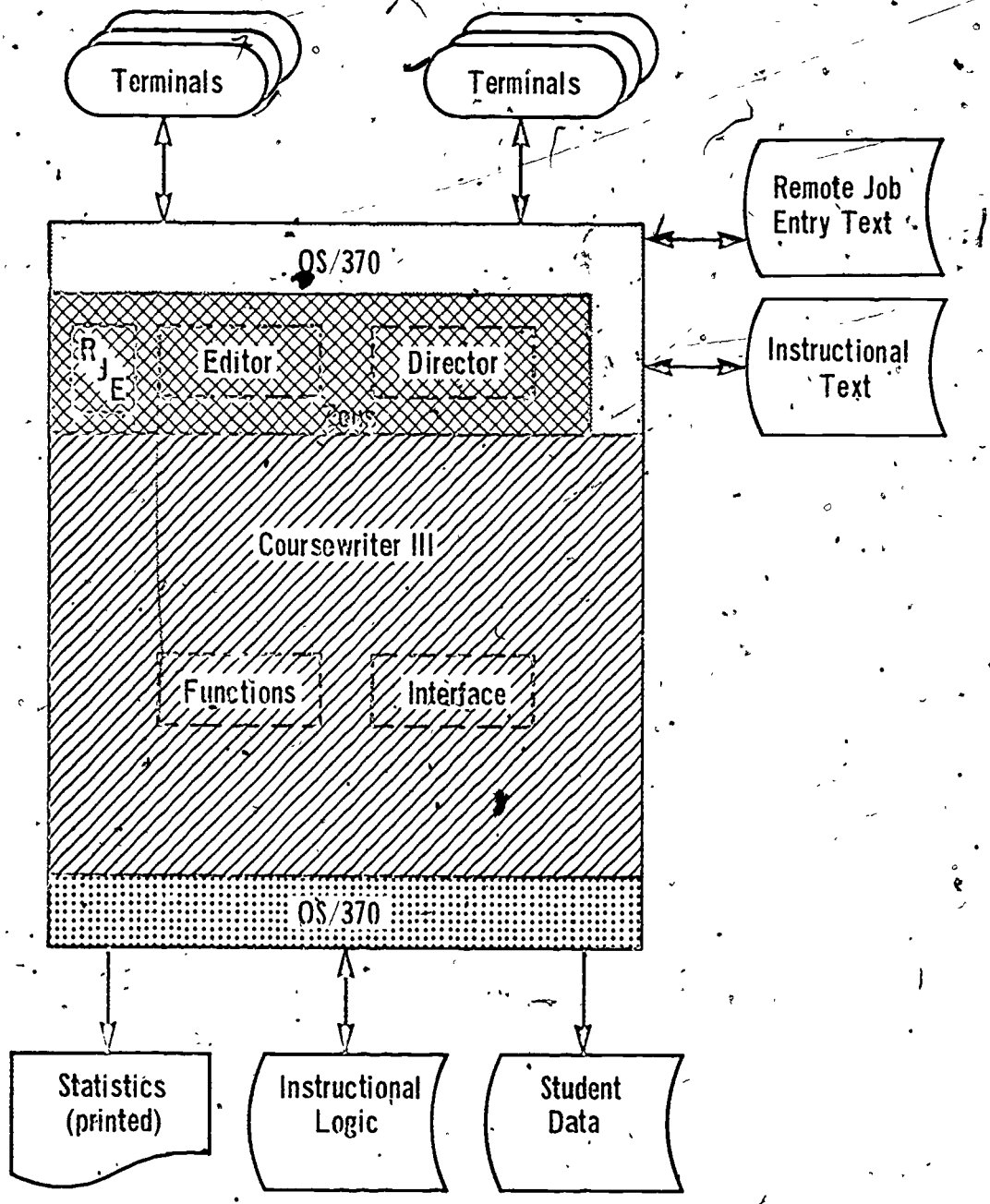


Figure 12

From this brief description, it can be seen that authoring activity consists of two distinct operations. On the one hand, a pool of T, P, Q sections must be developed for student use. Given the pool of T, P, Q sections, the author must next structure them into a course for *each* student individually. This entails specifying the individual's course divisions, division modules, and each module's structure. Incidentally, the author can specify a different module structure with, if desired, different T, P, and Q sections for successive encounters with the module by the student. In addition the author specifies through Interface prerequisites for the individual student for individual modules.

Interface assembles these data into an on-line record used to control inter- and intra-module transfers. Specifically a student can exercise several Interface options including the following:

REVIEW	Restart the current module at the beginning.
PRACTICE	Move forward in the present module to the P section.
QUIZ	Move forward in the present module to the Q section.
RECAP	Recapitulate a module already completed (System presents a menu of completed modules for selection).

The Interface also schedules remediations and presents appropriate options after a module is completed (end of Q section). Thus, in on-line operations with the student, the Interface is in complete control of branching and has an updated description of student attributes.

Coursewriter extensions have been affected through the development of 12 Coursewriter functions. The functions provide the following capabilities.

(1) Selective clearing of Coursewriter III counters.
(2) Selective clearing of Coursewriter III switches.
(3) Determining the length of messages in any Coursewriter III buffer.
(4) Performing complex arithmetical operations.
(5) Scanning a buffer and transferring its numeric content to a counter.
(6) Selective loading of multiple counters from buffers or auxiliary storage.
(7) Selective placement of an EOB in a buffer.
(8) Recording student data when required.
(9) Removing specified character strings from a buffer.
(10) Determining frequencies of occurrences of specified character strings in student responses.
(11) Selective storing of multiple Coursewriter III counters in buffers or auxiliary storage.
(12) Scanning student responses and recording data concerning items in the response relative to the COBOL language requirements regarding programmer defined names, reserved words, and literals (syntax).

In particular, a sophisticated extension of Coursewriter has been achieved that permits the analysis of complex student responses. This advance was essential for the analysis of student input of COBOL syntactic structures. Appendix I provides the details concerning these Coursewriter extensions.

Off-line software components include the following:

IDES	A data storage, retrieval and analysis subsystem with two components.
	BMD: Biomedical Statistical analysis package.
	Storage/Retrieval: The data storage/retrieval component.
FACS	File Activity Control System—an author aid that provides management control of and reports on authoring activity.

These off-line components are depicted in Figure 13 which also depicts data input, program card input, random access storage and report printing.

Of particular importance is the storage/retrieval aspect of IDES. Initially (IDES-1) IDES structured data in the form of lists. That is, list processing techniques were used to store and retrieve data. This was accomplished through an extended SLIP processor. Subsequent research indicated that more standard data processing techniques could be used to perform these functions. This led to the development of a more conventional storage and retrieval version of IDES (IDES-2). This improved version of IDES is PO/1 based.

Thus, IDES has evolved into a comprehensive system capable of storing and retrieving student generated data. Coupled with the BMD, this allows for selective analysis of data in an off-line environment for research and modeling purposes. Details of IDES-1 are given in Appendix J and details of IDES-2 are given in Appendix K.

Off-Line Software Components

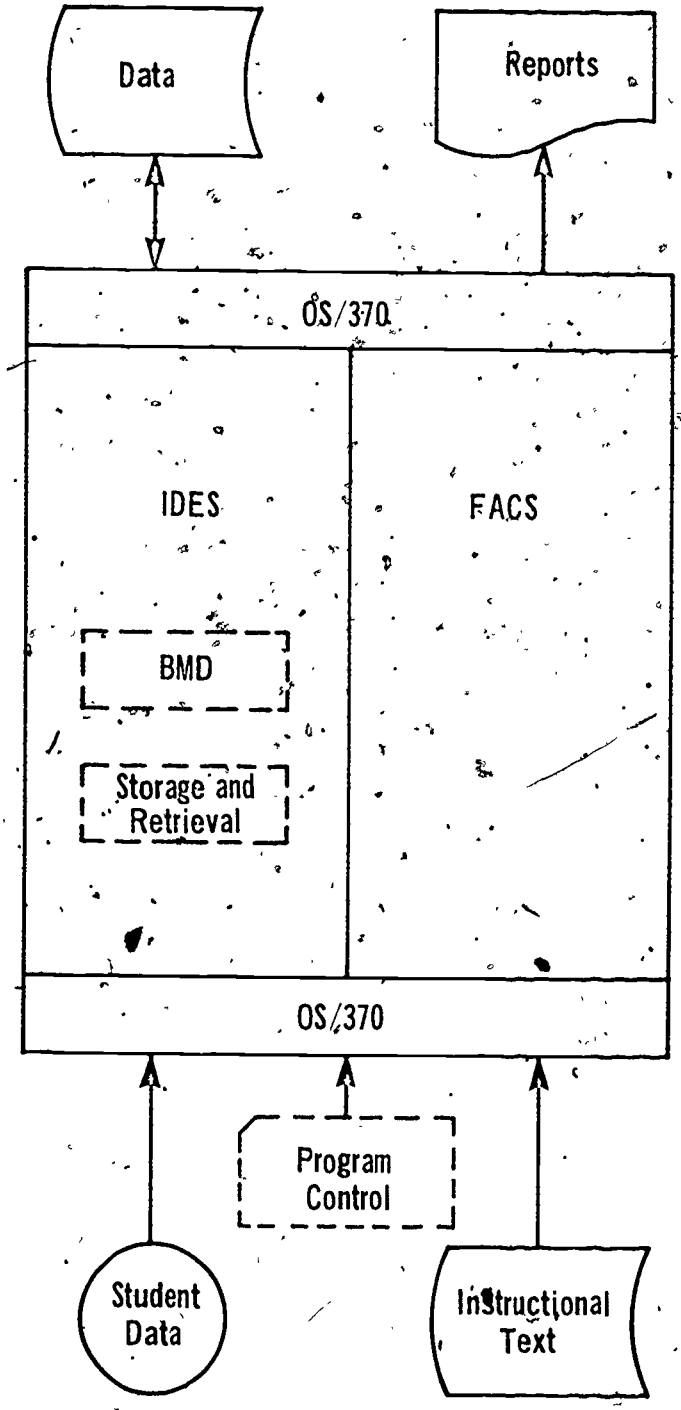


Figure 13

Section 6

PUBLICATIONS

SOFTWARE DOCUMENTATION SERIES

Overview of the Computer-Administered Instruction System, by John Stelzer and Jean Garneau, Technical Report 72-21, August 1972.

II. *The IMPACT Data Evaluation System—Version 2 (IDES-2)*, by Leslie Willis and John Stelzer, Research Product RP-D1-72-1, August, 1972.

III. *The IMPACT Data Evaluation System—Version 1 (IDES-1)*, by John Stelzer and Leslie Willis, Research Product RP-D1-72-2, August, 1972.

IV. *The Interface Subsystem Framework for Instructional Decision Modeling*, by William Underhill and John Stelzer, Research Product RP-D1-72-3, August 1972.

V. *File Activity Control System (FACS)*, by Leslie Willis, Jean Garneau and John Stelzer, Research Product RP-51-72-4, August 1972.

VI. *Volume 1, Zeus Functions and Design Concepts*, by Jean Garneau and John Stelzer and *Volume 2, Zeus Program Logic Descriptions*, by Jean Garneau, William Underhill and Doris Shuford, both Research Product RP-D1-75-5, August 1972.

VII. *IMPACT's Computer-Administered Instruction Software Subsystem, Coursewriter III, and Its Functions*, by Doris Shuford and John Stelzer, Research Product RP-D1-72-6, August 1972.

VIII. *Computer-Administered Instruction Computer Program Logic for COBOL2 Course of Instruction*, by Douglas Spencer, Elizabeth Sowell, Leslie Willis and Jean Garneau, Research Product RP-D1-72-7, August 1972.

Project IMPACT—Computer-Administered Instruction: Functions for the Coursewriter III Language, Research By-Product, RBP-D1-71-2, by Project IMPACT Staff, June 1971.

PROFESSIONAL PAPERS

Course Modularization Applied: The Interface System and Its Implications for Sequence Control and Data Analysis, by E. W. Schneider, Professional Paper 10-73, November 1973 (Also presented at the meeting of the Association of Instructional Systems (ADIS), Chicago, Illinois, April 1972.

Who Should Develop Instructional Materials for CAI? by Robert J. Seidel, Professional Paper 20-71, October 1971. (Also presented at the Computers in Instruction Conference, Los Angeles, California, October 1970.)

Theories and Strategies Related to Measurement in Individualized Instruction, by Robert J. Seidel, Professional Paper 2-71, March 1971. (Also presented at the American Psychological Association Convention, Miami Beach, Florida, September 1970.) Published in *Educational Technology*, September 1971.

MISCELLANEOUS PRESENTATIONS

"Towards Understanding the Value of Learner Controlled Instructional Sequencing," presentation by Robert J. Seidel at Association for The Development of Computer-Based Instructional Systems (ADCIS) August, 1973, Ann Arbor, Michigan.

TECHNICAL REPORTS

"Project IMPACT Courseware Subsystem: Vol. 1, Innovative Procedures for Development and Administration," by Michael J. Hillelsohn, Technical Report in preparation.

"An Axiomatic Theory of Subject Matter Structure," by Edward Kingsley and John Stelzer, Technical Report in preparation.

REFERENCES

Brennan, R.L. *Some Statistical Problems in the Evaluation of Self-Instructional Programs*, Dissertation, Harvard University; University Microfilms, Ann Arbor, Michigan, No. 70-23080, 1970.

Bunderson, C.V. *Transfer of Mental Abilities at Different Stages of Practice in the Solution of Concept Problems*, Research Bulletin (RB-67-20), Educational Testing Service, Princeton, New Jersey, 1967.

Carnap, R. *The Continuum of Inductive Methods*, University of Chicago Press, Chicago, Illinois, 1952.

Carnap, R. *The Logical Foundation of Probability*, (2d ed.) University of Chicago Press, Chicago, Illinois, 1963.

Dunham, J.L., and Bunderson, C.V. "The Effect of Rule Instruction Upon the Relationship of Cognitive Abilities to Performance in Multiple-Category Concept Learning Problems," *Journal of Educational Psychology*, vol. 60, 1968, pp. 121-125.

Dunham, J.L., Guilford, J.P., and Hoepfner, R. "Multivariate Approaches to Discovering the Intellectual Components of Concept Learning," *Psychological Review*, vol. 75, 1968, pp. 206-221.

French, J.W., Ekstrom, R.B., and Price, L.A. *Kit of Reference Tests for Cognitive Factors*, Educational Testing Service, Princeton, New Jersey, 1963.

Gagne, R.M. *Conditions of Human Learning*, Holt, Rinehart and Winston, New York, 1965.

Gagne, R.M. "Context, Isolation, and Interference Effects on the Retention of Fact," *Journal of Educational Psychology*, vol. 60, no. 5, 1969, pp. 408-414.

Gagne, R.M. "Instruction Based on Research in Learning," *Engineering Education*, March 1971.

Guilford, J.P. *The Nature of Human Intelligence*, McGraw-Hill, New York, 1967.

Guilford, J.P., and Hoepfner, R. "Structure-of-Intellect Tests and Factors," *Reports from the Psychological Laboratory*, No. 36, University of Southern California, Los Angeles, 1966.

Hansen, D.N., and Papay, J.P. *Forgetting as a Function of Forget Cue, Sentence Categorization and State Anxiety*, Florida State University, 1970.

Mandler, G., and Sarason, S.B. "A Study of Anxiety and Learning," *Journal of Abnormal and Social Psychology*, vol. 44, 1952, pp. 166-173.

Mukherjee, Bishwa Nath. *Development of a Forced-Choice Test of Achievement Motivation*, Report Number CRP-S-113, Indiana University, Bloomington, Indiana, August 1964.

Pask, Gordon. "Computer Assisted Learning and Teaching," Proceedings of a Seminar on Computer-Based Learning Systems, National Council for Educational Technology, Leeds University, Councils and Education Press, London, September 1969.

Perry, D.K., and Cantley, G. *Computer Programmer Selection and Training in System Development Corporation*, Tech. Memo. 2234, System Development Corporation, Santa Monica, California, 1965.

Rowan, T.C. "Psychological Tests and Selection of Computer Programmers," *Journal of the Association for Computing Machinery*, vol. IX, 1957.

Shuford, E.H., Jr., Albert, Arthur, and Massengill, H.E. "Admissible Probability Measurement Procedures," *Psychometrika*, vol. 31, no. 2, June 1966.

Spence, K.W., and Spence, Janet Taylor (eds.). *The Psychology of Learning and Motivation*, vol. 1, Academic Press, New York, 1967.

Appendix A

AN AXIOMATIC THEORY OF SUBJECT MATTER STRUCTURE

HumRRO Technical Report, by Edward H. Kingsley and John Stelzer (in press)

Summary

An adequate theory of instruction must include as components at least the following:

A representation of the subject matter
A representation of the instructional goal
A representation of the initial and current student states
A representation of the instructional strategies

This paper focuses on the first of these components: representation of subject matter. It is the hope of the authors that the rigorous descriptive theory that is developed will serve as a foundation on which can be built theoretical extensions to include the remaining listed components.

It must be stressed that at present the theory is purely descriptive. That is, it is not at present possible to derive very many prescriptions to be used in controlling instructional branching, structuring material, etc. These results hopefully will be achievable when the descriptive theory is more complete. Regardless, the development of the descriptive theory is recognized as the first step in an ultimately pragmatic and applicable theory. The descriptive theory itself will result in a significant step forward with regard to clarity and precision in terms of conceptualizing about the instructional process.

The theory as presented, in contradistinction to most theories for structuring subject matter, divides subject matter into two distinct categories: Content and Tasks. Assumptions are introduced that lead to the structuring of the content portion via a net or graph approach. A hierarchy of content elements is introduced leading to a hierarchical definition of the notion of dependency. Linear algebra methods are employed in order to derive methods used to determine and to display the complete dependency structure for subject-matter content.

On the task side, assumptions are introduced that lead to a structured task space underlying the content structure. The behaviorally oriented tasks are then related to the content structure via coordinating relations. The notion of comprehension of content elements is introduced and related to the task structure.

A very detailed example of the complete theory is introduced and the theoretical concepts are discussed and described within the framework of the example.

Appendix B

PROJECT IMPACT COURSEWARE SUBSYSTEM: VOLUME 1, INNOVATIVE PROCEDURES FOR DEVELOPMENT AND ADMINISTRATION

HumRRO Technical Report by Michael J. Hillelsohn (in preparation)

Summary

Background

The objective of Project IMPACT was to evolve a series of prototype systems of Computer-Administered Instruction (CAI) in order to produce a total CAI system that is effective, efficient, and cost-effective for operational use in a training/instructional environment.

The total prototype system includes four main components:

(1) Hardware—the computer, student stations, and related equipment.

(2) Software—the computer programming systems that control operation of the hardware.

(3) Courses of Instruction—the actual content and logic of courses administered by the computers.

(4) Instructional Decision Models (IDMs)—the rules and strategies by which specific course content is provided to an individual student.

The way in which the content of instruction is prepared, stored in the computer, managed by the computer while students are taking a course, and managed off-line by course authors and administrators is the key to the efficiency and effectiveness of the total CAI system. All components of the system interact in the courseware subsystem, where one visible product, a course of instruction, interfaces with the learner. Development of the instruction within the constraints specified by the other three subsystems, and administration of the instruction so that all facets of the total system are optimally employed for the ultimate benefit of the student are the subject of this report.

Objectives

The original IMPACT objective (in 1967) for developing a course of instruction was as a vehicle enabling validation of the components of the total CAI system. After the first iteration of the course was tested, a period of reassessment resulted in more stringent and exhaustive objectives for the courseware subsystem.

Maintaining the course materials as a valid vehicle for instructional design remained the primary objective. Additionally, the subsystem had become so large (one course was over 30 hours long) that efficient management techniques had to be incorporated—both on-line and off-line—for the development and administrative activities related to the courseware subsystem. Increased cost-effectiveness in the production of CAI materials was another goal during the second iteration of the courseware subsystem.

Also of extreme importance in the second iteration was the objective of flexibility of courseware. In individualized instruction, a teacher/author must be free to develop creative and adaptive materials to meet the student's needs as they vary over time

throughout a course of instruction. Flexibility also implies that the courseware is amenable to change without disrupting the entire course administration.

Approach and Development

All of the objectives were met by developing an innovative set of standardized procedures for the design, production, and documentation of the IMPACT Courseware Subsystem.

The most important software development, using the computer as an efficient tool, was the physical and logical separation of instructional text from the instructional logic files. This IMPACT innovation aided the attainment of the aforementioned objectives by permitting on-line creation and modification of text of logic independently.

This capability resulted in effective file management, economy of text storage, reduced on-line development time, efficient on- and off-line retrieval and modification of each of the files, and more adaptive instructional decision models (IDMs). Off-line developments, such as the File Activity Control System (FACS) and title listings, were also made possible and practical because of the text and logic separation. These products in turn made it feasible for course managers and authors/teachers to do much of their work off-line and thereby use relatively expensive computer time most effectively and efficiently.

Development of the subsystem was evolutionary and iterative. The requirements of potential users were anticipated, and a multidisciplinary team designed and implemented solutions to these requirements. After implementation, a period of assessment occurred, during which the innovation was evaluated. If successful, the technique became part of the subsystem; if not, additional solutions were tested until a satisfactory one was found.

Appendix C

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

HumRRO Professional Paper 10-73, by E.W. Schneider, November 1973; Presented at the Meeting of the Association for the Development of Instructional Systems (ADIS), April 1972

Abstract

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.

Appendix D

COURSEWARE

This appendix consists of the COBOL2 and INTRO courses, with all supplementary materials. The form in which the materials are delivered is as follows:

COBOL2 and INTRO (EXT and Logic)	- microfiche
Glossary	- microfiche
Student Reference Manual	- loose-leaf bound
Auxiliary visuals	- 16mm film

Appendix E

LETTER ON DISSEMINATION OF COURSE MATERIALS

THE UNIVERSITY OF TEXAS AT AUSTIN
COMPUTER-ASSISTED INSTRUCTION LABORATORY
AUSTIN, TEXAS 78712

Sutton Hall 312

April 19, 1972

Dr. Robert Seidel
Senior Staff Scientist
HumRRO
300 North Washington St.
Alexandria, Virginia 22314

Dear Bob:

I have had a chance to review the documentation and computer listings you and Ed Schneider gave me during my recent visit to HumRRO. I can report that my initial impressions during the visit were quite correct; this is indeed very useful material. Already I have decided to incorporate several of your ideas and developments, which I know are based on sound theory as well as applied experience, into the design of the TICCIT data management system.

It seems that after a number of years of R & D in a very complex field, you and your group have found solutions to numerous problems that have stood in the way of the development of really effective CAI systems. I think that as soon as you catch up on some of your documentation, the field of computer-assisted instruction is going to profit substantially from what you have done and are doing.

Cordially,

C. Victor Bunderson
Director, CAI Lab

/amz

Appendix F

FILE ACTIVITY CONTROL SYSTEM (FACS)

HumRRO Research Product by Leslie Willis, Jean Garneau, and John Stelzer, Volume V in *Project IMPACT Software Documentation*, RP-D1-72-4, August 1972

Abstract

The Project IMPACT File Activity Control System (FACS) is an authoring aid used to assist in the development of instructional text. FACS provides printouts of textual elements in the exact format that they appear to the student on the cathode ray tube. FACS also provides printouts of logical units of instructional elements in compressed form. FACS allows an author to perform character string searches on the instructional text files in order to identify elements that contain specified character strings. Instructional elements can be stored with administrative data identifying the author of the text and the date of preparation. Generally, FACS prints this information with its reports and allows modification. This report describes the use and operation of the FACS system. Reports produced by FACS are also described.

Appendix G

PROJECT IMPACT SOFTWARE DOCUMENTATION: OVERVIEW OF THE COMPUTER-ADMINISTERED INSTRUCTION SUBSYSTEM

HumRRO Technical Report 72-21, by John Stelzer and Jean Garneau, August 1972

Summary

Military Problem

The combination of shrinking financial resources and the prospects of a smaller, all-volunteer Army will increase both the demands made on Army personnel and the importance of the individual soldier. There will be a greater need for more effective and efficient training, adequate to the task of providing an increasing number of complex skills to widely differing students, while using fewer skilled instructors.

The most promising approach available to meet these new training demands is computer-administered instruction (CAI), if it is developed as a comprehensive, total system.

The goal of Project IMPACT is to provide the Army with an effective, efficient, and economical CAI system in a total system framework. To be effective, the system should maximize the achievement of the students and the instructors to a greater extent than is possible in the traditional classroom; to be efficient, it should provide maximum productivity per unit time on the part of instructors, administrators, and students; and to be economical, the cost and resources must not exceed those of a comparable effective non-CAI instructional system.

Development Problem and Approach

Project IMPACT was established by the Department of the Army in 1968 as an advanced development effort to provide a total system of CAI for the effective and efficient training of military personnel. Accordingly, a Technical Development Plan (TDP) was conceived that provides for the concurrent development of the four facets of a total CAI system: instructional content, hardware, software, and instructional decision model (IDM). The Project was organized to keep these facets in balance over a span of two generations of CAI systems and four successive cycles of development and testing. The initial two cycles covering the development and test of a "breadboard" CAI system have been completed. The second two cycles were planned as a period for refinement of all facets of the system, to produce a prototype model to be tested, evaluated, and then delivered to the Army as specifications for an operational instructional system.

In pursuing its goal, Project IMPACT has followed an evolutionary approach toward developing products usable by Army instructional staff. This document describes the overall first generation, IMPACT-A, software products. The intent for widespread Army use is to provide functional requirements for a cost/effective system.

Products

The documents in the software series have been prepared to assist systems programmers in incorporating all or some of the IMPACT-A software products into on-going CAI efforts. While the primary purpose of the initial generation was to develop and test a provisional total CAI system, many of its products, such as the time-sharing software, data management capabilities, and IDM guidelines can fulfill user needs now. Subsequent products from the continued effort (IMPACT-B) will document the revision and refinements to these items. The software products are:

(1) Zeus Documentation—operationally available time-sharing software; authoring command set; separate text and course logic facility.

(2) FACS (File Activity Control System)—a set of computer programs that provides information concerning display pages that are disk stored; a system to assist in editing and coordinating displays.

(3) IDES-1 (IMPACT Data Evaluation System—Version 1)—a set of computer programs that manage the data collected, stored, and processed by the CAI system (no longer used).

(4) IDES-2 (IMPACT Data Evaluation System—Version 2)—an updated software system that provides for storage, retrieval, and analysis of student generated data.

(5) The Interface—a system that maintains on-line records of the prerequisites that have been satisfied by each student; it also controls intermodule and intramodule transfers.

(6) Coursewriter III Functions—a version of IBM's Coursewriter III that performs response analyses, data generation, and branching for students and data manipulation capabilities.

An overview of this software system is presented in this report. The products are described in detail in a series of Research Products intended primarily for personnel working in the computer software field.

Appendix H

THE INTERFACE SUBSYSTEM FRAMEWORK FOR INSTRUCTIONAL DECISION MODELING

HumRRO Research Product by William Underhill and John Stelzer,
Volume IV in *Project IMPACT Software Documentation*,
RP-D1-72-3, August 1972

Abstract

The IMPACT Computer-Administered Instruction (CAI) software subsystem utilizes Coursewriter III as its primary vehicle for providing student instruction. IMPACT Coursewriter III instructional material is structured into divisions, with each division having one or more instructional modules. Each module has a Telling (T) section with a Practice (P) subsection, and a Quiz (Q) section. A student may recapitulate any completed module, review his current module's T-section, or jump to the current module's practice or quiz sections. System-scheduled remediation is also provided for in IMPACT's instruction. The Interface controls all intermodule and intramodule transfers. It is used to assemble the appropriate label when a transfer is made. The label is returned to Coursewriter III and is used in a Coursewriter III branch instruction. Interface permits an author to specify, for each individual student, a separate and unique division and module structure. Thus, it also allows the author to specify an individual course for each student with the course components being drawn from a pool of instructional material.

Appendix I

IMPACT'S COMPUTER-ADMINISTERED INSTRUCTION SOFTWARE SUBSYSTEM, COURSEWRITER III, AND ITS FUNCTIONS

HumRRO Research Product by Doris Shuford and John Stelzer,
Volume VII in *Project IMPACT Software Documentation*,
RP-D1-72-6, August 1972

Abstract

The computer-administered instruction (CAI) language component in Project IMPACT's CAI system is an IBM program product, Coursewriter III Version 2, which has been modified slightly at IMPACT. The modifications concern what data are recorded by Coursewriter III and how and when data are recorded. The modifications also provided for special handling of invalid sign-on attempts, and special processing of commands and symbols not normally recognized by Coursewriter III. IMPACT has also developed and has in use several Coursewriter III functions for clearing counters and switches, for storing and loading buffers and counters, for special processing of buffers, for processing student response, for recording data, and for performing arithmetic computations on-line. This document provides detailed documentation on all Coursewriter III modifications made at IMPACT and on all Coursewriter III functions used at IMPACT.

Appendix J

THE IMPACT DATA EVALUATION SYSTEM—VERSION 1 (IDES-1)

HumRRO Research Product by John Stelzer and Leslie Willis,
Volume III in *Project IMPACT Software Documentation*,
RP-D1-72-2, August 1942

Abstract

This report describes the IMPACT Data Evaluation System—Version 1 (IDES-1); IDES-1 has two main components—storage/retrieval and analysis. The storage/retrieval component is used to update and maintain an extensive data base of Computer-Administered Instruction (CAI) generated data, as well as to retrieve selective data elements from the data base. The data are used for psychological research in learning, and for evaluating the instructional material. In IDES-1, the storage/retrieval function is performed through a list processor, SLIP. IMPACT's version of SLIP has been modified and extended for more efficient operation. The analysis function in IMPACT is intended to provide statistical analysis of data base subsets. This function is performed through the BMD statistical analysis package, augmented by specially prepared programs. This document describes in detail the storage and retrieval portion of IDES-1 (SLIP itself and the BMD package are not described).

Appendix K

THE IMPACT DATA EVALUATION SYSTEM—VERSION 2 (IDES-2)

HumRRO Research Product by Leslie Willis and John Stelzer,
Volume II in *Project IMPACT Software Documentation*,
RP-D1-72-1, August 1972

Abstract

The IMPACT Data Evaluation System—Version 2 (IDES-2) provides a storage, retrieval, and analysis capability for data generated in Project IMPACT's CAI environment. IDES-2 uses standard PL/1 techniques to perform the required storage, retrieval, and file maintenance activities. Statistical analysis in IDES-2 is provided through the Biomedical (BMD) statistical analysis package, augmented as required at Project IMPACT by especially prepared routines. IDES-2 provides extensive, standard reports summarizing student activity, which are used by authors to evaluate the effectiveness of the instructional material. IDES-2 reports are also used by IMPACT's research personnel to monitor student activity. As a result, IMPACT is able to develop increasingly more efficient instructional decision models. The storage and retrieval component in IDES-2 is documented in detail in this document. All IDES-2 reports, including the method through which each report is generated and its contents are described, and examples of reports, are provided.