DOCUMENT RESUME

ED 145 826 IR 005 211

AUTHOR Meadow, Charles T.; And Others
TITLE Individualized Instruction for Data Access (IIDA). Final Design Report.
INSTITUTION Drexel Univ., Philadelphia, Pa. Graduate School of Library Science.; Franklin Inst. Research Labs., Philadelphia, Pa.
SPONS AGENCY National Science Foundation, Washington, D.C. Div. of Science Information.
PUB DATE Jul 77
GRANT DSI-76-09737
NOTE 171p.

EDRS PRICE MF-$0.83 HC-$8.69 Plus Postage.
DESCRIPTORS Computer Assisted Instruction; Computer Oriented Programs; Computers; Data Bases; *Information Retrieval; Information Services; Information Systems; Models; Relevance (Information Retrieval); Scientific Personnel; Scientific Research; *Search Strategies; Technological Advancement; Technology

ABSTRACT

This report describes the development of a design and a small working model for a system for teaching and assisting in the performance of bibliographic on-line searches through the use of an intermediary computer. The system in its present form is intended for direct users of scientific and technical information systems. However, it is felt that the intermediary computer may become the standard way to receive information about, gain access to, and make productive use of the increasing number of complex and non standard computer services. Appended are tutorial programs and a transcript of a test use of the model. (Author/STS)

Individualized Instruction for Data Access (IIDA)
Final Design Report

Prepared by

Charles T. Meadow, Principal Investigator
Thomas T. Hewett
D. Jean Rafsnider
David E. Toliver

Drexel University
Philadelphia, Pennsylvania 19104

Bernard Epstein
Janet V. Edelmann
Ann Maher

The Franklin Institute
Research Laboratories
Philadelphia, Pennsylvania 19103

Under Grant number DSI-76-09737 from the Division of Science Information, National Science Foundation

Published by the Graduate School of Library Science, Drexel University, Philadelphia, Pennsylvania 19104. July, 1977

ABSTRACT

Individualized instruction for Data Access (IIDA) is a project whose intent is to provide active assistance to users of on-line bibliographic data base searching systems while they are solving an information search problem. It also provides instruction prior to use. The system is being primarily developed for scientists and engineers who are the end-users of scientific and technical information. Assistance is provided through an intermediary computer which monitors the user's interaction with a data base processor, analyzes his performance and provides assistance to him in the conduct of the search. Tutoring, in the form of computer-assisted instruction, is also provided through the same computer.

The diagnostic procedures were developed in recognition that there is no accepted model of a successful search, but that there are certain known errors or practices which can be machine-detected and whose occurrence will deter successful results. In using IIDA, a user is searching a publicly available data base through a commercial search service. IIDA is interposed between him and the data base computer.

Evaluation and planning for implementation and productive use of the system are included in this report.

KEYWORDS: data base search, information retrieval, search strategy, man-machine communication, user performance, user performance evaluation, search intermediaries, computer-assisted instruction, information system users, information systems evaluation, information systems marketing.

PREFACE

This report covers the design of a system for Individualized Instruction in Data Access (IIDA), a method for teaching and assisting in the performance of bibliographic on-line searches entirely through use of an intermediary computer. The purpose of the project is to develop a method of teaching potential users of scientific and technical bibliographic data bases how to use these resources. Preliminary tests indicate the method can be successful and the design indicates that its cost should not add materially to the cost of direct use of the data base search services, and may reduce it.

Bibliographic searching is a problem-solving activity. The complete plan for performing a search cannot be specified in advance. Learning the mechanics of searching -- the commands one may issue to the computer -- does not constitute learning how to search. There is technique or strategy to be learned as well. Because of the ill-defined nature of searching, there can be no mechanistic way to measure the correctness of a particular users search. IIDA is a heuristic system -- it depends upon techniques that appear to 'work' even if not soundly based in theory.

IIDA is not solely a teaching system. Its most important role is to assist in the performance of searches, helping users in the actual performance of a real task.

The concept of IIDA is new, but we envision it as becoming the routine way, in the future, for users to deal with the increasing number of complex and non-standarized computer services. The intermediary computer will become the standard way to receive instruction about, gain access to, and make productive use of these many on-line facilities.

IIDA is intended, in its present form, primarily for direct users of scientific and technical information systems. These are typically scientists and engineers who perform a small number of bibliographic searches per year and who have not found working through a technical library to have been a productive process in the past.

The system is not intended to supplant the professional intermediaries, but to enable those who do not use library services to gain access to information systems. We believe that greater understanding by users of how these systems operate will also enhance the ability of users to work successfully with intermediaries.

This report concludes the first year of work on the IIDA project. We undertook to design an operational IIDA system, to produce a working model, and to develop plans for use and evaluation of the technology.

In Section 1 of the report we discuss the objectives and the overall plan for development of IIDA.

Section 2 is a description of the instructional subsystem. It is a performance-oriented description of IIDA; a set of performance requirements.

Section 3 is a technical description of the computer system, hardware and software, needed to implement the requirements set forth in Section 2.

Section 4 discusses plans for implementation of the system and its initial use, marketing and evaluation.

Section 5 contains a description of the working model of IIDA and the early results of its use.

Appendix A contains specifics of proposed course outlines.

Appendix B contains the transcript of an actual use of IIDA during our testing period.

For any reader desiring a quick overview of the system, Section 1 might suffice. Readers not concerned with computer and programming details might prefer to skip, at least initially, Section 3.

IIDA has been a joint project of the Graduate School of Library Science, Drexel University, and the Franklin Institute Research Laboratories. The project has been sponsored by the Division of Science Information, National Science Foundation. We acknowledge with thanks the guidance given us by Dr. Joel Goldhar and Dr. Carole Ganz of NSF. We also wish to thank Dr. Robert Rich of Princeton University, who worked with us and gave invaluable advice as a consultant on evaluation and utilization planning.

Philadelphia, Pennsylvania

July, 1977

Charles T. Meadow, Professor of Information Science
Thomas T. Hewett, Assistant Professor of Psychology
D. Jean Rafsnider, Research Assistant
David E. Toliver, Research Assistant
all of Drexel University
Bernard Epstein, Associate Director, Science Information Service
Janet V. Edelmann
Ann Maher
Franklin Institute Research Laboratories

TABLE OF CONTENTS

1. OBJECTIVES AND OVERALL PLAN OF DEVELOPMENT

1.1 The IIDA Concept

IIDA - Individualized Instruction for Data Access - is a system that provides instruction in the performance of a man-computer interactive activity. IIDA can administer conventional computer-assisted instruction, a mode in which IIDA takes the initiative, provides instructional material to a student, and asks questions about it. Its uniqueness, however, comes from monitoring the interaction between student and computer, evaluating the performance of the student, and communicating to him information about errors detected and positive suggestions for proceeding. In this mode, IIDA assists a user in the performance of a complex task. The interactive activity being taught is bibliographic searching. It is the intent of the project to produce a system which can teach science students or scientists how to perform searches without requiring them to take classroom instruction.

Mechanically, IIDA makes use of an intermediary computer which is placed between the student user and a data base processor. The intermediary computer administers computer-assisted instruction and performs analyses of student actions, maintains student records and provides the conversational programs that are used to help the student understand his errors and his options. The IIDA intermediary computer does not itself search a bibliographic data base. Instead, it communicates with a commercial data base search service so that the student, even while learning, is working with a real data base and a real search service.

IIDA operates in any of three instructional modes: tutorial, exercise, and assistance. In tutorial mode, IIDA is a computer-assisted instruction system. In this mode, students learn search mechanics, data base structure, search strategy and how to use the special IIDA features not available from the data base service.

In exercise mode, students may work on searches of their own choosing, but are constrained in how they perform the search. IIDA expects them to try all commands available and to search in a manner prescribed by the system. The purpose of the exercise mode is to give students practical experience in use of the commands -- it is not to accomplish a searching objective.

In assistance mode, the student works on a search of his own choosing and his objective is to accomplish that search successfully. IIDA does not constrain him in choice of commands or their sequencing. IIDA does provide him a great deal of diagnostic assistance if he needs it, the approximate equivalent of a coach helping him to do the search, but not telling him, step by step, how to do it.

In the exercise and assistance modes, IIDA does not prestore answers to questions and expect the students to match these answers. The system cannot 'know' how to solve the student's problem. The diagnostic facilities are the heart of the system. They help the student understand what errors he has made or why he has failed to

progress in his search. They offer alternative possibilities. The better the student's performance, the less of IIDA's diagnostics he will see. For users who make only occassional bibliographic searches and who are likely to forget the details of the command language from time to time, IIDA can always be used. That is, IIDA is not a system which provides a few hours of instruction and then leaves students on their own. They can always use IIDA in assistance mode to help them solve an actual working problem.

In addition to administering the computer-assisted instruction and the diagnostics, the IIDA computer performs a number of user-invisible data communications tasks. One of these is to serve as a concentrator, so that several users can share a single communication line between the IIDA computer and the data base processor. This line can operate at a higher speed than most user terminals and can hold an incoming message until some free time is available on the line. The cost of the IIDA computer will be far less than that of the data base processor; hence the cost of using the data base processor through IIDA and its concentrator need be no more than the cost of going direct, with a single line per user. After the diagnostics -- this is the second major innovation of IIDA -- it can reduce the cost of using data base services, even for experienced users.

The primary intended users of IIDA are the scientists and engineers who are the end-users of the information. Normally, users work through an intermediary who may be a librarian or a specialist in the user's discipline. We are departing from what has become tradition in bibliographic searching. But, there has been no proof of the efficacy of the intermediary system and little discussion of alternatives. IIDA is an alternative. In Section 1.2.5 we discuss some recent studies of information use involving intermediaries.

Offering a service not involving intermediaries does not mean that we propose to abolish them. As end-users are trained and use IIDA for occasional searches, they can become more effective partners when they do work with intermediaries on the more difficult search problems.

1.2 Characteristics of the User Group

We are concerned with five categories of user characteristics: some personal characteristics, such as educational background and computer experience; how they perceive a search to be organized; how they select commands and make use of them; their behavior as a market for information systems; and user attitudes toward human intermediaries.

1.2.1 Personal characteristics

The student is assumed to be a scientifically trained person. While this has no rigorous definition and it is not important for us to have one, we mean by it an engineer, chemist, physicist, biologist, psychologist, etc., who has or is pursuing the equivalent of at least a bachelor's degree in some field of science. Thus, even if not familiar with the literature of his field, the student will know that there is a literature, will know the vocabulary of his field and should not be bothered by the task of learning a new skill which is largely mechanical. Indeed, a high proportion of students will have studied computer programming or have at least worked with programmers or computer output. We should not have to contend with the problem of overcoming a student's resistance to the idea of using a computer in a "humanistic" endeavor.

Similarly, although the student may not be formally trained in mathematics, the notion of set or of the elementary boolean set operations should be either already known or trivial to learn. At any rate, these concepts should not be a major intellectual challenge, either to student or teacher.

A key assumption about the student is that he is a serious professional person who wants to learn this new skill. It is not a requirement of IIDA tutorial material, therefore, to create in the student the basic motivation to learn bibliographic searching. However, since we have also assumed that the student is a person who, for whatever reason, has elected not to work through an intermediary in his searching efforts, we must avoid giving the impression that this is a trivial task, easily mastered in an hour or two of instruction. In particular, we would like to be able to motivate him at least to be willing to acquire a user's manual to use as a reference and as a means of self-instruction in learning other data bases or search systems than we will teach.

Some students will be motivated to learn more about on-line searching and become more frequent, direct users of the existing search services. Others will elect to continue their future work through IIDA. Still others, having gotten an initial exposure, may wish in the future to work with an intermediary. When this last group uses librarian services, their new knowledge will enable them to work in a completely new relationship with librarians than previously. They will know the vocabulary and much of the mechanics of searching. Without frequent practice, they will not know the current contents of the data base and this will prevent them from becoming expert searchers. Because of their new ability to communicate with the intermediary in the latter's terms, they may well become far more effective users of information systems than ever before.

1.2.2 General organization of a search

Penniman [1.1] identifies four phases of a search: index search, logic formulation, document display (actually record display - the documents are rarely represented in bibliographic retrieval systems), and other (including off-line print and such administrative commands as BEGIN, END or FILE, the last of which changes the file being searched). In Lockheed DIALOG terms, index search would include the EXPAND command and SELECT when it was used with a single descriptor. When SELECT is used with more than one descriptor for full text searching, we would place it in the logic formation category.

Standera [1.2] categorizes commands in a slightly different way: start/stop, executive (all logic and display commands), supportive (index search, set history, etc.), informative (information about the data base or procedures for search), and auxiliary (e.g. ordering copies of hard copy documents).

For the purposes of our tutorial approach, we have used Penniman's phases modified somewhat by Standera's:

1. Index search. (Search of thesaurus or inverted files on a single term.)
2. Logic formation. (Combinations of sets or text search using a combination of descriptors.)
3. Record display.
4. Procedural. (These are commands that do not produce search results but are necessary to organize the search, e.g. BEGIN, END, FILE.)

5. Diagnostic. (These are requests for information or for assistance in recognizing errors or poor performance.)

The first three of these categories define the order of proceeding that we will teach our students, that is that a search begins with commands of the first type and proceeds through the second to the third. Procedural commands and diagnostics come where they must. This order of searching seems to have some degree of naturalness to users and so will be used during the early stages of training. Students will not be required to follow this order in doing their own searches.

1.2.3 Selection of commands and timing

Penniman and Standera both report on approximate timing of searches and, very roughly, they indicate that a typical search has these characteristics:

Duration: 15 - 30 minutes
Number of commands used: about 15
Interval between commands: 1 - 2 minutes

Penniman's data on elapsed time indicate a negative exponential curve, with mode in the range of 0-5 minutes per search, and mean about 15 minutes.

We lack good data on how long it takes D.\LOG to execute a command, but our intuitive feelings and unrecorded experience suggest that it takes far less than 1 - 2 minutes consumed by users. Thus, most of the time the user's terminal is either free or being used for output typing, but since the communications lines are full duplex, there is considerable unused capacity in the entire system.

The research reports we have used do not describe users precisely. We assume that they were relatively experienced searchers. We further assume that less experienced searchers will consume more time between commands but not necessarily use more total elapsed time. They may feel pressured to minimize their total time.

Standera states that only a relative handful of different commands are actually used in most searches. Users, therefore, are generally confining themselves to a small subset of the total language. This behavior also follows our institution and information observation in Drexel's Information Systems Laboratory. It suggests to us that the teaching of the commands be broken into segments: first the basic command set, then practice with its use, then the advanced command set and then practice with its use. The advanced training may be made optional.

1.2.4 Market behavior

In developing the concept of IIDA, it had been our original intent to take it directly to industrial scientists for use -- this, after all, was the user group for whom it was primarily developed. More investigation led to the conclusion that this would be an unusually difficult group to introduce to an unproven system. We decided, therefore, to begin testing IIDA with college engineering students and graduate students. Industrial scientists or engineers will also be tested for their reactions and for comparison with student reactions. The main point is to be able to 'sell' people on the use of IIDA. It is a new service for which a market must be created and the market is national or even international in scope. By working, even for several years, primarily with student groups, a user demand can be created as graduates move into positions in industry and government.

Most of the market studies we have found relate to the use of data base search services through the mediation of a library. Wish and Wish [1.3] for example, report on their own work and survey that of others in this domain. Because IIDA is primarily directed toward delivering data base services directly to the user without an intermediary and to offer him the possibility of continuing to work through IIDA even after training is complete, we feel these studies are not particularly relevant to our project.

Some work will be done in the next phase of IIDA development on planning market strategies. The concentration, however, will be on proving the efficacy of the techniques. Only by actual experience with some users can we begin to make specific plans for marketing. Hence, the question of acceptability to industrial users is being postponed, not dropped from consideration.

1.2.5 The setting for on-line services and the use of intermediaries

The way the on-line searching industry has evolved, provision of its services through a library, with a librarian as an intermediary, is the norm. Probably the principal exception is that in some highly technical organizations, typically firms or laboratories in the chemical industry (including petrochemical and pharmeceutical). In these a terminal may be located organizationally closer to end-users and the intermediary is likely to have a professional degree and perhaps a title such as information chemist.

Little if any consideration has been given to studying whether this state of affairs is best or what the alternatives might be. In particular, we find no research on end-user reaction to the quality of intermediary services although there is evidence that library shunners constitute a substantial portion of any non-library population [1.4].

Thus, we feel justified in assuming that IIDA is, indeed, aiming for a previously unexplored market -- a market for search services provided directly to the end user, in his own organizational environment, without the necessity of an intermediary, but also without denying the possibility of using one should he so elect. Hence, rather than base our marketing on previously done work, we must carefully distinguish between user studies based on in-library use and our own projected in-office use.

Two of the best known studies of user behavior were those of the Systems Development Corporation -- Impact of On-Line Retrieval Services [1.5] -- and Lockheed's Investigation of the Public Library as a Linking Agent to Major Scientific, Educational, Social and Environmental Data Bases [1.6]. These studies were concerned primarily (SDC) or exclusively (Lockheed) with in-library use of systems. Both implicitly assumed the use of a professional intermediary and neither evaluated the end-user's attitudes toward the quality of the intermediary's services.

1. The SDC Study. SDC notes that search services aimed primarily at "individuals in organizational units other than the library community" such as Mead Data Corporation and the New York Times Information Bank were excluded from the study (p. 2).

A figure possibly significant for IIDA is that among "infrequent searchers" slightly more than half (56%) felt "fairly comfortable and efficient" at the terminal and the remainder "barely comfortable and not very efficient" or "had to relearn the system every time I use it." About 22% of the sample surveyed put themselves in the infrequent searcher category (again, recall, they are almost always librarians or

their managers, not end-users). All judgments of searcher efficiency are self-made (p. 67).

Some data is provided on search time: separate figures for search times when the respondent was learning and "now," as the survey was conducted. Both distributions are so flat as to suggest that there are other, unreported variables involved; that is, the probability of a search taking t minutes, until t gets fairly high (over 94 for beginners) is nearly equal for all t. Penniman's data [1.1] on the other hand, shows a markedly different distribution, but is based on a different user population (p. 68).

We concur with the following:

> Our study shows that most on-line bibliographic searches are performed by information intermediaries . . . In the early days of on-line systems, there was a strong belief among many designers and planners that these systems should be designed for, and used by, end-users. This belief is still held by some but it has not been translated into practice. However, on-line systems permit -- and, some would say, demand -- a new relationship between the end-users and the information intermediary, and the challenge today is to define the kinds of interfaces that take full advantage of the potential of the on-line, interactive technology. (p. 193)

This confirms the lack of study of the intermediary process for on-line systems.

Our conclusion from the SDC study is that it hints at the existence of a body of users for an IIDA-like system but does not directly address the problem.

2. The Lockheed Study. This project is, of course, intended to be concerned with a library setting. In the study there was a period during which users had free access to DIALOG followed by a period during which half the usual rate was charged. Average search time ranged from about 30 to 18 to 15 minutes as the cost and user experience changed. These figures are comparable to Penniman's.

About half the users, high for average users of a public library, were scientists of some kind (p. 3-5). This is likely, then, to have been a sample of the eventual IIDA user group.

During the period when users had to pay for service, 80% said the service would be useful several times a year, "several" being undefined but consistent with the IIDA assumption (p. 3-7).

Lockheed cites a problem (p. 4-4) of maintaining familiarity with "all data bases [as] one of the most difficult faced by reference librarians conducting the on-line searches." This is a hint of a problem with intermediary services and suggests the possible value of IIDA to the end user or the intermediary.

1.3 Objectives of the Project

The general purpose of the Individualized Instruction for Data ACcess project is to teach users of bibliographic interactive information retrieval systems how to search and to assist them in the performance of searches. It is aimed primarily at the end user of information, rather than the professional intermediary or librarian. The typical prospective user was characterized in the previous section.

It is important to stress that the primary target group is one whose members are not now bibliographic searchers to any important extent. Thus, IIDA does not attempt to lure them away from other forms of bibliographic system usage; it attempts to provide yet another option for those who wish to become users.

Why the concentration on users, rather than intermediaries? First, a negative answer. It is not because of any lack of interest in or appreciation for the role of intermediary. More positively, it is because this prospective user group, the true consumers, is a large and often unserved one. We do not forsee any dimunition of the overall role of intermediaries, however successful IIDA may be. On the contrary, we see an increase. We believe there is a valid analogy with the computational use of computers. Prior to the development of FORTRAN in 1957, programming was almost exclusively in machine or assembly language (a language one-for-one with machine language commands). Such programming took much time to learn and was tedious to carry out. One result was that virtually all programming was done by professional programmers - the intermediaries of their day. Often, the problem of communication between user and programmer surpassed that of the technical design and production of the program. The user could not speak in programming terms. The programmer did not always understand the user's vocabulary and needs. Result: delay, high cost, frustration. We suggest it is often so today with bibliographic searching The intermediaries do not always speak the user's language (there are, of course, exceptions). The user has not the knowledge to speak in search system vocabulary. Result: poor communication, poor results, frustration.

FORTRAN and subsequent high-order programming languages made it possible for people with some technical backgrounds to become programmers. Often, they could write their own small programs faster and more easily than could a professional, when interpersonal communication time was considered. Did this replace the professional programmer? Indeed not. The demand for professional programmers is as vigorous today as 20 years ago. But professional programmers more often work on large, more complex projects, the ones that the partly-skilled user cannot handle. Furthermore, the partly-skilled user can now talk to the professional programmer in the latter's language, vastly reducing the communications problems between them. The user is a better user of intermediary services than when he was entirely untrained in programming.

So it will be with bibliographic searching, we feel. When users have some personal knowledge of the content and mechanics of on-line systems, they will be better able to communicate and use the services of an intermediary. When the job to be done is simple, they can do it themselves.

The formal objectives of this project are:

1. To teach searching. This will include tutoring in data base structure, search strategy, search mechanics and the use of the IIDA system and it will include providing assistance to users in the performance of actual searches.

2. To demonstrate the validity of the IIDA approach. This will be done both by evaluating the system in use under test conditions and by demonstrating the extent of users of the system, once completed.

3. To improve the use of scientific information services. The successs of IIDA depends not only on making the system work mechanically, but on bringing bibliographic information services to more people.

4. To make the system self-supporting. IIDA is a new idea in training and it is experimental in nature. Its success will be beneficial to the science and technology community and eventually to other fields. It is therefore a proper project for support by the National Science Foundation. It is an objective, however, eventually to make IIDA not only self-supporting, but to bring it to the point at which it can support additional research in the field.

5. To extend the technique to other applications. Not a direct part of this project, but a long term objective of project members is the extension of this training method to other fields. Presently, we feel it will become applicable and useful in teaching the use of any complex but structured operation done through a computer, such as the use of packaged computer programs.

1.4 Project Development Plan

To accomplish the objectives set forth above will require a development effort lasting several years. The major phases are: (1) Design, (2) Implementation, (3) Initial Use, (4) Revision, (5) Extended Use and (6) Evaluation.

The purpose of the design phase was to produce technical specifications for the system and a plan for implementing them. This report presents the results of that phase and brings it to a conclusion.

The implementation phase will produce the working computer system, programs, courseware, and evaluative techniques needed to bring the complete system to real users. It is the longest and most expensive phase. Its analogous task in the physical sicence world is construction of the apparatus upon which to perform experiments. To justify the assumption that IIDA can actually be successfully implemented, a working model was constructed during the design phase. This is reported in Section 5.

Following implementation will be a period of closely monitored initial use and formative evaluation. In this phase, actual users who will be engineering students at Drexel will use the system in conjunction with routine classroom assignments. It will be important during this phase to make others aware of the system and its accomplishments to prepare the way for extended use in Phase 5. This can come about through workshops on the system, demonstrations and papers in professional journals. Details of this planning will be developed during Phases 2, Implementation, and 3, Initial Use.

The fourth phase is called Revision. Based upon feedback from the initial user group, changes will almost certainly be required before the system can be put into broad use. The nature of the changes to be made cannot be anticipated, but will fall into the following categories: (1) content of the tutorial material, (2) diagnostic techniques and interaction with users, (3) mechanical performance such as computer response time or reliability, and (4) system coverage -- the specific data bases and search services taught through the system.

Upon revision after initial use, the system will be vigorously publicized and offered for use in a wide area. Again, the details will be developed in a later phase, but we must address questions of how to publicize, prove the value of the system, price the system, and provide supporting services such as maintenance programming and instruc-

tional services. It is our expectation that the system will become self-supporting during this phase.

Evaluation is not, in our view, a single action. We are receiving some evaluative feedback at this time from use of our model system and we will be performing other evaluations during the implementation and initial use phases. All these, however, are formative evaluations. They serve primarily to help the systems designers to improve the system in some respect and to convince prospective users of the system's possibilities. Once IIDA is well into extended usage, it will be appropriate to perform, or to have an independent organization perform, an evaluative study of the effectiveness of IIDA in carrying out its original, primary objective: the teaching and assistance in performing of bibliographic searching.

Approximate planned timing of the major phases is:

Design	July 1976		June 1977
Implementation	July 1977	-	September 1978
Initial Use	October 1978	-	June 1979
Revision	October 1978	-	September 1979
Extended Use	October 1979	-	Indefinite
Evaluation	1980 or beyond		

1.5 Progress to Date

The initial grant for Phase 1 of IIDA was awarded on June 10, 1976 to cover a period of approximately twelve months. This is the report of that activity.

The products of the study are three-fold: a technical design, a plan for utilization and evaluation, and a working model. The technical design work falls into two broad areas: design of the instructional system and design of the computer and data communications subsystem.

The instructional subsystem encompasses the teaching methodology, design of specific instructional material, design for exercise and assistance modes of operation which provide assistance to users in conduct of actual searches, and diagnostics used to help students understand errors and omissions. This subsystem is described in Section 2 of this report.

The computer and data communications subsystem is the means of implementing the instructional design. It includes computer hardware, software for the instructional and diagnostic programs and software for the intricate linking of the IIDA computer with the data base computer. It is described in Section 3.

The plan for utilization and evaluation (Section 4) describes how we propose to test the system, evaluate test results and make plans for the use of IIDA in real searching environments, whether in a university or industrial setting.

The model of the system is a computer program, implemented on a PDP-10 computer, that performs a realistic example of every major IIDA function. Such a program was developed and linked to a data base system using a small file of air pollution bibliographic records. Tests were run and the data from these tests used in preparing the final design. The model and its use are described in Section 5.

REFERENCES

1.1 Penniman, W. David, "A Stochastic Process Analysis of On-Line User Behavior," Proceedings of the Annual Meeting of the American Society for Information Science, Vol. 12, Washington, D.C., 1975, pp. 147-8.

1.2 Standera, Oldrich, "On-Line Retrieval Systems: Some Observations on the User/System Interface," Proc. ASIS, Vol. 12, 1975, pp. 38-40.

1.3 Wish, John R. and Mary Ann Wish, "Marketing and Pricing of On-Line Services," Proc. ASIS Mid-Year Meeting, Washington, D.C., 1975, pp. 1-16.

1.4 Gallup Organization, The Use of and Attitudes Toward Libraries in New Jersey, Gallup Organization, Inc., Princeton, N.J., 1976.

1.5 Wanger, Judith, Carlos A. Cuadra and Mary Fishburn, Impact of On-Line Services: A Survey of Users, 1974-75, System Development Corporation, Santa Monica, 1976.

1.6 Two-Year Interim Report, Investigation of the Public Library as a Linking Agent to Major Scientific, Educational, Social and Environmental Data Bases, LMSC-D502595, Lockheed Palo Alto Research Laboratory, Palo Alto, Cal., 1976.

2. INSTRUCTIONAL SUBSYSTEM

2.1 Teaching Methodology

IIDA will offer a range of teaching materials to the student, with no intent that any one student must use all of it or that it be used always in the same sequence. We assume our users will differ too much in prior experience, computer familiarity, knowledge of the data bases and experience or interest in searching for them all to start at the same point and proceed in the same fashion. Hence, the system will be designed with maximum flexibility for student control of sequencing. In doing this, we recognize that we incur a certain risk that some people will try to work beyond their level of skill and will become confused in doing so. However, completely preventing that would require too much control over students.

It would be desirable to be able to set down the exact teaching and learning objectives of the IIDA system. Unfortunately, there is no accepted statement of what skills are required of a competent searcher. There simply is no way to make measurements of a student's performance and pass judgment on whether he has or has not achieved some minimal level of competence. On the other hand, it is possible to make some measurements of his performance and to indicate, in various categories, how well he performs in comparison to other students. Over a long enough period, these and similar measurements yet to be devised may evolve into a true measure of skill in the task. To date, searching remains an art whose success is primarily self-judged: if the user is satisfied with results, cost and time, it is hard for an impartial observer to intrude and deny the satisfaction of the search.

IIDA's primary task is to perform in the assistance mode. We ascribe no particular merit to good performance in the tutorial mode if this is not followed by successful performance of actual searches. In other words, a 'grade' in the tutorial material carries no weight except as diagnostic information.

There are three modes of instruction: tutorial, exercise and assistance. The terms instructional material or instructional program may refer to any of these. The tutorial mode provides the student background material in the form of cognitive information -- facts about the data base and search system. The exercise mode represents the recommended starting point for learning performance. When operating in this mode the student will perform one or more actual searches, generally of his own choosing, but under control of the exercise program. He will be restricted in what commands he uses and how he uses them. The primary purposes of the exercise mode are to give the student some actual practice in searching, to get him to try each of the commands at his disposal and for him to achieve success as early in his training as possible. The assistance mode permits the student to work on his own search in an uninhibited manner. The system's diagnostic facilities are available when needed.

Students will be expected, normally, to work through the tutorial and exercise material as quickly as they can. They may remain in assistance mode as long as they like. There is no requirement that a student be 'weaned' of dependence on IIDA. We do not expect that searching through IIDA will cost more than searching without its aid, since our primary intended user group comprises only occasional users, they are not likely to retain all the details of search mechanics from time to time. Hence, they may elect always to work through IIDA's assistance mode programs.

These design decisions on the instructional material were made with a specific user group in mind. To repeat, these are working scientists and enginaers, or students in these fields. They are not professional intermediaries. They are not necessarily the office 'gate keepers' who function as informal intermediaries. They are not frequent (say one search per month or more) searchers. They are not uncomfortable with computers, although they may be uncomfortable with libraries. We do not wish to make the claim that this approach to training is best for all classes of users.

2.2 Tutorial Material

Given unlimited time and some way of requiring students to start with the tutorial material and work through all of it before proceeding to the exercise and assistance modes, we would prefer to start with a general appreciation for the literature of the user's field. If we could control the type of data communications terminal he used, we would include training on how to use it. However, we cannot make either of these assumptions. Hence, our approach is to teach the minimal mechanics needed to allow him to by-pass some tutorial material, or all of it, but to return for more complete training as he begins actual searching and only then realizes that there are some facts he would like to know but does not.

Most of the tutorial courses are, of necessity, specific to a data base or search service or both. There is strong sentiment in the profession for a universal command language, a lingua franca of searching. We are in sympathy with this feeling but recognize the technical problems of creating it as too imposing to be subsumed under this project. Our intent and obligation, therefore, are to teach the languages and data bases that are there and not to create new ones.

We will devote our initial efforts to teaching students one data base through one search service. Throughout the implementation phase of the project we will restrict ourselves to a single search service. Eventually, both the number of data bases taught and the number of search services taught, should be expanded. In expanding from one to many search services, it may be possible to switch to a universal language, if developed, to teach each language independently, or to use the language of our first service to search the others. The last approach probably implies some dimunition of service. For example, there is considerable difference in how the services search for strings or words embedded in a text. Using the language of any one of them to command any other of them is likely to reduce the effectiveness of the second service. In the basic commands, for selecting terms, combining sets, or browsing in retrieved records or a thesaurus, there is little difference among the major languages.

The following are the 'courses' that will be produced for the first implementation of the full IIDA system:

The search service to be used is DIALOG, offered by Lockheed Missiles and Space Company. The data base will be Compendex, offered by the Engineering Index, Inc. Each of these organizations will participate in the design of teaching materials relevant to their own product.

Below are the courses that will be produced for the first implementation of the full IIDA system. More complete course descriptions are found in Appendix A.

Structure of Compendex. This course covers material in the file, the method of indexing and abstracting, the structure of the bibliographic record. As the first course in data base structure, this will cover what is meant by data base structure as well as giving information about Compendex.

Search mechanics. This course will teach the DIALOG command language. Insofar as it is necessary or desirable to customize the course for the data base being taught, it will emphasize searching of Compendex. The course will cover all commands available through DIALOG and primarily teaches the syntax of commands and the results the commands produce. This course will be in two parts, teaching first a subset of the language, then the remainder of the commands.

Search strategy. This course emphasizes how commands should be used, rather than what results each one produces. It teaches the student how to view his search as a goal-oriented task that requires preparation and planning and for which some progress measures are possible. To the greatest extent possible, this will be taught independently of the specific search service or data base in use. Probably students will take this course after having done some searching.

IIDA diagnostics. This course reviews the diagnostic facilities available to the student user of IIDA. This course, too, might be more profitably taken after the student has had some actual searching experience. A clear distinction must be drawn in this course, between IIDA diagnostics and those provided by the search service, so the student will be able to work independently if he chooses to begin searching without IIDA assistance.

All of the foregoing material is administered as conventional computer assisted instruction. Records of student performance may be stored but there is no meaning imputed to performance at this level, nor would the results of this part of the instruction ever be reported to an instructor who is grading student performance in searching as part of some other effort, such as preparing a term paper.

2.3 Exercise Mode

The purposes of the exercise mode are to: (1) introduce the beginning student to actual searching, (2) virtually guarantee success in at least a simple search and (3) have the student experience the use of each of the commands available to him. Although the exercise mode program permits the student to define his own search, it is still an instructional program. The student's attention is focussed on how to use commands rather than on the completion of a search. Students returning to IIDA after a prolonged absence might use this exercise as a refresher training course. In either case the student must be aware that this course is not intended to conduct a search in the fastest possible way -- that it is primarily trying to teach him the use of the commands.

The exercise will be divided into three parts: (1) a minimal, 'canned' search involving the use of only a few commands, to enable the student to see quickly how a complete search is done and to give him the feeling of having stepped successfully through the entirety of the process, (2) a full exercise using only a subset of the command language to enable him to learn the basic commands in a practical context without getting bogged down in too many commands, and (3) a full exercise involving all commands in the repertoire. The last of these might be made optional.

2.3.1 The general design of an exercise

The exercises are designed to provide a transition from the tutorial format to the search format. An exercise resembles a search although it is designed to be a teaching tool, not a searching device. Part of the objective of an exercise, especially the minimal exercise, is to give the student a feeling of accomplishment by getting him through a search as quickly and painlessly as possible.

In appearance, the three levels of exercise all mimic a real search. Behind this appearance of freedom to search at will are the resources of the diagnostic programs. At each level of the exercise diagnostic programs are functioning. The /HELP function is available to the student at all levels of the exercise. This function can be called by the student whenever he feels the need for assistance (details are given in 2.4). The performance analysis routine (PAR) is alert to unsatisfactory trends of student input and calls the /HELP function for the student when deemed necessary. Details of the PAR are given in 2.6.3.

The exercises introduce search language and search strategy in stepwise fashion. Each exercise is complete and self-contained. At any time the student can proceed to the assistance mode where the purpose is searching not teaching, and the program becomes transparent except when /HELP is called. Program control becomes less visible as student search skills improve. The program structure of the exercises is discussed in 3.2.

The apparent structure of the exercises is based on what little information is available on real-world searching. That information suggests that searching can be regarded as a series of phases or states (see 1.2). These are: thesaurus search and set formation, logical combination of sets, and print of retrieved citations. The DIALOG search language commands that belong to each phase are;

Phase 1. SELECT and EXPAND, the latter used both to display alphabetically related terms and terms related in meaning.

Phase 2. COMBINE, in varying forms and formats, and SELECT when used with multiple descriptors.

Phase 3. PRINT.

These phases are not strictly serial in the real world, nor are they completely clear cut. However, it was decided to structure the exercise mode along the lines of the three phase search so that some structure could be imposed for teaching purposes.

The student is required to start in Phase 1 and to use all of the search commands associated with thesaurus searching and set formation. The formats of SELECT and EXPAND vary with the level of the exercise.

When the student has successfully tried all of these commands and wishes to continue to Phase 2, he is informed that he is now able to use only Phase 2 commands. The formats and exact commands vary with the

level of the exercise. These include the COMBINE command to form sets by logical combinations of previously created sets and the SELECT in its text search form, allowing multi-descriptor commands. Again, use of phase 2 commands is required before the student can proceed to phase 3.

We recognize the flexibility of on-line searching as being one of its major benefits, so in all but the minimal exercise, branching possibilities are presented when the student finishes phase 2, and can go on to 3 or back to 1. A menu is provided to choose from, although the choices are no different from those choices available at any time to an on-line searcher. The choices presented allow the student to pass back to another phase, to continue to the next phase, or to ask for help. The requirement of choosing which phase he wishes to be in works to emphasize the planning required in searching.

If the student chooses to go back to a phase already experienced, the controls on the required use of each command are relaxed. The commands belonging to each phase must still be used together, i.e. he must finish using the single-descriptor SELECT before using COMBINE which is in another phase. However, it is no longer necessary to use all command types in a phase.

If the student elects to proceed with printing citations from sets already formed, a relevance judgement is required after each citation is printed. We are stressing the importance of self-evaluation of the results of a search.

After viewing some citations, another branching point is reached. The student can return to phase 1 or phase 2, he can declare himself done with the search, or he can request /HELP. If he declares himself done, he is asked to evaluate the citations retrieved to emphasize again the need for searcher evaluation of results.

2.3.2 Implementation of the exercise mode

1. Minimal exercise. This exercise is intended as a bridge from the question-answer format of the tutorial material to the command format of an actual search and to get the student through a complete search as quickly and painlessly as possible. To illustrate the entire search process from beginning to end, a short search is laid out for the student in the minimal exercise. The student does the actual input of commands in the order of a real search, but the commands, descriptors and order of input are all prescribed by the program.

The commands available in the minimal exercise are

SELECT using a descriptor

COMBINE using AND(*) logic and two set numbers

PRINT using the format 2, which shows bibliographic information as well as indexing data. Format 5 of the PRINT command is explained as including format 2 plus the abstract, but its use is not required.

The minimal exercise is patterned after the first search taught by the National Library of Medicine's MEDLEARN system.

2. The subset exercise. The subset exercise allows the student more freedom in searching, more commands to search with, and more strategy for searching. The student picks his own search topic. The concept of search phases is introduced in this exercise as a structure for the student to hang his commands on. The three phases of selecting terms, combining sets, and printing citations are required to be used in this order to familiarize the student with all the phases and to allow the student to see that the phase structure really leads to a productive search.

The commands available in the subset exercise include those available in the minimal exercise with the addition of EXPAND. The SELECT command will be usable with a reference number from an EXPAND display as well as with a descriptor, but text searching will still not be allowed. The COMBINE command can be used with AND, OR and NOT logic. Two new print formats, 1 and 6 are introduced allowing for printing of only accession numbers and of titles with accession numbers. The program controls when the commands may be used and requires the use of all the commands available.

In keeping with the attempt to simulate an interactive search, branching is possible within this exercise. When phase 2 is completed, the searcher may return to phase 1. When all three phases have been experienced, the searcher is free to use the phases in any order.

3. The full command language exercise. With the full exercise, the search language is completely presented. Sophisticated search techniques such as full text gearching are made available and an attempt to use each of them is required. The phase structure of the search is maintained until all phases have been utilized once, then the searcher is free to move from phase to phase. The control over the search is still divided as in the subset exercise. The student chooses the subject and the descriptors, while the order in which the commands may be entered and the required use of all the commands are under the control of the exercise.

2.4 Student Controls and /HELP

The IIDA system provides several student controls which may be exercised any time the system expects an input. Their purpose is to secure assistance in searching, permit control over program sequence, and allow communication to other terminals. All of the commands begin with a slash (/) to clearly distinguish them from DIALOG commands. The commands are /HELP, /GOTO, /QUIT, /SEND, /DONE and /MORE.

2.4.1 /HELP

The /HELP command calls an assistance program that operates in conjunction with the exercise and assistance modes. Its purpose is to present descriptions of search and system commands and specific information on the search in progress. In its first aspect, /HELP can be used to review the format and purpose of all legal search and system commands, and thus may be used in lieu of much of the tutorial material by computer-experienced students. In its second aspect, /HELP provides data on the search in progress and tries to elicit ideas from the student on problem identification and solution.

In response to the /HELP command, a menu of types of help is printed. The searcher picks which type is most appropriate to his needs. The help menu includes the following:

1. Definition of search commands. The purpose and format of all available search commands can be reviewed, one at a time.
2. Definition of system (/) controls. The purpose and format of all system controls can be reviewed, one at a time.
3. Commands and search phase review. All valid commands used in the search so far are printed with the associated descriptor in the order entered. Additional data is provided on phases associated with the commands.

4. Review of sets created. The sets created so far are presented in tabular form with set number, number of items in the set, and creating command. If the command was a COMBINE, the constituent sets of the COMBINE command are translated into their definitions in terms of descriptors.
5. Review of sets already viewed. Sets viewed through the use of the PRINT command are reviewed in a table which shows the total number of records in the set, the range of records viewed, the format and the average relevance assigned to the range when viewed. If more than one range of records has been viewed for any set, each is shown.
6. Review of descriptors used. All descriptors used so far in the search are listed together with the type of command of which they were the argument.

2.4.2 /GOTO

The major use of /GOTO is in the exercise mode where the command may be entered only after having passed through all three phases at least once. The /GOTO command must be followed by the number 1, 2 or 3 which indicates:

/GOTO phase 1 (that part of the course where SELECT and EXPAND may be entered).

/GOTO phase 2 (that part of the course where COMBINE and commands of similar effect may be entered).

/GOTO phase 3 (that part of the course where PRINT commands may be entered.

In the assistance mode, /GOTO is meaningful only as a memory aid since the system has no control over search sequence. In this case, a prompting message would be returned reviewing which commands belong to the phase requested.

2.4.3 /QUIT

The /QUIT command terminates the operation of the IIDA program. All data about the search is cleared and control is returned to a reception program which offers the options of logging off or logging in to any IIDA program including the one just left. Users may /QUIT temporarily, in which case all results are saved until the search is resumed.

2.4.4 /SEND

The /SEND command initiates a communication procedure enabling the student to send a message to the proctor (2.7.4).

2.4.5 /DONE

The /DONE command denotes the completion of a phase or completion of a use of /HELP.

2.4.6 /MORE

The /MORE command is a signal by the student of interest in further information on the subject under discussion. /MORE can be used only when listed as a possible response.

2.5 Assistance Mode

The objective of the assistance mode of IIDA is to enable students to perform actual searches of their own choosing, in their own manner, while being provided assistance whenever needed. Unlike the exercise mode, assistance mode allows the student to use any search commands in any sequence. He is not constrained by search phases and need not make use of all the commands if he elects not to. The assistance mode programs monitor the student's performance and are ready to respond with diagnostic or other assistance whenever needed Need is determined either by the student or the program. The student can invoke assistance with the command /HELP or, if the program determines that his performance indicates any of a variety of error patterns or lack of progress toward a successful conclusion, the program can invoke the /HELP facilities itself.

We are again constrained by the lack of formal definition of how to proceed in a search. If there were an algorithm by which IIDA could determine exactly what the student should do next, presumably it could do it for him. Thus, we are dependent upon a set of empirically and intuitively derived diagnostic procedures. The intuition, however, is guided by extensive experience in teaching new users how to search and in observing the kinds of errors and misunderstandings that arise.. One of the long term outgrowths of this project, we hope, will be a definitive study of the measurement of progress of a search, or of degree of success, of of some related measure.

The assistance mode is the heart of the IIDA system. It is not primarily a tutorial program. The student's attention is intended to be focussed on achieving his search objective, not search mechanics. The system should intrude as little as possible, consistent with its objective of helping the student when he needs help. The various thresholds to be used to decide when the system will intrude will have to be adjusted with experience, as we learn both how much help students need and how much intrusion they will tolerate, as well as how much frustration they will tolerate without active help from the system.

Mechanically, the assistance mode is simpler than the exercise mode. It does not control choice of command or phase of search. It does, however, make use of a performance analysis routine (PAR) which determines when to interrupt a student. This program is based upon analysis of the student history data maintained for, and used by, the /HELP program. The performance analysis routine is described in more detail in Section 2.6.3 and 3.3.4.

In general, the assistance mode program will allow the student to use any command he desires. Syntax errors, occurring singly, can be corrected as quickly as the student can be made to understand his error and re-enter his command. The PAR will perform a series of threshold checks after each student input of a command. When PAR determines the likelihood of a serious problem, it triggers a series of more detailed analyses which are not performed after each command in the interest of saving time. When PAR determines the nature of the student's problem, it informs him and provides the appropriate diagnostic data, then offers other diagnostic assistance through /HELP. Thus, the assistance mode program is primarily a data recorder, augmented by PAR and /HELP.

When the student logs onto the IIDA system the ground rules are briefly discussed: any subject covered by the data base may be searched, any commands in any order may be used, assistance is available on call

by use of /HELP and assistance will be provided if IIDA feels the search is not progressing as it should.

The student is asked to indicate whether his search objective is to retrieve a few very pertinent citations (1-10), a short bibliography on his topic (10-25 citations), or an in-depth bibliography of the contents of the data base on the subject of the search (more than 25 citations). It will be pointed out that the student is in no way bound to this choice. The data is used to make the performance analysis routine function more adequately by having an idea of the student's quantitative goal for the search. The methods for searching for these three types of result could differ, a fact which could be confirmed by the way in which students use the system. In any case, the projected result of the search is of use in the PAR calculations.

Following this interchange with the student, the program disappears from the student's point of view. The reappearance of the program is conditional on the needs of the student. If he feels no need for assistance, he may never call the /HELP routine. However, the PAR continues to monitor his search. The PAR operates by checking thresholds as explained in 2.6.3. Each student input is checked against these thresholds. Since the number of thresholds has purposely been kept to a minimum, there should be no effect on search time as far as the student can tell. If the PAR discovers that a threshold has been crossed, an in-depth analysis of the area monitored by the threshold is initiated. At this point, the student is notified that the search may be going less well than it could. The PAR in-depth routines are capable of locating problem areas but not in all cases suggesting remedies. The problem is presented to the student and his own analysis is invited (for details see 2.6.3 and 3.3.4).

When the student and the program are satisfied that the direction of the search has been reconsidered, the student is returned to the search. Possibly a short review of his most recent searching activity before the PAR was invoked is provided. This is dependent on the type of assistance needed when the PAR is called.

2.6 Diagnostics

The general IIDA approach to diagnostics consists of: (1) recognition of syntactic errors in commands entered by the student, (2) detection of procedural errors (these are violations of rules imposed for pedagogical purposes only), and (3) recording and analysis of patterns of usage.

Search services are normally quite laconic in responding to syntax errors in commands. A typical response is 'Invalid command,' which gives the student little to work with when he tries to understand in what way his command was invalid and what he has to do about it. The opposite extreme in syntatic analysis is to try to pin-point an error using as evidence only the user's input, not his intentions. Here is a typical example. Suppose the user has entered the search statement.

SELECTION

This could have either of the following meanings:

SELECT ION

S ELECTION ('S' being the valid DIALOG abbreviation for SELECT)

or he may have meant but omitted to prefix a command abbreviation to give:

S SELECTION
or E SELECTION

Given only the character string SELECTION, there is no way a program can tell what was intended. An attempt to do so may be more misleading than the unhelpful "invalid command." For example, treating it as SELECT ION (which DIALOG does) returns a set number, count and description which can be quite confusing if "E SELECTION" was the intent.

Hence, our objective is to give more information than merely that an error was made but to try to avoid being misleadingly specific about the cause. IIDA performs its own parsing of user commands for two reasons. First, it stores data about the nature of errors made for later analysis. Second, some parsing is necessary in the concentrator program (See Section 3.6.4). Since the work has to be done, all useful information might as well be extracted at one time.

Procedural errors are encountered only in the exercise mode of IIDA, which restricts the student to use of certain commands at certain times, requiring that he use only those commands authorized during any given phase of a search. He cannot, for example use a print command in phase 1. Detection of procedural errors is a simple matter of looking up each incoming command in a list. The list to be searched is a function of the phase of the search.

Analysis of patterns of usage constitutes the bulk of the diagnostic work of IIDA. This analysis is performed at four levels: (1) collecting basic performance data from each student, (2) microanalyses of data elements, (3) threshold checking, and (4) overall assessment of performance.

2.6.1 Diagnostic data

The most fundamental data to be recorded is a simple record of the commands entered by the student and the responses to these sent from the data base processor. To this is added the results of some of the microanalyses. The data are classified as to purpose and recorded in the various history files. Each file represents a record of student performance from a different point of view and they are somewhat redundant with each other. The purpose of this organization is to enable the student, proctor, or diagnostic program to view the student's performance from different perspectives. Some students may prefer to review their own work in terms of sets created. Others may prefer to see the sequence of commands issued. Still others may be concerned only with the records retrieved.

This diagnostic data is available to the searcher in response to the /HELP command and is available to be used by the PAR. The PAR uses this data to decide whether a problem situation has arisen and uses the data again to present the problem to the student and to help the student resolve the problem.

The eight types of data to be maintained are: (1) command history (2) set history (3) sets viewed history (4) descriptors used history (5) error history (6) /HELP history (7) records viewed history (8) expanded index data. Typical examples of the kinds of data maintained are listed here, for detailed information on contents and maintenance of data files see 3.5.

1. Command history
Text of commands
Search phase for commands
Command type code
Time data

2. Set history
Set number
Type of command creating set
Number of items in set
Expansion of set description for those sets defined in terms of other set numbers, the numbers being replaced by the descriptors generating the original set.
Cluster number (see 2.6.2)

3. Sets viewed history
Set number
Which records of set viewed
Format in which viewed
Relevance of records

4. Descriptors used history
Text of descriptor
Number of times used as argument to SELECT, EXPAND, COMBINE
Number of records indexed under it
Number of related terms

5. Error history
Text of error
Code for type of error

6. /HELP history
Number of times each kind of help used
Location of /HELP call in search

7. Records viewed history
Record accession number
Set from which record came
Record number in that set
Format in which viewed
Relevance assigned at that viewing
Above is replicated for each time record viewed, whether as member of same or different set.

8. Expanded index data
First and last terms in every EXPAND table generated in this search (used to be able to tell whether any given descriptor descriptor was seen in an EXPAND table).

Files 1-4 or perhaps 5 will be available to the searcher through the /HELP command. All of the files are used by the PAR for diagnostic purposes.

2.6.2 Microanalyses

The microanalytic programs are those needed to produce the elements of the history files which are not mere listing of student or DBP inputs. The following programs are required.

1. Command parser. Purpose: to verify that student commands are syntactically correct and procedurally acceptable. Functioning: the program is given a list of expected elements by the instructional program that calls or invokes the parser (i.e. by the tutorial, exercise or assistant program). The calling program might specify that any of a set of commands is acceptable, with any legal argument (as a SELECT or EXPAND command with any descriptor). The parser determines if the required conditions are met and, if not, records the defects. The parser than converses with the student, informing him of the nature of the error and eliciting from him a new response (i.e. new command). Responsibility rests with the parser to get a correct command from the student. The parser maintains a record of errors made. See Section 3.3.1 for more detail.

2. Phase detector. Purpose: to determine in what phase of a search the student is working, recognizing that there are not necessarily sharp lines of demarcation between phases once beyond the exercise mode. (This program is of use only in assistance mode: the exercise mode controls the student's phase for him). Functioning: The problem is this: given a string of commands, there are embedded within it sequences of "like" commands, i.e. commands all belonging to a given phase. The difficulty is that these sequences may not be pure, there may be a phase 2 command amid the phase 1's. The phase detector must be able to recognize one or a small number of out-of-phase commands in a sequence as not necessarily breaking the sequence. For example the sequence 1112222233 clearly divides into discrete phases. In the sequence 11121122 we might say that all of 111211 is from phase 1, in spite of an out of phase command.

3. Set description expansion. Purpose: to assist the student by displaying descriptions of sets he has created previously in terms of descriptors rather than set numbers. Functioning: A record is kept of the argument of each command which creates a new set. When a COMBINE command is added to the data file, the set numbers used as the argument are traced back to the command which created each of the sets. If the argument to that command is another group of set numbers, they are traced back to their creating commands. At some point, the creating command will be a SELECT command. The argument of that command will replace the set number in the description of the command using that set number. The descriptors replace set numbers up to the command which initiated the trace. The COMBINE argument with elementary set numbers and the argument with terms replacing set numbers are both stored for each set formed with a COMBINE command.

4. Set description clustering. Purpose: As complex sets are created out of elemental sets (those defined by a single descriptor), some of these new sets will closely resemble others in composition and some not. The clustering algorithm will relate together those sets that use 'similar' descriptors in their definitions. It detects the point at which a student departs from work on a previous cluster and begins a new one and it measures the relationships among clusters. Functioning: A measure of similarity S(i,j) is used to compare the most recent command (C_i) with a previous one (C_j). If the measure of similarity is high enough, the new command is included in the same cluster as the previous

one. If not, a new cluster is created. Relations between clusters are measured by links. A link exists between two clusters if a command in one cluster has a high similarity measure to a command in another cluster. The measure between the clusters is the number of such links. Information on the point at which a student makes the transition to a new cluster and the relationship among clusters is provided to the performance analysis program. The clustering program is described in more detail in Section 3.3.1.

5. Descriptor viewing status. Purpose: this program will determine whether a given descriptor has ever been seen by the student, during this search, as part of an EXPAND display, irrespective of whether it was the search term in the display request or merely appeared in response to a search on another term. The program will make a distinction between being viewed in an EXPAND [term] display or a subsequent EXPAND display showing related terms, rather than alphabetical neighbors of the search term. Functioning: When a command has been identified as EXPAND [term] in the process of updating the data file, the first and last terms of the table generated are recorded on the expanded index data file, in alphabetical order by first term. When requested, the descriptor viewing status program checks the expanded index data file to see if the term in question falls between any of the pairs of terms listed there.

Some of these analytical programs can consume significant amounts of computer time. It is necessary, therefore, to carefully ration the frequency with which they are run. A clustering algorithm, for example, may be run only when the set history display is requested as part of /HELP. It probably (subject to timing estimates when actually programmed) should not be run after each command.

2.6.3 Performance analysis

The performance analysis routine (PAR) operates in conjunction with the assistance mode of IIDA. Its purpose is to detect difficulties or problems in the conduct of a search that might lead to failure. It does not look for specific syntactic or procedural errors. It looks for adverse trends. A well trained, highly perceptive student could operate without PAR, because he would be able to detect his own difficulties and summon the diagnostic facilities of /HELP to get specific information and ideas on proceeding. PAR is intended to make that initial decision for the student -- the decision to call for /HELP.

In order to avoid excessive delays in responding to students, it is necessary to avoid lengthy calculations or processes following each student input. The approach to be taken, therefore, will be to have the PAR make some rapid checks to see if there are any indications of problems and then to make detailed analyses only when the preliminary checks so indicate.

1. Preliminary Analysis. There are five preliminary analysis subroutines which are used to make rapid checks of student performance. These are: (1) phase and cluster analysis, (2) zero set formation, (3) use of extraneous and redundant commands, (4) use of /HELP and delays in responding, and (5) error analysis. The purpose of each subroutine is to quickly check one or more threshold values to determine if there is an indication of a serious problem. Since these checks are run after each student command, only a limited amount of time can be devoted to them if we are not to delay the student excessively. The exact threshold values can only be determined on the basis of experience. In the remainder of this discussion values are occasionally mentioned, but

these are illustrative only -- all are subject to change. Upon detection of a possible serious error, more detailed analytic programs are called and it is these that develop the data presented to the student in discussions of errors with him.

Threshold values may vary with the student's progression through a search. During the first few commands, thresholds may be set quite low, because too many errors here may indicate an unprepared student who is likely to produce only chaos if allowed to proceed. On the other hand, once he gets successfully underway, the thresholds can be relaxed to permit him an occasional error or an unsuccessful excursion into some particular subsearch. If the total number of commands issued exceeds some number (which will be based on our own and others' experience in monitoring searches) the thresholds may again be lowered on the assumption that the student is unable to bring the search to a conclusion and will need help in doing so. We use the term convergence to indicate that the number of elements in a series of similarly-defined sets (cluster) is tending toward a number consistent with the student's initial search goal. Failure to converge may indicate that the student searcher is trying to be overmeticulous or is expecting the system to produce results it is not capable of. We use the term thrashing to indicate the opposite behavior of excessive concern with one set: a rapid switching from one set definition to another, dissimilar one, without taking the time to adequately explore the earlier possibilities. Preliminary analysis is concerned with spotting either of these trends.

The five preliminary analysis subroutines are described below.

1. Phase and cluster analysis.

Together, these provide a quick analysis of the overall trend of a search. They can detect failure to converge as well as thrashing, in checking such values as: number of clusters formed, number of commands issued per cluster, number of commands per phase, number of phase changes, number of records per set within a cluster (i.e., tendency to converge within a cluster).

2. Zero set formation.

A zero set is one with no records in it. While zero sets are not always an indication of error, as in fine grain searching of patent files, an "excessive" number is an error indication. A student will be able to inform IIDA that his zero sets were intentional or knowing, not the result of error. However, the zero set checks can be valuable in detecting misuse of commands, lack of understanding of boolean logic or attempts to be overspecific in set description.

3. Extraneous and redundant commands.

Except for commands such as PRINT or DISPLAY SETS, any repetition of a command within a search is normally unnecessary. In a lengthy search it is not necessarily a serious problem, but over-repetition, especially the immediate repetition of a command, can indicate lack of understanding of the commands or of how to use IIDA's or DIALOG's facilities for reviewing past performance. A not uncommon error among beginners is simply to repeat a command that has led to an unwanted result, or to repeat a command that, because of communication delay or machine overload, has not yet been responded to. An example of an extraneous command is a SELECT

(descriptor) which produces a non-zero set, but whose resulting set is nonetheless never used. Again as an occacional thing this is no problem, but if done too much it is an indication of lack of planning on the part of the student.

4. /HELP and delay analysis.

/HELP and time use, when excessive, are indicators of searcher problems. Thresholds here are the number of calls for /HELP compared to the total number of commands, amount of time spent in /HELP and the amount of time between commands. When more time is spent using /HELP then doing searching, the student should probably go back to the tutorial material. In-depth analysis of the data on /HELP use will allow a reasonably accurate assessment of the kind of tutorial material needed by the student or perhaps recommend that the student call the proctor. Excessive time between commands can indicate that the student is unsure of what to do next, of that he is being distracted.

5. Error analysis.

Error analysis examines the number of syntactic or procedural errors and computes the total number of errors in the search, and the total per last n commands issued. Early in the search, the threshold will be set low and any repetition of an error will draw comment. Later, the threshold will be raised. It is the command parser that detects errors and conducts the initial conversation with the student about them. But the parser is limited to trying to get the error corrected. Error analysis becomes concerned when patterns of error repetition are detected.

2. Detailed analysis. Whenever a threshold is crossed, detailed, in-depth analyses are called. Whenever the detailed analysis program is called, some negotiation with the student will follow. The purpose of the detailed analysis is then to provide the detail for this conversation. The kinds of analyses performed are described below.

1. Phase and cluster analysis.

The detailed version of this program is given the complete history of the search and data on the related thresholds exceeded. If far enough into the search, measures of relatedness among clusters are computed and this data used to indicate the pattern of the student's use or lack of use of results of early clusters in forming later ones. The student will be asked to indicate, in natural language, why he has taken the course of action he has. This information serves two purposes: it makes him think carefully about what he was doing and, when read and analyzed by the proctor, provides information about student behavior that might not be available from the programmed calculations.

2. Zero set analysis.

This time a fine grained analysis is made of all zero sets, the command types and the search logic that led to them. If there is a pattern, the student is so informed (e.g. excessive / number of ANDs in a COMBINE command). If a single-descriptor SELECT results in a zero set, a check will be made on whether that descriptor has appeared on an EXPAND display. The student will be

reminded that he might have checked the term, that he seems to be developing a pattern of use of unchecked terms. He will also be given a list of the number of terms on a display centered on his zero-set descriptors that have m out of n letters in common. For example, if he uses SELECT COMPUTABILITY and gets a zero set, IIDA will see if the term COMPUTABILITY had been seen by this student on any EXPAND table. If not, the program issues an EXPAND COMPUTABILITY and counts the number of words appearing on the resultant table that have a given fraction of its letters in common with the search term. Clearly, letters in common do not imply equality of meaning, but the student will make that decision.

3. Extraneous and redundant commands.
This program classifies and sorts the extraneous or redundant commands. For example, it will point out repeated adjacent commands, failure to use the results of a particular type command, excessive use of a type command, etc.

4. /HELP and delay analysis. Data produced herein is for the proctor, not the student. If a student is using /HELP excessively or taking an unusual amount of time between commands, it will probably do no good to have the computer nag him about it. Data will be sent to the proctor who should then do his own analysis of the situation and communicate with the student. Data given to the proctor will include analysis of how and how often /HELP is used and in what situations, if any pattern can be found.

5. Error analysis. The preliminary error analysis program merely counts errors. The detailed analysis program classifies them and presents the result to the student.

3. Result of use. When a threshold has been crossed and the in-depth analysis has been run, the next step is negotiation with the student. The PAR is designed primarily to detect trends, not to determine the cause of errors, so the information from the in-depth analysis on which PAR based its decision is presented to the student or proctor for determination of cause. Some alternative remedies may be presented, depending on the problem. The student is then directed to formulate his solution to the problem and indicate to the program when he is satisfied. It is at this point that the student can tell the program that what the PAR has detected is not an error, but part of the student's strategy. The student evaluates his search, makes the changes he desires and goes back to his search.

In the case of the student who has no idea of how to proceed, the options of calling for further data and assistance from /HELP or calling the proctor are always open. PAR will be helpful and non-threatening to the student. It is not a panacea, however, nor is any attempt made for it to recognize every type of problem. It would hardly be cost effective even to try, so long as a proctor is available and his involvement costs little.

2.6.4 Measuring performance

We have previously made the point that we cannot measure the acquired searching skill of a student in any absolute sense. There is no numerically defined measure of how well one searches, hence no way to

measure the degree to which any given individual has progressed toward that goal. However, there are some measures that can be made of the relative performance of students by comparison with one another. It is important not to attempt to use such figures to assign grades to students or to measure on-the-job performance in industry. The measures can be meaningfully used to gain insight into aggregate student performances, sequencing of training material, and value of diagnostics. The following kinds of measures can be made.

1. Performance in tutorial material. A record will be kept of the courses taken by each student and the sequence in which they are taken. Performance can be measured by recording right and wrong answers to questions. It should be noted that we have often written that IIDA does not involve the concept of 'right' and 'wrong' answers, but that applies to exercise and assistance modes. The tutorial material is not to be written to serve as a test of knowledge or skill and it is important to realize this. However, knowing what kinds of errors students make is of some value in assessing how well they have mastered the material and where the tutorial material is weakest.

2. Performance in exercise mode. Records maintained here should indicate the kinds of errors detected and the manner of using the /HELP facilities. The exercise mode program will offer three levels of instruction: a very brief search designed to assure that the student can successfully negotiate it, a search using a minimally useful subset of commands, and a search using the full repertoire. Certainly in the second of these it is expected that errors will be made, as is the case with the new commands introduced in the full-language search. Thus, raw score is not too meaningful, but measures can be made of errors committed and subsequent behavior relative to that error, i.e. whether the student eventually stops making that type error. Since the emphasis in the exercise mode is to gain familiarity with the commands rather than to retrieve useful information, not too much should be inferred from the student's reaction to the data actually retrieved at the conclusion of the search.

3. Performance in assistance mode. When the student reaches this point in his training, a direct attempt to measure his success in the search is appropriate. Counts of errors made, time spent using diagnostics, number of commands and amount of time used can be useful, as will the student's own assessment of the relevance of the records he eventually retrieves.

Using evaluative information of the kinds outlined above, we will surely see differences among students. This can be combined with interview data taken from selected students, in which the interviewer attempts to ascertain the student's attitude toward his IIDA experience and the degree of success he feels he achieved.

2.7 Operation of IIDA

Each of the three major modes of use of IIDA -- tutorial, exercise and assistance -- has its own primary training objective. These are: (tutorial) to provide basic instruction on data base structure, search mechanics and strategy, (exercise) to provide experience in the use of the commands in carrying out a search, and (assistance) to assist users in the performance of actual searches. In practice, the objective the student may have and the benefits the student may derive will not be so cleanly separable. Also, the student may not view the change-over from

mode to mode so distinctly. Students can accomplish search objectives in exercise mode and can learn in assistance mode. They can invoke tutorial material in assistance mode when needed hence use tutorial material for reference while performing searches. The purpose of this section is to review IIDA facilities from the point of view of the student.

2.7.1 Modes of student use

The average novice user of IIDA should start his training with the data base structure course and also take the search mechanics and search strategy courses before proceeding to the exercise. The diagnostics course may be left for later. It will be possible, however, for a student to begin with any tutorial he wants, or even with the exercise mode program. A record will be kept of courses taken and their sequence. If a student shows signs of difficulty in handling the more advanced work, he will be referred back to tutorials not taken.

The search mechanics course will provide instruction on two levels: the minimal subset and the remainder of the commands. It will be possible for an inexperienced searcher to take only the subset mechanics and data base courses, and be ready for the first exercise within an hour.

To summarize, students will be permitted to start at the beginning of any course and proceed in any sequence. We will have a recommended sequence and we will be able to suggest, under certain circumstances, that the student now divert himself to a specific course in order to help solve a problem in searching.

Records of experience of students taking various paths through the teaching material will undoubtedly provide a basis for further refining our suggestions to new students on their choice of sequence.

It will not be necessary for a student to take all his work in a single sitting at the terminal. Once a student signs on, his record will be kept for a reasonable period and he may resume at any time.

2.7.2 Student records

For each student using IIDA, a record will be maintained of his use and performance. There will be a background record of each student, a record of performance in tutorial material, a detailed history of his performance in an exercise or assisted search, and summary information on exercises and searches previously completed. Specifically, the following information will be maintained:

1. Background data.
   - Student identification number M
   - Major subject, academic discipline, or profession
   - Time since bachelor's degree
   - Self-assessment of computer experience
   - Self-assessment of typing skill
   - Self-assessment of library use skill
   - Self-assessment of skill in on-line searching
   - Total amount of time spent in tutorial, exercise and assistance modes.

2. Tutorial record.
   - Course or course segment
   - Time and date begun
   - Time and date completed

Percent of questions correctly answered
Specific questions answered erroneously, with text of wrong answers (Note: While courses can only be begun at the beginning, the student may sign off temporarily, hence complete only a segment rather than the complete course. Also, individual frames may be invoked when a student is in exercise or assistance mode. Information on wrong answers is for use by IIDA staff in analyzing performance of the tutorial material, not the student.)

3. Exercise
   For each level of exercise, the complete set of history files will be maintained.
   For a completed exercise, a summary of each history file for analysis of performance of aggregates of students.

4. Assistance mode.
   Same data as exercise mode.

2.7.3 Privacy aspects of student records

There is no need or desire to collect or retain any information of a private nature about users of IIDA. However, because of the great amount of sensitivity on the subject of privacy and because the initial use of IIDA will be in conjunction with class assignments, several positive steps will be taken to assure privacy and to assure students that their privacy is being maintained. These steps are:

1. A positive statement will be made to the student, through his terminal, at the start of his first encounter with IIDA that: (1) A record of his performance is being maintained for research purposes, (2) This record will in no way become a part of his academic record nor will it be used to determine, in any part, his grade in the course through which he is using IIDA, (3) These records will not be disclosed to his instructor, and (4) His name is not a part of the record; instead his record identifies him only by a number assigned by the IIDA staff solely for this purpose (not related to employee, social security or student numbers, for example).

2. The only record of a student's name will be a master list, not stored in any computer, that relates students' names and identification numbers. Such a list is necessary to assign numbers and to remind students who forget them (a common occurrence).

3. No identification of individual students, even by number, will be made in any published record of IIDA activities or accomplishments. Records will be used only to study factors affecting performance and the only reason ever to publish part of an individual record would be to illustrate a particular behavioral pattern. Identification would not then be necessary.

2.7.4 Proctor.

The proctor is an IIDA staff member familiar with the various teaching programs, data base searching, and the use of data communi-

cations terminals. The reason for having a proctor is to have an experienced person on call when unexpected problems arise or the computer is unable to resolve an issue. An example of the need for a proctor is when the student is having trouble using the terminal and is unable to send a meaningful message. Then, either the proctor can see that the student is having a problem and call him or the student can call the proctor on a voice telephone line.

The proctor will also serve to collect data for revision of IIDA. Again, when unforseen problems arise the human proctor may be the only means by which the nature of the problem can be communicated.

The proctor will have the following capabilities:

1. Student monitoring. The proctor will be able to request the IIDA computer to send to his terminal a copy of all messages into and out of a specified student's terminal. Thus, he will be able to monitor the student's work. Whenever a student is being monitored, he will be so informed by a message from the computer.

2. Retrieve student performance data. By use of the /HELP command the proctor will be enabled to browse through the history records of any student currently conducting a search. He will have access to the same data the student has, except that the proctor can see the student's /HELP history file, but this is not offered to the student.

3. Retrieve summary data. The proctor will be able to invoke previously written programs that will provide summary information about all students. For example, he may ask for a report which includes a statement of how many students gave the wrong answer to question x in a tutorial, or how often students make syntactic errors in PRINT commands. The proctor will not have a program that enables him to ask ad hoc questions of any file in the system. This is simply too expensive a facility to provide.

4. Typed messages. Students will be able to type messages for transmittal to the proctor's terminal, whether or not they are being monitored, and the proctor will be able to send a message to any student terminal. This capability can be used for asking and answering questions when the terminal is running and the student needs advice or the proctor feels he must tell something to the student.

5. Voice telephone. The proctor will have a voice telephone line separate from the one used for his terminal. Students may or may not have such an extra line at their locations. At any rate, if the student can get to a telephone or can disconnect his digital terminal, he can get to the proctor directly, by voice. There will be no cost to the student for this service. In the future, when IIDA reaches a national audience, this free service should be continued as it is by the major search service contractors.

Use of proctor assistance by a student will be considered a use of /HELP and will be logged in the /HELP history file.

3. COMPUTER AND DATA COMMUNICATIONS SUBSYSTEM

3.1 Overview of the Computer System

Three major computing functions are performed within the IIDA system: data base processing, instructional processing and communications processing. The computing system that performs each of these functions is called a processor which term denotes a program running in a computer. The three processors, in concept, could be all implemented in one physical computer or in three separate ones. In the IIDA model (See Section 5) all three were implemented within a single computer. In the full IIDA system there will be two physical computers, one operating the data base processor and one operating the instructional and communications processors.

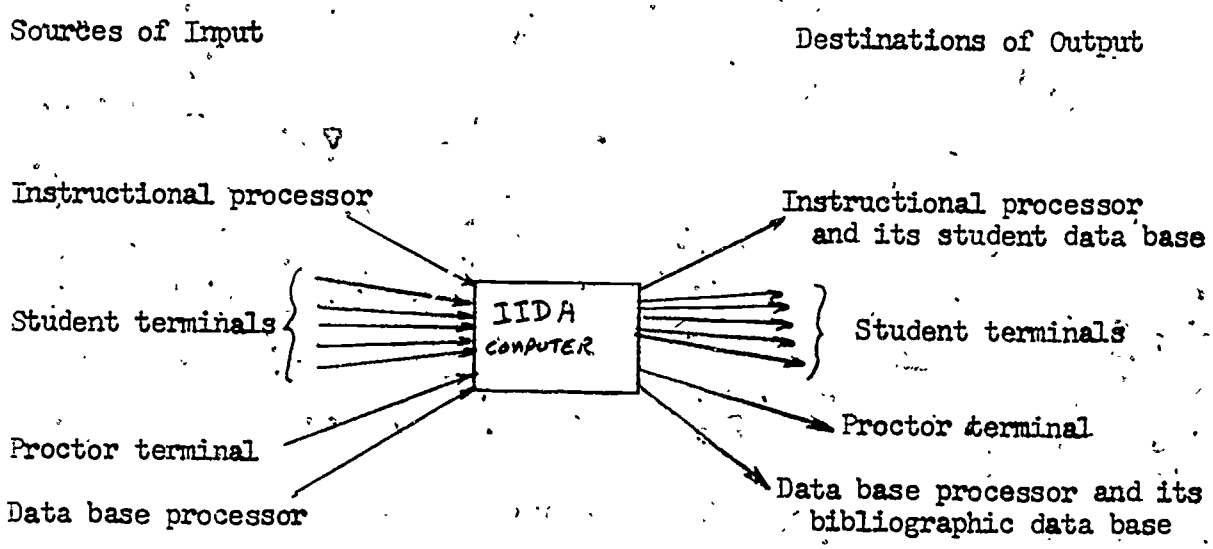
The data base processor (DBP) performs information retrieval functions and has access to the bibliographic data bases, including Compendex. Initially, there will be a single data base processor, that operated by Lockheed for its DIALOG search service. Eventually, it will be possible for IIDA to communicate with other data base processors. IIDA students will use exactly the same service that is available to other, direct users of the DIALOG service. That is, they will have the full, up to data data bases and they will even face problems of slow response when the data base processor is overloaded.

The instructional processor (IP) is a set of programs which administer the computer-assisted instruction or tutorial training, the exercise and assistance modes, and provide the diagnostic programs for use by these programs and the students. The instructional processor deals only with messages given it by other programs, and provides messages to be delivered to students or the data base processor. It is not concerned with the mechanics of message receipt or transmittal.

Because the flow of messages can sometimes be complex, all functions related to the handling of messages are combined and together comprise the communications processor (CP). It serves as a central communications facility receiving messages from the other processors and student terminals and delivering them to the appropriate addressee, which is one of the other processors or student or proctor terminals. The communications processor also maintains a log of all message traffic, serves as a concentrator to allow more than one student to operate on a single line between the CP and the DBP, supports the proctor and computes the charges for all users.

3.1.1 General flow of data

The data flow is illustrated in Figure 3.1. Inputs may originate with student terminals, the proctor terminal, the instructional processor or the data base processor. All pass through the CP wherein they are tagged as to origin and destination and put in the appropriate queue for delivery



Functions of the CP

Message switching
Message storing and forwarding
Message logging
Concentrating
Billing
Proctor support

Figure 3.1. General flow of data in the IIDA system.

to addressees. There may be more than one addressee when, for example, the proctor monitors a student's performance, hence both receive DBP output. A log is kept of all communications transactions (arrival, enqueuing, dequeuing, transmittal) for later use in analyzing traffic patterns and delays. Messages to and from the DBP may be modified by the concentrator. Finally, costs are accrued for each student using the system. Except for the fact that the concentrator must parse student commands as does the IP, in order to be able to modify them, there are no functions in common to the IP and CP. The CP should be "transparent" to users, i.e. they should be unaware of its existence.

3.1.2. Hardware configuration

The hardware configuration used as the basis for design is the smallest Digital Equipment Corporation PDP-11 that could handle the IIDA job for the forseeable future. Other computers can do the job and we do not wish to foreclose on this decision until the last possible minute. The PDP-11 line was selected because of our early discussions with the National Bureau of Standards, which had done some related work on such a machine and had oriented our thinking in that direction. Also, similar applications are operated from approximately this configuration at the National Library of Medicine, Medical College of Pennsylvania and, formerly the University of Delaware. At least one larger machine, the PDP-20, probably can do the job with less than its full capacity, although at this time the PDP-11 and PDP-20 are not software compatible. A new IBM computer, the Series 1, looks promising also. This was announced by IBM after most of the IIDA design decisions had been made, but its low price may make a switch worth the effort.

The configuration assumed in the design of the softwar is the following:

PDP-11/34, with 96,000 bytes of main memory
2 magnetic disk units, of 5 million byte capacity each (one largely taken up with operating system software). Disks are dismountable.
A 16-channel multiplexer to which will be attached one high-speed terminal, either CRT or printer, operating at at least 120 characters per second and up to 15 30-cps dial-up lines.

The programming language will be BASIC-PLUS and the operating system system RSTS/E.

3.1.3. Software configuration

We have already noted that the data base processor is resident in a computer external to IIDA. IIDA communicates with it but plays no role in its operation. The IIDA computer contains the instructional and communications processors. These consist of the following software components, respectively.

1. Instructional processor. From the student point of view the IP consists of the tutorial courses, the exercise and the assistant mode. These are supported, to various degrees, by a HELP program and a series of diagnostic programs. The courses, exercise and assistant mode, and the HELP program, redesigned around the concept of a shell, a program which serves primarily as sequencing logic among a set of subroutines. Thus, the tutorial shell

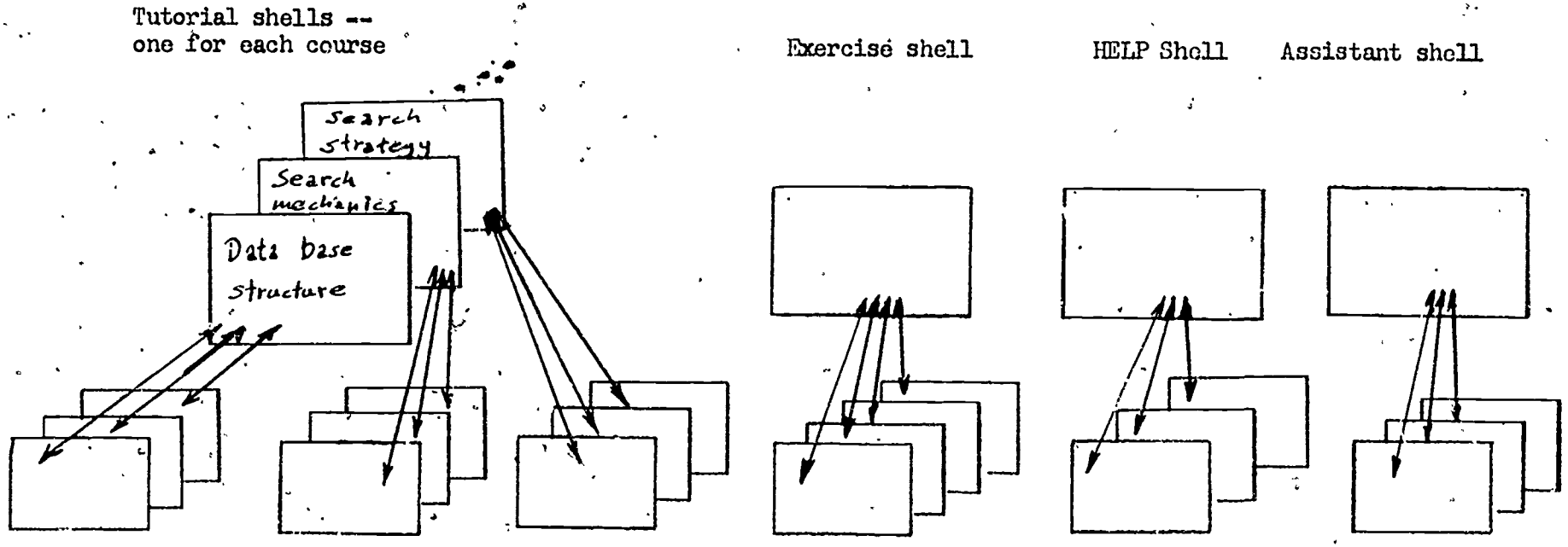


Figure 3.2. General software configuration.

sequences the student from one frame of training material to another, a frame being the unit of presentation of tutorial material. The exercise sequences the student from one frame to another, also. In this case, the frame is the unit of programming required to elicit a command from a student. HELP does its sequencing by presenting the student a series of Menus, lists of choices of options open to him. Any choice might result in another menu offering even more details of the choice just made. The diagnostic programs are not bound to a shell. They are called, as needed, by frames in other programs. The relationship of shells to subroutines is shown in Figure 3.2.

2. Communications processor. This set of programs consists of the elements previously noted, in Figure 3.1: message switching, store and forward, logging, concentrating, billing and proctor support.

3.1.4. Logical processor

Common memory and internal channels are two distinctive features of BASIC-Plus under the RSTS/E operating system. Both place constraints on and provide capabilities to the IIDA system. A short general discussion of these features follows. Their application in IIDA is developed further in this chapter, expecially Section 3.6.

1. Common memory. Sixteen bytes in the main memory are available for reading and writing by all programs in operation. The bits of this common memory area are used for rudimentary interprogram communication.

2. Internal channels. Up to a maximum of twelve data files may be assigned to any legal internal I/O device. In the IIDA system, these channels are exclusively assigned to files on the disk. Files may be shared by programs. The same file may be read and written by different programs, if the protection code allows it. Only the program which first opens the file, however, may extend the file by writing to it. For this reason, communications files (see Section 3.6.1.) must come in input-output pairs. Writing extension privileges for the output file are held by the instructional program. Extension privileges for the input file are held by the message switching program.

3.2. The Instructional Processor

The instructional processor comprises all the programs which teach the student or diagnose or monitor him. We have designed the IP as a set of "shells" or programs whose function it is to control or guide the sequence of other programs. There are five such shells: the tutorial, exercise, assistance, HELP and user control shells. The diagnostic programs, which make up a good bit of the programming under each shell, are discussed in Section 3.3.

3.2.1. The Tutorial Shell

Tutorial programs are CAI programs. Each consists of a set of frames. A frame is, in our usage, the program that presents a unit of tutorial information, elicits a response to it from the student, and performs an analysis of that response. There can be branching within a frame if, for

example, a student gives an incorrect but anticipated reply and the program author simply wants him to try again, the shell then serves to determine the sequencing among frames.

In a CAI program, a shell is quite simple. For each frame there is a limited set of possible next frames and the choice is made within the frame. The shell has only to call the next frame, as a unit of programming, and branch to it. If a frame leads to the decision that a student has given too many unrecognizable answers, or that the program of the frame cannot understand the replies, the shell may branch to a generalized routine for handling unrecognizable replies.

In this way, the CAI course author is given the responsiblity to decide what frame to execute next, depending on the student's reply to his question, but not for any of the programming logic necessary to invoke that next frame.

One advantage of this approach to programming is that every frame can be operated as a subroutine, and in any sequence. Thus, the HELP program can make use of tutorial frames in providing explanations of commands and reviews of specific subjects can be assembled from existing frames.

It is our intention to allow a student to begin his IIDA instruction almost anywhere and to have maximum freedom to traverse the material as he wishes. The shell concept, by making each frame a separate unit of programming, assures that students can begin almost anywhere. There will be some frames which make no sense as starting points, but the student can be given a table of contents, which are effectively entry points, and allowed to choose for himself.

3.2.2. The Exercise Shell

This shell has the greatest number of control options, hence decisions, of the five basic shells. It also makes use of frames, but this time not units for which there is a right or wrong student reply. Instead, there are acceptable and unacceptable replies and there may be any number of acceptable ones.

The basic logic governing the exercise is the three-phase model of a search. Students will be started in Phase 1, in which they perform index search type commands: simple SELECTS and EXPANDS.

There will be three different levels of exercise shell, one to take a student through a "canned" search in which he is told what to search for and how, and is given the experience of doing it. The second permits him to do a search of his own, using a limited subset of the command language. This will be similar to the exercise in the IIDA model. The third shell works with the full language. In all cases, the exercise shells are built around the concept of a search phases, and they expect the student to begin with phase 1 and work through to phase 3.

A reply, within a frame in an exercise, is a command. In most cases, the student may enter any number of commands, so long as they are valid. An excess of invalid commands may be treated essentially as a sequence of unrecognizable replies in the CAI mode. Progression from frame to frame

Frame No.	Frame topic	Sections	Next programmed frame
1.	BEGIN		2
2.	PHASE 1	SELECT EXPAND	3
3.	PHASE 2	COMBINE	4,2,1
4.	PHASE 3	PRINT	5,3,2,1
5.	LOGOFF	-	-
6.	CONTROL COMMANDS	one for each cmd.	Return to calling frame
7.	HELP	one for each menu	Return to calling frame or go to frame of choice
8.	ADMINISTRATIVE	-	-

Figure 3.3. Shell structure for the limited language subset version of the exercise.

Frame No.	Frame Topic	Section	Next Programmed Frame
1.	BEGIN		2
2.	PROCESS	Elicit command Parse Response analysis performance analysis	1, 3
3.	LOGOFF		-
4.	CONTROL COMMANDS	One for each command	Return to calling frame or go to frame of user's choice
5.	OTHER ADMINISTRATIVE		

Figure 3.4. Shell structure for assistance mode.

requires completing a use of each of the commands defined for that exercise level. After the student reaches and completes phase 3, he may return to any earlier phase of his choosing.

A student may make use of what amount to phases 4 and 5 if he uses an administrative command or a request for information or control. Thus, HELP is a phase 5 command, valid at any time. BEGIN is a phase 4 command, valid in the exercise mode only before commencing phase 1.

The shell for the exercise in the model consisted of 11 frames, and probably not much more will be required in the full implementation of any level of exercise shell. Each frame elicits a series of commands of a given class from the student. Analysis of the commands takes place within the frame. Transition from frame to frame takes place upon completion of all phase requirements, with the understanding that transition to phases 4 or 5 may take place at any time, or is dependent upon the specific command (LOGOFF is not valid as a first command, etc.)

Figure 3.3 illustrates the basic set of frames to be used in an exercise and the connecting logic needed.

3.2.3. The Assistance Shell

This shell, although it drives the most important of the instructional programs, has the simplest structure. There is no prescribed order of commands in assistance mode, except those few imposed by the logic of the DBP: that, for example, it is meaningless to PRINT before having defined any sets, or to SELECT E6 if there has been no EXPAND command defining a term corresponding to E6. The diagnostic programs will detect and comment upon such misuses. Thus, the assistance shell will accept any input, have it parsed and results posted to the performance history files. The performance analysis routine examines student performance, looking for indications of unacceptable or unproductive behaviour. PAR serves as a sort of after the fact frame sequencer, consistent or meaningful in light of what happened previously.

In operation, the assistance shell calls the frame which elicits a command. That frame calls the parser as a subroutine and the parser passes the command, if valid, to the CP for transmittal to the DBP. When the DBP response is received, the assistance shell calls a frame which calls updating program that posts results of the command to the performance files. Then, the shell calls the performance analysis routine. This sequence of events is illustrated in Figure 3.4.

3.2.4. The /HELP Shell

/HELP is a legitimate command at any time, for an IIDA user. There are many different forms of help available. The /HELP shell is invoked from another shell whenever the student calls for this facility. The/HELP shell consists of the programs needed to present to the student information about what is available and to ask him what he wants. Providing the help, then, falls within a frame of the /HELP shell. This structure is illustrated in Figure 3.5.

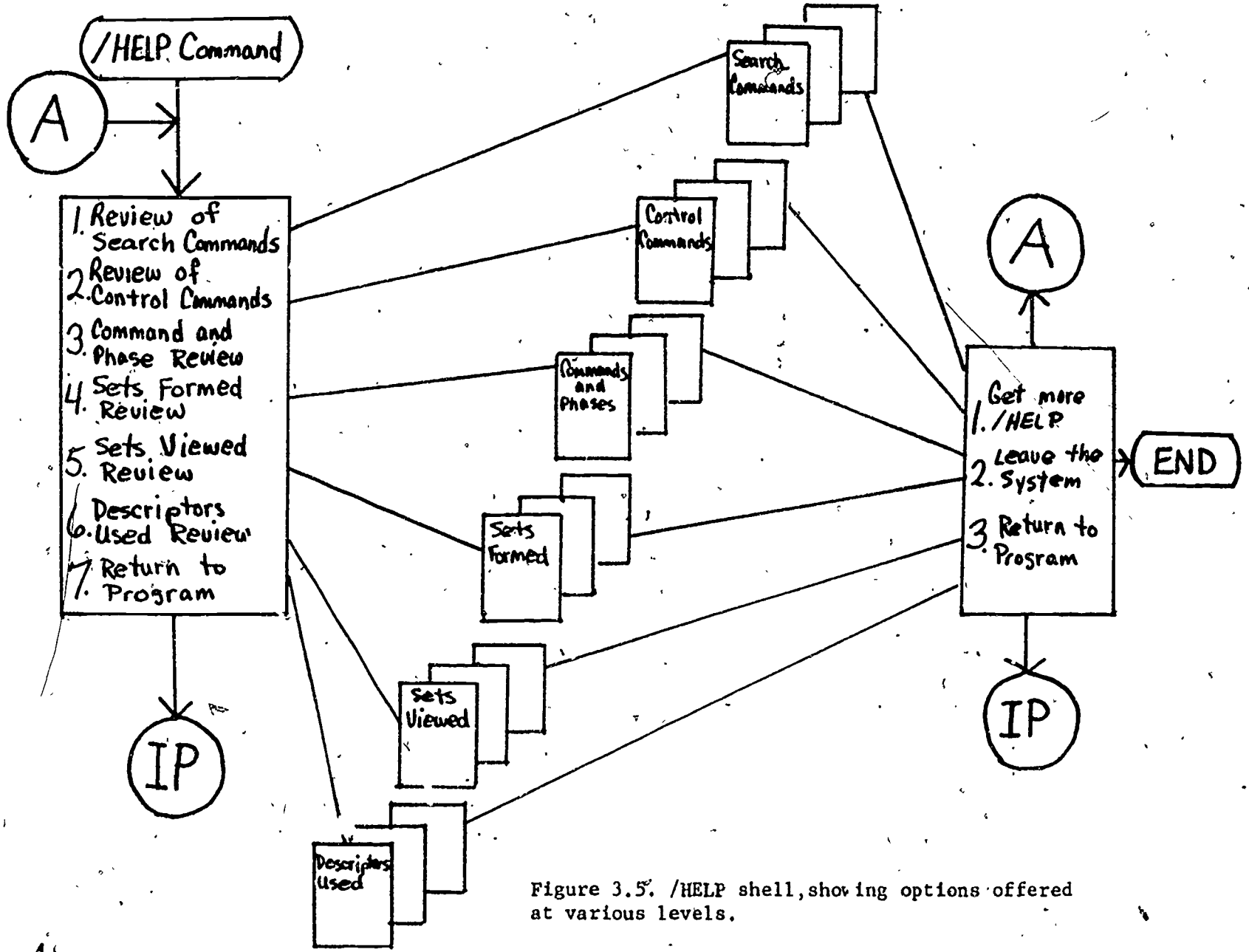


Figure 3.5. /HELP shell, showing options offered at various levels.

The basic shell presents to the student a "Tree" of menus, i.e. a series of lists of features available. Each time he selects one, a more detailed list is provided, until he has finally made an action decision, i.e. one calling for some substantive information to be provided, a diagnosis run, or a change in sequence of the program.

3.2.5. User Control Shell

As described in Section 2.4 there is a set of controls by which the student can to some extent modify the sequence of IIDA operations. These commands are directed solely at IIDA, not the data base processor. T. assure that they are not mistaken for DBP commands, each is preceded by the character "/". For each command there is a program, or set of frames, designed to implement the command. All that is required of a user control shell is that, upon recognition of the issuance of a user control command, program control be transferred to the appropriate frame.

The user control shell, therefore, has very little to do. A shell is created in anticipation of future expansion in the number and complexity of user control functions.

3.3 Diagnostic Programs

Diagnostic programs are invoked, in general, by program frames within any of the instructional programs. Their purpose is to analyze both the most recent student input and the entire pattern of this and previous inputs. Analysis is done at three levels: diagnosis of the current student input, micro-analysis and performance analysis. (See Section 2.6.3).

Analysis of the current input is done by two programs: the first looks for control commands, those beginning with the character "/". If an input is not one of these, it is assumed to be a DIALOG command and is passed to the command parser which checks on its procedural and syntactic correctness (Section 3.3.1) and interacts with the student if the command is in any way unacceptable. Following the input of an acceptable DIALOG command, the student's performance history files are updated to show the new command and its results and this may involve some microanalysis. The microanalysis programs operate on raw data extracted from the student's command to produce other information that is entered into the performance history for use by the PAR. These programs create new data elements. They do not make decisions.

Next, quick checks are made on the performance history files for signs of strategic problems, i.e. patterns of commands that appear not to be leading toward a successful conclusion of the search, or for patterns of error that the student should be made aware of.

When the preliminary threshold checks indicate the existence of a problem, and occasionally even without such indications, more detailed pattern analyses are made of student performance.

The command parser is described in Section 3.3.1. The student performance data base is described in Section 3.3.3.

3.3.1. The Command Parser

The Parser will verify the syntactic and procedural acceptability of commands entered by students, identify the nature of errors whenever possible, and provide information to the student or a calling program on the nature of any errors detected. In its most common use the parser will converse with the student about a command error and try to elicit from him a correct form.

The parser will detect errors of syntax, i.e. commands which do not follow the rules of a construction of the search language, and errors of procedure, i.e. attempts to use commands which, for tutorial purposes, are not acceptable at the time, even if correct in syntax. The parser is a subroutine. The program that calls it will provide it with information on what commands are procedurally acceptable at the time of calling (including the possiblity that all commands are acceptable) and will specify the level of analysis desired on the command's argument. When used by a tutorial program, a limited form of analysis is needed because the tutoring program can normally make an exact match comparison on the argument, while in assistance mode, there is no way to anticipate what the argument should have been. Therefore, a tutoring program will ask for a minimal amount of argument analysis but the assistance program will ask for the maximum amount. The levels are:

1. No analysis. The parser will return to the calling program the text of the argument, for its use in making comparison.
2. Blank extraction. The parser will remove all blanks from an argument and return to the calling program: the original text, the blank-extracted text, and the number of blanks extracted. A calling program would likely find the blankless form of the argument easier to test against a standard.
3. The parser will completely parse the argument and if no error is found post the components of the argument to the various performance history tables. If an error is found, the error text is posted to the error file and an error indicator is turned on.

1. Overall Logic Description. All DIALOG commands begin with a verb, an abbreviation, character representing a verb, or a period followed by a verb. Most then have an argument, such as the descriptor following a SELECT command. But not all command verbs have an argument, e.g. LOGOFF. The Parser identifies the verb first and then the argument. If a valid verb cannot be identified, no attempt is made to parse the argument. When an error is discovered, it is often not possible to be certain that what the parser detects as an error was the actual error. See the example in Section 2.6 of how an error on the part of the student may lead to an ambiguous situation.

In general, there are two principles which guided the design of this program: First, it is not possible to resolve all ambiguity in interpreting a student's input; however detailed the analysis, the program cannot know what was in the student's mind nor can it know how many mistakes may have been made in a single input. Second, IIDA is an on-line system that must provide rapid response to the user. It cannot take too much time to execute any of its many programs. Hence, parsing is done on this basis: first a command is recognized and if one is found the argument is parsed. In parsing the argument, a quick decision is made as to which form of argument this is

(several commands, notably SELECT, have more than one form of argument) and any errors or inconsistencies are interpreted in the light of that assumption. In the worst case, this could be somewhat confusing to a student but it cannot lead to error. In other words, it could lead to the Parser telling a student there is an error in his type a argument, while he actually intended a type b argument, but made enough errors to make it look like type a. But the parser will always tell the student what assumption it made and erroneous assumptions on its part can be quickly overriden by the student. We do not attempt to use the argument to help identify the command entered, nor do we use any evidence other than that involved in the first cursory test to determine what form of argument was attempted. Frequent and rapid communication with the student can overcome these lacks and the saving in delay response can be great.

Identifying the verb. The logic of verb recognition must follow the construction of the language and the rules used by DIALOG in its parsing. For example, if DIALOG finds that the first n characters of a command constitute a legal verb, it assumes that to be the verb even though there may be other interpretations of the command, as in SELECTION. This command will be interpreted as SELECT ION, rather than S ELECTION, both of which are correct forms.

The IIDA parser examines the first character of the string of characters constituting the student input. Each initial character delimits the number of complete command names that need be searched for, and if the initial character is not a valid first character (i.e. not the initial character of a valid command) the parse can end there. For each initial character, the commands beginning with that letter are scanned for. If the full verb is not found, then if the character is a command abbreviation, the parser assumes that is what was intended. If the first character was not, itself, a valid abbreviation and no full verb can be found, then there is no valid verb and the parsing halts with an error indication.

In scanning for either verb recognition the parser is guided by input from its calling program. Procedural restrictions are placed upon commands by the calling program telling the parser what commands are acceptable at this point in a search. A student may, then, enter a syntactically valid command which is unacceptable because it was out of phase, or for some other reason. No procedural restrictions may be placed upon the argument through this means -- only the command verb. Argument checking is done by the calling program.

2. Command Recognizer. Specifically, the following steps are taken in recognizing the command verb and parsing the argument:

1. Scan the entire string (as received from the student, consisting of the command and argument). If there are blanks preceding the first non-blank character, shift the entire string left the number of blanks that occur. If there are blanks following the last non-blank character (excluding the carriage return or end of message indicator), delete them. Make a copy of the remaining string that has had all internal blanks removed possible for use by the tutorial programs.

2. Scan the string and record the position of all occurrences of the following characters: = (,) - , ?.

Command Verbs	Abbreviations	Symbols	
BEGIN	none	!	(not used in IIDA)
COMBINE	C	$	"
DISPLAY	D	%	"
DISPLAY SETS	DS	@	"
END	none	=	"
END/SAVE	none	=/SAVE	"
END/SDI	none	=/SDI	"
EXPAND	E	"	"
EXPLAIN	none	?	
KEEP	K	(	(not used in IIDA)
LIMIT	L	)	"
LIMITALL	LALL	)ALL	"
LOGOFF	none	none	
PAGE	P	0	(not used in IIDA)
PRINT	PR	&	"
SELECT	S	#	"
TYPE	T	'	"
.EXECUTE	none	none	
.FILE	"	"	
.RECALL	"	"	
.RELEASE	"	"	
.SORT	"	"	

NOTE: The notation not used in IIDA refers to the single-character abbreviation, not the command or its abbreviation.

Figure 3.6 Alphabetic list of DIALOG command verbs, abbreviations and symbols.

3. Perform command recognition. The result of this subroutine is the return of: an indicator showing whether or not the command verb was valid, an indicator showing whether or not the verb was immediately followed by a blank, the command or its abbreviation or symbol, whichever was used by the student.

4. Perform an argument scan. There will be a different scan program for each command verb. If a command that should not have an argument has one, the student will be informed that IIDA is assuming the command entered and asked if the assumption is correct. He has the chance either to cancel out the needless argument or to change to command. In general, they will:

a. If the command is valid, tell the student the assumed source of the error, post the nature of the error to the error history, count the number of times an error has occured on this command, and elicit a new input from the student.

b. If the command verb is valid but no blank follows it, tell the student that the command is being assumed and offer the opportunity for him to reenter the command if the assumption is wrong. Suggest he use a blank separator in the future.

c. If the argument is blank or null, i.e. missing, reject it, even though DIALOG will accept it. Allow him to override this rejection.

d. If the command verb is valid but the argument is not, inform the student of the nature of the error, post the error to the error history file and elicit a new command, counting the number of times this is done.

e. If more than a to-be-determined number of tries is necessary, invoke the PAR.

f. If command verb and argument are valid, post the elements to the appropriate history files, set an indicator to show valid command, and exit from the parser.

3. Program Logic. To illustrate the program logic, the following figures show:

Figure 3.6 the list of DIALOG command verbs, their abbreviations and symbols.

Figure 3.7 the recognition logic for the first character of the command string.

Figure 3.8 the illustrative logic for one particular initial character, E, which could lead to any of several command verbs.

Figure 3.9 the Backus-Naur formal representation of the argument of a SELECT command, which is a formal expression of the ways in which a valid SELECT argument can occur.

Figure 3.10 A flow chart of the parsing logic for the SELECT argument

In Figure 3.6 the complete list of command verbs is shown. Some of these have an argument and some do not. In several cases, it is arbitrary whether two words are combined as a verb or one word is the argument of another, such

C1 = C				Y								N
C1 = D					Y							N
C1 = E						Y						N
C1 = K							Y					N
C1 = L								Y				N
C1 = P									Y			N
C1 = S										Y		N
C1 = T											Y	N
C1 = .												Y
C1 = ?												N
GO TO TABLE FOR C1 = C				X								
GO TO TABLE FOR " = D					X							
"						X						
"			K				X					
"			L					X				
"			P						X			
"			S							X		
"			T								X	
"			.									X
"			?									
COMMAND ERRO.												X

Figure 3.7. Decision table for analysis of first character of command string.

C1 = E				
C1 - C6 = EXPAND	Y			N
C1 - C7 = EXPLAIN		Y		N
C1 - C3 = END			Y	N
C1 - C8 = END/SAVE				Y N
C1 - C7 = END/SDI				
COMMAND = EXPAND	X			X
COMMAND = EXPLAIN		X		
COMMAND = END			X	
COMMAND = END/SAVE				X
COMMAND = END/SDI				

Figure 3.8. Decision table for recognition of command verb on assumption that first character (C1) = 'E'. If no full verb name is found, the command is assumed to be EXPAND, entered in abbreviated form as E.

as DISPLAY SETS, which defined by Lockheed as a separate command with no argument, but could be interpreted as a DISPLAY command with the argument SETS. DIALOG recognizes the (usually) single-character symbo_, a hold-over from the early RECON system. Because these are not mnemonic we do not plan to teach them, and will not recognize them except for the ? (EXPLAIN). It can be seen that the first character often, but not always, identifies the command verb. In all cases, however, testing of the first character reduces the number of further tests that need to be made.

Figure 3.7 is in the form of a decision table. It shows what decision is made from the first character. The decision is always one to branch to a next level of testing unless none of the accepted first characters occurs. In that case, the command is immediately declared invalid.

Figure 3.8 is a decision table illustrating the logic for secondary testing after the first character has been found to be E. Such a command could be any of the five listed. If none of them, it is assumed that the command was abbreviated by the E and thus treated as an EXPAND command.

4. Argument Parsing Each type argument for each type comand must be formally defined as to acceptable syntax. Form includes both syntactic structure and in some cases content. For example while nearly any string of characters is acceptable as a descriptor, the number of descriptor suffixes or tags is quite limited. An illegal suffix or tag invalidates the argument, as does such an error as omitting a / in the argument of a PRINT command. For example, PRINT 3 2/1-4 is invalid because there is no separator between the 3 which identifies a set number and the 2 which identifies a format. While separation by a space rather than a / seems acceptable, without the space there is no way to know whether the student intended PRINT 3/2/1-4 or PRINT 32/?/1-4, etc. Rather than guess or dwell overly long on what might have gone wrong, as soon as an error is detected, the command is sent back to the student.

Figure 3.9 shows the formal definition of the SELECT arguments. We have chosen this for illustration because it is more complex than most. There are the following types of arguments for a SELECT:

EXSTR (Expand string) which denotes one or more line numbers (EXEL-Expand element) from an expand display. These are of the form Enn or Rnn and may be connected by commas or hyphens.

FREE (Free text) which connotes the form used when searching the text of a field for co-occurrences of terms. This configuration always has a set of parentheses embedded in it and within them must be any of a limited number of link codes.

SUFF (descriptor with a suffix) which has the form: descriptor/SU where SU can be any of a limited number of suffix codes. This form of descriptor may be used at the end of a Free argument.

PREF (Prefix) which has the form: TG=descriptor. The = must be in position three and the TG may be any of a limited number of tags.

STRING (any string of characters not one of the above) A string may terminate in ?, implying it is truncated (TRUN).

[1] <SELECT phrase> :: = SELECT <sarg> | S <sarg>

[2] <sarg> :: = <string> | <pref> = <string> | <desc>/suff |
<free> | <ex str>

[3] <string> :: = desc | desc ?

[4] <pref> :: = AU |...

[5] <suff> :: = CS |... {<suff>,} <suff> TI|DE|ID|CS|AB

[6] <free> :: = {desc (<link>)} desc | <free> /<suff>

[7] <link> :: = W|L|S <int> W |...

[8] <ex str> :: = <exel> | {<exel>,} <exel>

[9] <exel> :: = E<int> | E<int> - E<int> | R<int> |
R<int> - R<int>

<int> - Note that certain limits apply,
e.g. in E<int>, <int> may not
exceed largest number in EXPAND
table, E<int_1> - E<int_2>,
$int_1 > int_2$

Terminal Symbols
Primitives = desc, <int>
Capitals and other symbols are literals.

Figure 3.9. Backus-Naur representation of the syntax of the SELECT command.

IIDA treats an argument of the FREE type or EXSTR, if it uses more than one line number, as a phase two command, one concerned with combining terms not just defining or searching our individual terms.

Figure 3.10 shows the overall flow chart for the argument analysis following a determination that the command verb has been SELECT or S.

First, test for the EXEL or EXSTR type argument. EXEL and EXSTR, of course require that one or more EXPAND commands have been given. If at least one valid EXEL is found, then assume this is the form intended, treat all subsequent errors in the argument as deviations from this form.

Next, test for FREE. If the argument is not EXEL or EXSTR and if there is at least one pair of parentheses with non-blank characters on either side, treat this as a FREE and any errors as deviations from FREE.

Next, test for SUFF. If/ occurs in the third position from the end of the string, check for valid suffixes. If it occurs in any other position, treat as an error in trying to form a valid SUFF argument.

Next, test for PREF. If the character = occurs in position three, check for valid tag. If = occurs in some other position, treat it as an error in trying to form a PREF argument.

Next, test for a truncated descriptor. If the character ? occurs anywhere except the terminal position (if first, it would have been deemed a verb) treat as a possible error.

Any other character string is accepted as a valid descriptor, except an all-blank argument.

3.3.2. Performance Data Recording

Sources of the data written to the student performance file include : (1) the command parser which evaluates student entries, (2) response parser programs which capture significant elements from data base responses and (3) expansion routines which "explode" codified data into more useful form and (4) output of microanalysis and PAR programs.

The command parser and its resulting output were discussed in Section 3.3.1.

The response parser is a set of programs which are called to examine a data base processor response to a student command. The response error detection routine is called automatically. Its purpose is to identify error messages or changes of state (e.g. the system going down) from the data base machine.

The other response parsing routines are invoked to extract data from responses to a specified group of commands. The SELECT, COMBINE and LIMIT command responses provide information needed to update history files. The EXPAND and RELATE commands elicit tables referenced by various other programs.

The expansion routines are called primarily to translate truncated and E- and R- series arguments into more meaningful formats for analysis

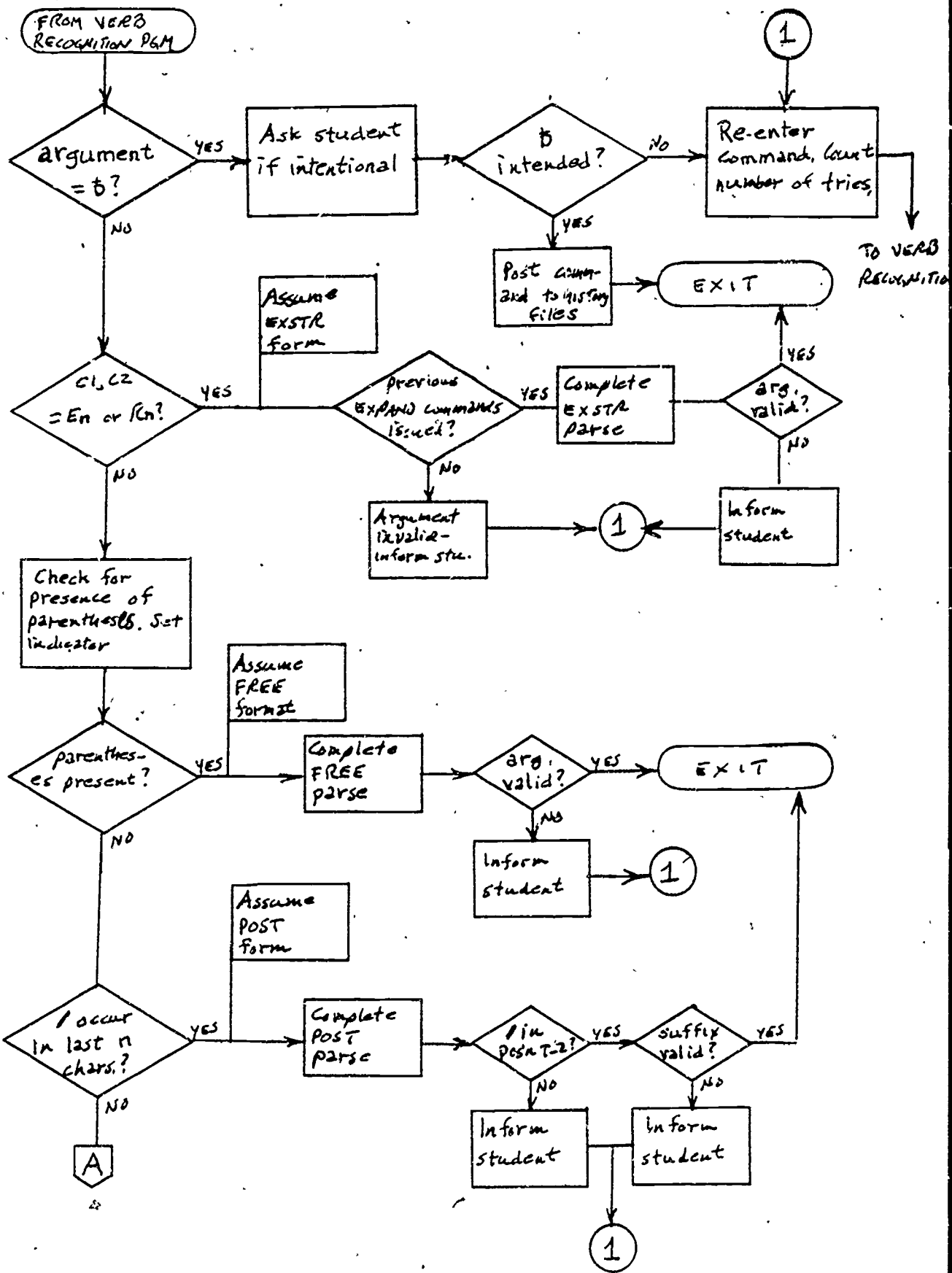
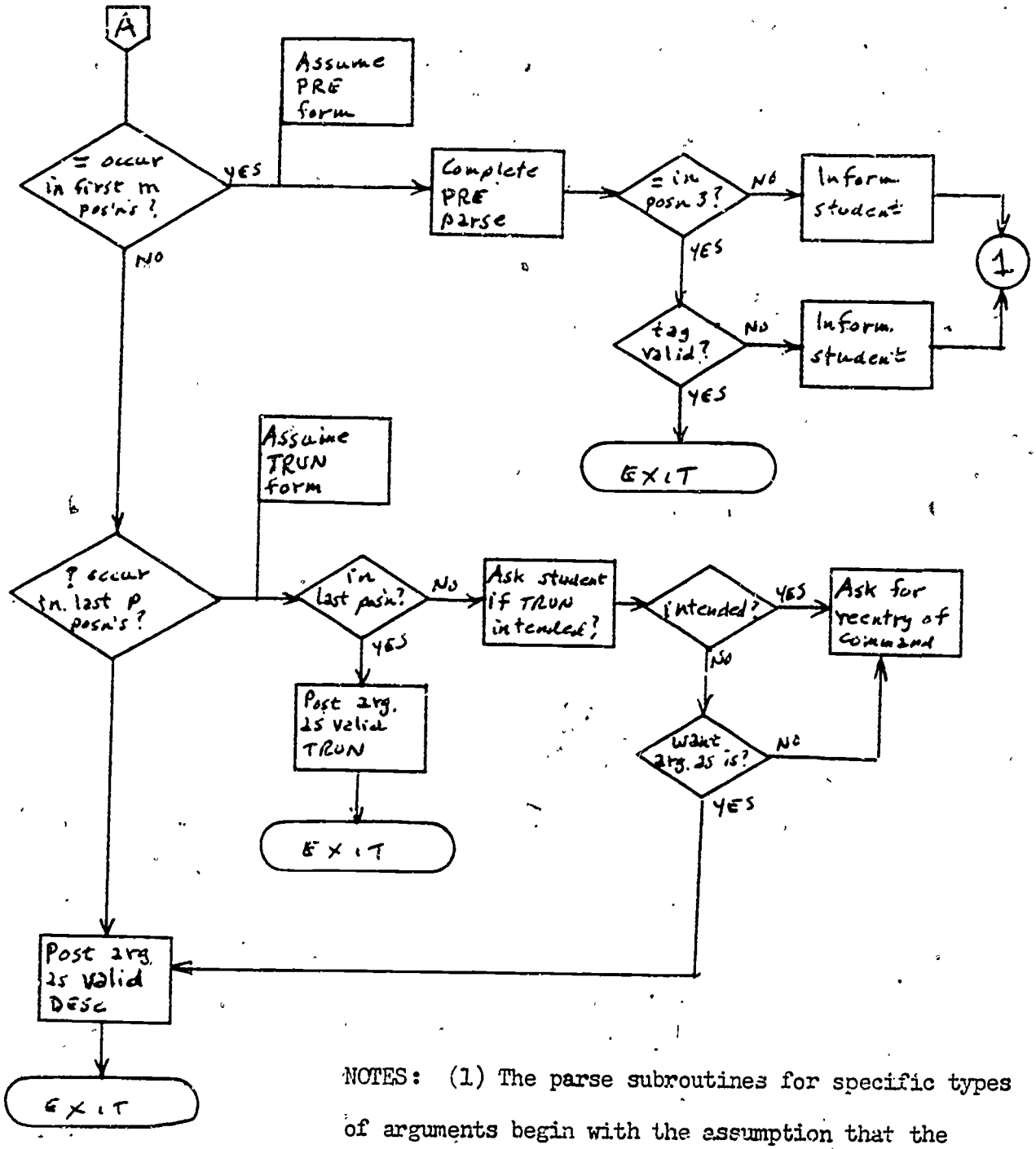


Figure 3.10 (Part 1)



NOTES: (1) The parse subroutines for specific types of arguments begin with the assumption that the argument is of the stated type. They detect errors and post either the correct data elements or error data. (2) the 'Inform student' subroutines tell the student the nature of an error, then go to a sub-routine that counts the number of re-tires and elicits another command.

Figure 3.10 Schematic of argument parsing program, showing quick decision logic.

purposes. The expansion routines are called after the user has entered a SELECT command with a truncated or E-/R- series argument. A truncated term automatically generates and sends EXPAND commands to the data base. The descriptors returned with the designated root word will be stored in a descriptor array for use by the calling program.

If the argument of a SELECT is coded En, the program addresses the n-th position descriptor of the EXPAND table. If the argument is a range of reference numbers, e.g. E1-E6, the first through sixth descriptors are transferred to the descriptor variable array. When the argument is coded Rn or Rn-Rn+m, the RELATE table is consulted and the related terms moved to the descriptor variable or array.

The net effect of calling the expand routine every time a SELECT command with expandable arguments is entered is the maintenance of currency in the set history file. If a set represents a combination of multiple descriptors, the terms will already exist in expanded format. The last generation of updates will include all "expanded" data thus eliminating redundant queries to the set history files.

If expansion of truncated or E-/R- series data results in lengthy lists of terms, an abbreviation code is used indicating that the succeeding data elements represent: the term which has been expanded or related, R or E, first and last term included. Length of descriptors will be interleaved. Terms and tables can in this way be regenerated if desired.

The sequence in which data is recorded is important, especially to the restart and recovery procedures. The command parser posts data about valid commands and errors made as it discovers them. When a valid command is achieved, it is sent to the DBP. The response from the DBP goes to the response parser which posts it to the performance history files. The expansion routines are service programs used to support various other diagnostic programs. Finally, performance analysis programs are run and they, too, may add data to the performance history files. The history updating cycle is complete only when PAR has finished its work. Indeed, a bit is set in the command history file when PAR has completed is final work on a given command to show that all files have been updated with the elements and consequences of that command. Any command for which all processing is not complete must be repeated in recovering from a computer hardware failure (See section 3.8).

3.3.3. Data analysis programs

These are a group of diagnostic programs that are primarily used to support other diagnostic programs. They consist of a program for phase analysis, clustering, expansion of commands and responses, similar term matching, and response parsing.

1. Stage analysis. The purpose of this program is to classify the commands of a search into groups of commands which cohere as stages. We have previously defined search phases. A stage is an instance of a phase. For example, a search might proceed from phase 1 to phase 2 back to phase 1. We would call each of these a stage and number them 1, 2 and 3. Stage analysis data is added to the command history file and is used by the performance analysis routine.

Stages are primarily recognized by the predominance of one of the five phases of searching: index search, logic formation, record display, procedural and diagnostic. Added to this is a mixed phase in which no phase predominates. Pragmatic rules are applied to distinguish borderline cases. These rules will be discussed below in terms of three passes through the search.

Each command is assigned one of five phase codes depending upon the phase to which the command ideally belongs. Groups of commands, referred to as "strings," will become stages, either in their own right or in combination with other contiguous strings.

On the first pass of the analysis, "nuclear strings" are identified. These are strings made up of : (1) any two or more contiguous commands of the same phase, or (2) any two or more contiguous commands each of a different phase. The former are pure strings dominated by the given phase; the latter are mixed strings in which no phase dominates. Limiting nuclear strings to at least pairs of commands in the same phase is a somewhat arbitrary decision. The choice of this and other constants are subject to revision following experimentation.

On the second pass, "string capture" may take place. This involves merging short intervening strings with longer bounding strings. The following rules apply: The intervening string must be shorter than the bounding strings; the bounding strings on each side must be dominated by the same phase; and the intervening string may not exceed two commands. This process is recursive in that a string resulting from a capture must be considered as a possible bounding string. Captures may not become so extended that less than 70% of the commands belong strictly to the dominant phase. Both the rule that intervening strings must be nuclear and the 70% rule are arbitrary and subject to revision.

By now strings have become stages. On the third pass, each stage is analyzed for its relationship to its contiguous stages, expecially the one following it. Relationships may be of three types: 1) transitional, where the stage falls between two other stages whose dominant phases differ; 2) excursional, where the stage falls between two other stages which have the same dominant phase; and 3) terminal where the stage is at one or the other ends of the search.

Each stage's relationship to its predecessor is also evaluated in terms of direction and distance. Direction is positive for forward movement in the search (index search to logic formation to record display) and negative for backward movement in the search. Distance is the difference between the dominant phase of the present and that of the previous stage. In general, it is a measure of the "steadiness of progress" in the search.

Thus, for each stage, three parameters are generated in the stage analysis program: dominant phase code, relationship code, and directed distance from previous stage. Stages identified by this analysis are posted in the command history display. The three associated parameters are used to direct an interactive discussion of strategy with the student.

2. Clustering program. The purpose of clustering program is to determine when the student has changed tack in his search, i.e. stopped pursuing variations on one particular combination of descriptors and begun pursuing

sets based on a different collection of descriptors. The purpose in doing this is to detect the extremes of excessive dwelling on one combination without detectable progress toward an acceptable final set, or frequent change of direction without taking the time to see if the initial logical combination can be improved upon. Clustering in this context is not quite the same as it is in such contexts as document classification. We are not looking for elements (commands) that are similar to each other; we are looking for points at which a sharp break is made with work immediately preceding.

The following terms are used in the logic description:

$C_1, \ldots, C_k$ ::= The sequence of COMBINE commands (or multi-term SELECTs in order of entry by student. There may have been intervening non-COMBINE or SELECT comman

N_k ::= Number of index search or print (or other non-COMBINE/SELECT) commands preceeding command C_k.

D_k ::= Number of descriptors used in COMBINE (henceforth used to include the multi-descriptor SELECT) command C_k.

$M_{i,j}$ ::= Number of descriptors in common to COMBINE command C_i and C_j.

R_k ::= Number of COMBINEs in succession that have indeterminate (see definition below) similarity to predecessor.

$S_{i,j}$ ::= Measure of similarity between command C_i and C_j.
$S_{i,j} = 1/2(M_{i,j}/D_i + M_{i,j}/D_j)$
$= M/2(1/D; \pm 1/D;)$
$S_{i,j}$ is the mean of the number of matching terms in the two commands divided by the number of terms in each of the two commands.

$G_{m,n}$ ::= The nth command in cluster m.

$L_{m,i}$ ::= The number of commands in cluster m that match ($S_{k,i} \geq T_1$) command C_i.

$K_{m,p}$ ::= The number of links between cluster m and cluster p. The number of links is
$$K_{m,p} = \sum_{i \in p} L_{m,i}$$

T_1 ::= Threshold value of $S_{i,j}$ such that if $S_{i,j} \geq T_1$ then C_i is similar to C_j.

T_2 ::= Threshold value of $S_{i,j}$ such that if $S_{i,j} < T_2$ then C_i is not similar to C_j.
NOTE: If $T_2 \leq S_{i,j} < T_1$ then the similarity between C_i and C_j is <u>indeterminate</u>.

T_3 ::= Threshold value of N_k such that if $N_k \geq T_3$ then if $S_{i,i-1}$ is indeterminate, C_i and C_{i-1} are considered not in the same cluster.

The procedure is as follows:

1. Whenever a new COMBINE or SELECT with multiple terms is received from the student, and is syntactically valid, store it in list as C_i.

2. Compute N_i and D_i if $i>1$ (use actual descriptors, not set numbers).

3. Compute $S_{i,i-1}$ if $i>1$.

4. If $S_{i,i-1} \geq T_1$ then C_i is in the same cluster as C_{i-1}. Add C_i to cluster $G_{m,n}$, increase value of n by 1.
   If $S_{i,i-1} < T_2$ then start a new cluster, i.e., m = m+1, reset n to 1.
   If $T_2 \leq S_{i,i-1} < T_1$ and $N_i \geq T_3$ then start a new cluster, i.e., increase m by 1 and reset n to 1.
   ELSE $R_i = R_{i-1}+1$

5. Compute $L_{m,i}$ and $K_{m,p}$ for all clusters from 1 to m-1.

The data available from this procedure is:
number of new clusters formed
number of COMBINE commands given
average number of commands per cluster
length of string of successive indeterminate similarity to previous command
last set number created in each string
array of cluster-cluster similarity measures
indicator: was a new cluster begun with the most recent command

3. Expansion routines. One of the advantages of a language such as used with DIALOG and other search services is that condensed notations can sometimes be used to avoid having to enter lengthy, often repetitive statements. For example, following a DIALOG EXPAND command another EXPAND or a SELECT can be given, making reference to a term on the first EXPAND table by line number. The next command could be, for example, EXPAND R6 or SELECT E12-E14. One of the functions of the expansion routines is to translate such usage back into original descriptor form, for storage and analysis purposes and for later use in displaying history data the student. Three such programs are planned. One translates EXPAND line numbers as illustrated above, one translates set numbers into their original defining descriptors, and one retains information about what terms might have been seen on an EXPAND display.

1. E- and R- expander. Usage such as E6, R9 or E2-E4 are permissible in the arguments of SELECT and EXPAND commands provided there has been the requisite number of preceding EXPANDs. This expander converts each term so referenced back into the original

descriptor the line number represents and stores this information for use with any program needing descriptor information.

2. COMBINE Expander. This program converts set numbers used with a COMBINE into descriptors or combinations of descriptors. Only SELECT may create a set using descriptors, rather than set numbers in the definition. Thereafter, COMBINE commands build upon previously defined sets, as in the sequence

```
SELECT A (Set 1)
SELECT B (Set 2)
SELECT C (Set 3)
COMBINE 1 AND 2 (Set 4)
COMBINE 4 OR 3 (Set 5)
```

In analyzing cluster formation and in displaying set history, the numeric arguments of the COMBINE commands will be converted into descriptors. In the illustrated case, the results would be:

```
Set 1:  A
Set 2:  B
Set 3:  C
Set 4:  A AND B (rather than 1 and 2)
Set 5:  A AND B OR C
```

3. Expand Index Program. The function of this program is to record the first and last descriptor of every EXPAND table sent by the data base processor in response to a student command. Input required by this program is generated by a subroutine which when called extracts fields of the EXPAND table. These field values will be stored in subscripted variables indexed from 1 to n (n = number of lines sent by the DBP minus header lines).

The program examines the descriptor field array and isolates the first and n-th position terms. The length of each of these descriptors is computed using the Basic-Plus LEN function which returns the position of the rightmost non-blank character.

The line to be written to the student history file is a string concatenating the following data elements:

EXPANDED INDEX ENTRY =	LENGTH DESCRIPTOR A	LENGTH DESCRIPTOR B	DESCRIPTOR A	DESCRIPTOR B
	(Fixed Fields)		(Variable Fields)	

This entry is written to the next block number and address recorded in the general series data on the seventh block. (See 3.5.7.)

General series data are updated by incrementing the total number of descriptor pairs and computing the next entry block and address. The directory references to pairs are also reordered to correspond with the alphabetical arrangement of first descriptors.

4. Word similarity program. This program is used when testing to see if a zero set based on a single descriptor used a term not seen by the student on an EXPAND display. This requires that an EXPAND ⟨descriptor⟩ be issued by IIDA and the results analyzed by this program, but not transmitted to the student. The descriptor will be the argument of the SELECT that led to a zero set. It will appear in position E6 of the display. All other words on that list will be compared with the base word to compute, for each, a measure of closeness of spelling. For each word compared with the base word the number of characters in common will be counted. A high proportion of letters in common does not, of course, imply semantic closeness. The table of measures will be sent to the student for his information and decision.

5. Response parser. The response parser examines and breaks down messages received from the DBP in response to valid commands. It must first check for error or change of state messages. Failing to find one of these, it knows, from the command issued, what form of response is expected. It should verify this, which can normally be done by checking only the first few characters of a response. If the correct form of message was received, the parser breaks the message into its constituent elements and posts the relevant ones to the various history tables. This program is clearly specific to the data base system used and is subject to change with any changes they may make in format.

3.3.4 Performance analysis

The performance analysis routine (PAR) monitors five aspects of the student's search. These five aspects are: (1) zero set formation, (2) phase and cluster analysis, (3) /HELP and time use, (4) error analysis, and (5) extraneous and redundant commands. These five areas are monitored on two levels. The preliminary analysis portion of the PAR is designed to catch problems in the search by comparing some rather simple statistics calculated in each area with threshold values. When the threshold values are exceeded an in-depth analysis of the area is run. There are appropriate in-depth analyses for each of the five preliminary analyses. The in-depth analysis performs calculations designed to enable the program to assess why the threshold was crossed. The reasons for crossing the thresholds are not unique. Each threshold can be exceeded for one of a number of related reasons. The objective of the in-depth analysis is to determine which of the possible reasons is the most plausible. There is always the possibility that none of the planned reasons appear causal.

Preliminary analysis is carried out after the response parser and student data base update are completed. As protection against loss of information in the event of a system failure, a system of flags is used. A master flag is set to show "processing" status when the preliminary analysis is begun. As each portion of the preliminary analysis is completed, a minor flag is set. When the preliminary analysis is complete, including any in-depth analyses that may have been called, the master flag is changed to "waiting" status. Control then passes back to the shell for the decision on successor process.

1. Preliminary analysis. The preliminary analysis is designed to operate in a minimum of time. The threshold checks made are listed below. Actual threshold values will be determined experimentally.

TC1. /HELP use

Is the number of /HELP calls T1% or more of the total number of commands entered?
Is the amount of time spent using the /HELP procedure more than T2% of the total time of the search?

TC2. Error analysis

Have more than T3% of the commands of any one type been in error?
Have more than T4% of the last T5 commands been in error?

TC3. Zero set formation

If the most recent command formed a zero set, have more than T6% of the last T7 set-forming commands formed zero sets?

TC4. Clustering threshold

If this command is a COMBINE-like command, is it in the same cluster as the previous COMBINE-like command?
If yes, does the number of consecutive commands in this cluster exceed T8?
If no, is the average number of commands per cluster less than T9?

TC5. Phase checking

Is this command in the same phase as the last command?
If yes, is the total number of consecutive commands in this phase greater than T10?
If no, is the number of phase changes in the last T11 commands greater than T12?

TC6. Extraneous commands

Is this command a duplicate of a previous command?
If yes, is it a command for which repetition is meaningful?
Is the total number of duplicate commands in this search greater than T13?

TC7. Time use

Was the time between the response to the previous command and the entry of this greater than T14 seconds?
Is the total number of commands in this search whose response time is greater than T14 seconds greater than T15?

2. Control sequence. The order in which the thresholds are checked is shown in Figure 3.11. In the following discussion, the threshold checks are referred to as TC1 or TC2, etc. These numbers correspond to the order in which the threshold checks were described in the previous section.

The type of command being checked determines which threshold check is used first. The command is considered to be one of the following four types: (1) /HELP, (2) an error, (3) set-forming (SELECT, COMBINE, LIMIT), or (4) any other. Type 1 commands begin with TC1, type 2 begin with TC2, type 3 begins with TC3 and type 4 begins with TC5. The command proceeds as follows: if the command is type 3, it is next checked by TC4, then TC5. Types 2, 3 and 4 are checked by TC6 and all types of commands are checked by TC7.

When any threshold is exceeded, the appropriate in-depth analyses are called. When the in-depth analyses are complete, the command resumes its path through the threshold checks. Thus, a given command could theoretically exceed more than one threshold in its path through the preliminary analyses. After the command has been subjected to all the appropriate threshold checks, control passes back to the shell of the instructional processor which called the PAR.

3. In-depth analysis - general. In-depth analyses are designed to analyze data from thresholds crossed in the preliminary analysis. There are five in-depth analyses: (ID1) /HELP and time use, (ID2) error analysis, (ID3) zero set formation, (ID4) phase and cluster analysis and (ID5) duplicate checking. Details for each of these analyses will be given later.

When called, the ID enters the code for its call on the command and control data file as well as recording the clock time of the call. The clock time of the return is also recorded. Each ID has a counter for the number of times it has been used. Time is logged to enable later analysis of the cost of PAR in light of the benefits. When entered, the ID checks to see if it has been used more than T16 times before. If so, the proctor is notified and can participate in negotiating with the student. In the case of passing the threshold for /HELP use, the proctor is notified immediately and all negotiation is conducted by the proctor. This seems the best way to help the student who has already extensively used the resources of the /HELP program. If the number crossings of the threshold does not exceed T16, the counter is increased by one, and the data from the preliminary analysis and the data files is analyzed to decide how to deal with the student's possible problem. When the in-depth analysis arrives at a plausible reason for the threshold crossing, the data is presented to the student with the conclusion reached. The ID negotiates an agreement with the student on what was the cause of the threshold crossing and what action the student intends to take. On completion of the negotiation, the ID logs the return to the preliminary analysis program which called it. The threshold counters for the thresholds crossed in the preliminary analysis are returned to zero. The PAR continues to the next threshold check in the preliminary analysis, or if all have been checked, returns control to the shell in the instructional processor which called it.

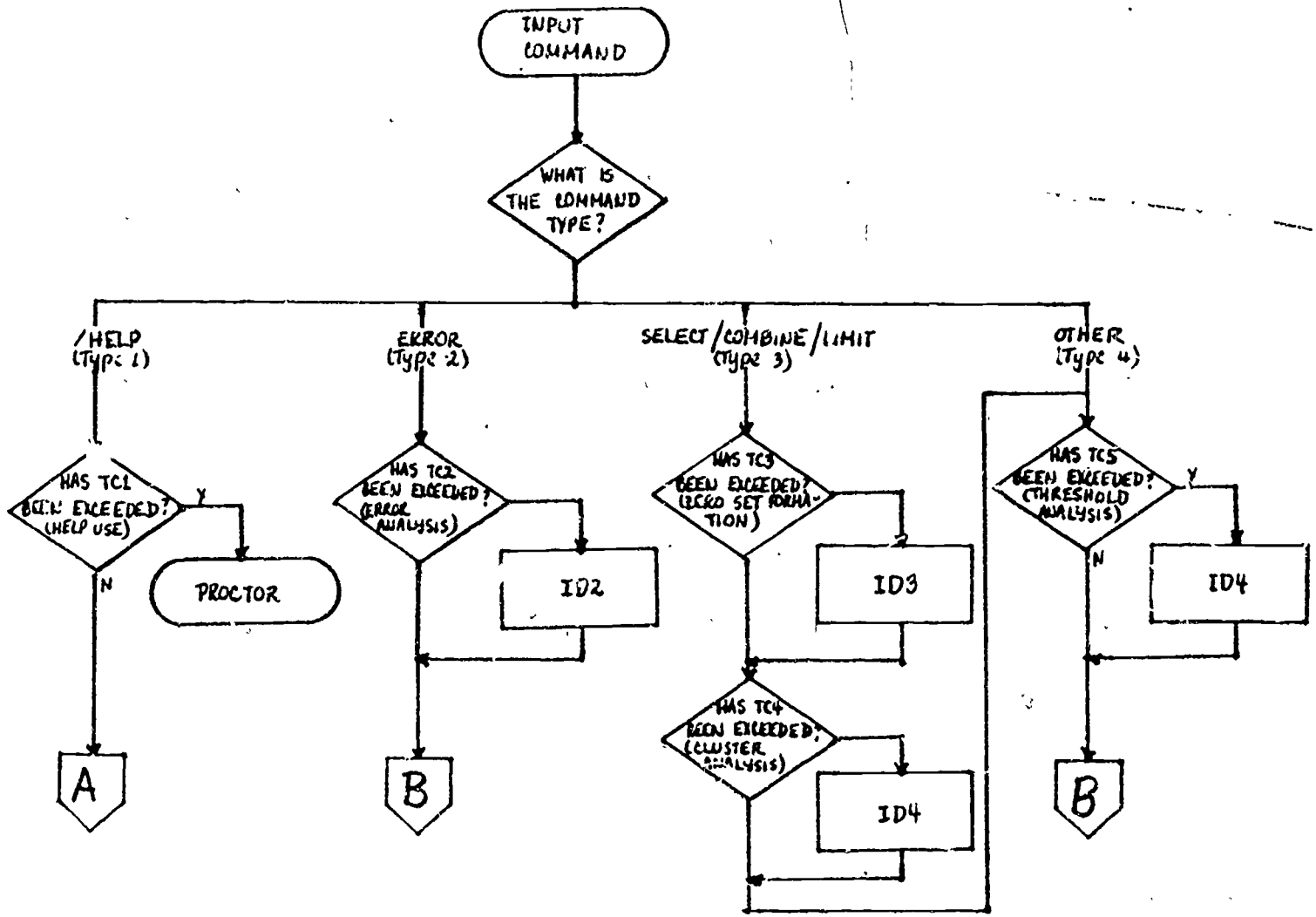


Figure 3.11. Order in which thresholds are checked by the PAR.

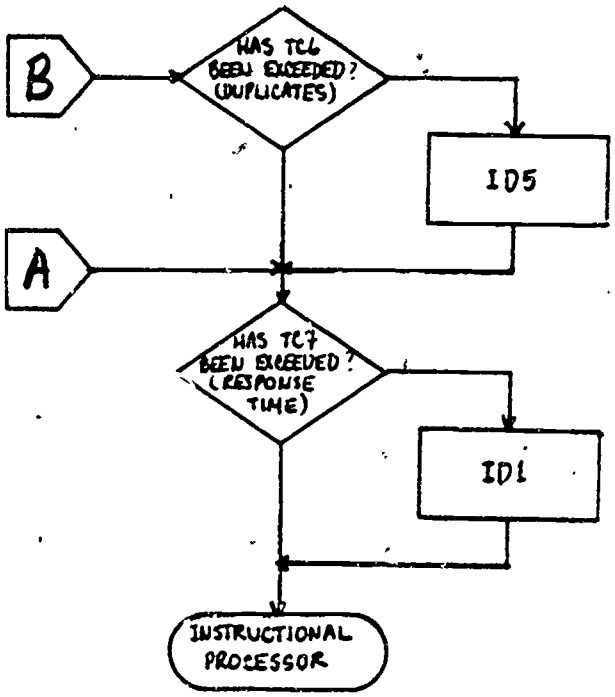


Figure 3.11. Continued.

As a result of the in-depth analysis, the program discusses the student's search with the student. In the process of the discussion, the PAR can use any of the resources of IIDA. Another in-depth analysis may be called, some of the displays from the /HELP program can be used and text from the tutorials is also available. The PAR is intended to be flexible not only in detecting trends in the student search but also in presenting alternatives. The ultimate judge of the search remains the student who may refuse the PAR suggestions without prejudice. Records of student use or rejection of PAR suggestions will allow for evaluation and growth of the system.

4. Details of the in-depth analysis. Each threshold can be crossed for several reasons. The in-depth analyses take the data which exceeded the threshold value and add to it some data from the student performance data base. Certain calculations are made in the in-depth analyses to check for patterns in student performance. The path of the in-depth analyses depends on how each threshold was crossed. Discussion of each in-depth analysis will be divided into sections by which part of the threshold check called it.

ID1. /HELP and time use

A. If use of /HELP exceeds T1% or T2%, notify the proctor

B. If time between commands has been greater than T14 seconds T15 times, ask if the student regularly works at this pace. If not, offer /HELP menu.

ID2. Error analysis

A. Of the last T5 commands of which T4% are in error:

1. have errors been in one type of command
2. have errors been in the command words
3. have errors been in the argument of commands
4. none of the above

B. Of the T3% errors in the use of one type of command

1. have errors been in the command word
2. have errors been in the argument of commands
3. none of the above

ID3. Zero set formation

Of the last T17 zero set forming commands:
A - how many were formed using SELECT with:

1. string search
2. a descriptor not seen in an EXPAND table
3. a descriptor seen in an EXPAND table
4. none of the above

B - how many were formed using COMBINE with:

1. AND with small or zero sets
2. AND with many sets
3. NOT
4. none of the above

C - how many were formed using LIMIT:

1. and, if limiting by accession number, failed to use UPDATE to determine the correct number range
2. and, if the set being LIMITed was formed using an identifier (ID), limited the set by major or minor descriptor
3. none of the above

ID4. Phase and cluster analysis

A. If the number of commands in the current phase is greater than T10:

1. have all three phases been used at some time in the search
2. if no, what phase or phases have not been used
3. if yes, what proportion of the total number of commands have been in each phase

B. If the number of phase changes in the last T11 commands is greater than T12:

1. have all three phases been used at some time in the search
2. if no, which two phases are involved in the current phase shifting
3. if yes, what proportion of the total number of commands have been in each phase
4. if yes, is there a most frequent order of occurence for the commands

C. If the number of COMBINE-like commands in this cluster exceeds T8:

1. is the number of items in the most recent sets in this cluster converging on the student's goal
2. is this the only cluster to be formed so far in this search
3. if yes, how many phases have been used
4. if no, what is the average number of commands for each of the other clusters

D. If the average number of commands per cluster is less than T9:

1. how many clusters have been formed
2. how related are the clusters which have been formed
3. is the number of items in the most recent sets in the cluster converging on the student's goal
4. how many phases have been used

ID5. Extraneous or redundant commands

Are all the T13 duplicate commands:

1. the same command type
2. set-forming commands
3. non-set-forming commands
4. entered in pairs, so that the duplicate is the next command after the original
5. lacking in a pattern

3.4 Tutorial Subroutines

In review, the shell of each tutorial: controls the sequence of frames, changes the frame number parameter at the top of each frame, and controls conditional branching from section to section within the frame, or to the top of other frames, using a code returned at the end of most sections.

A section is a unit of coding which accomplishes a single instructional task as part of the larger instructional goals of the frames. Three types of sections are possible, combined in various ways within each frame. They are: non-command response sections, command response sections, and program controlled command sections. Sections call a series of subroutines, usually between 3 and 7 per section. The characteristics of each type of section and of their subroutines are discussed below and are illustrated in Figure 3.12. Each of the three section types usually begins with a discussion although this may be omitted or inserted at the end of the section.

3.4.1 Non-command response sections

This type of section most nearly approximates coding found in conventional computer-assisted instruction (CAI). Text is printed, a response is elicited, the response is compared against several possible responses, the student performance data base is updated, and a code is returned to the shell to indicate which section should follow. At the top of every section a text-group code is set. A discussion of the subject follows by calling for the text-file printout subroutine. The discussion usually ends with a request for student input. The student input subroutine is called to accept the input.

At this point the expected responses may be stored as either explicit strings in the program or by data taken from the text file. In the latter case, the text-group code is changed and a subroutine

which sets up arrays from the text file is called. The frame number and text-group code together indicate a set of possible answers stored on the text file. If there is a match of the student response against the array, the index (pointer) of the response may become the code returned to the shell for choosing the next section. If there is no match, the subroutine sets the code to the maximum index plus 1. In any case, the student's response is recorded on his data base for later analysis.

3.4.2 Command response sections

This type of section allows students to input commands which are sent to the data base processor for a response. Text is printed, a command is elicited, the command is parsed and verified, it is passed on to the data base processor, the data base processor's response is printed, the student performance data base is updated, and a code is returned to the shell indicating which section or frame should follow. Again, a text-group code is set at the top of the section. A discussion of what is to follow may be printed here by calling the text-file print-out subroutine. This discussion will end with a request for student input, made possible by calling the student input subroutine.

Command response sections will then pass to the parser (Section 3.3.1) the limits of acceptable input. These limits may specify the explicit command or a range of acceptable commands.

When the command parser subroutine is satisfied that the command is acceptable, the command will be written onto the programs communications-logging file. (See Sections 3.6.1 and 3.6.2.) The command will then be transmitted to the data base processor. The response will return to the program over the input communications-logging file. The response will be printed and parsed. Data derived from the parsing will be stored on the student's data base. A code evaluating the input may be derived by the command parser and used by the shell to determine which section is run next. Alternately, the branch following this type of section may be unconditional.

3.4.3 Program controlled command sections

This type of section allows the program to send commands directly to the data base processor. Responses thus generated can be very useful for up-to-date examples. Also, data derived from the responses can be embedded in the text to reflect updates of the data base by the vendor. Text is printed, a command is issued, the data base response is parsed, and the response may be printed. Sections like this usually do not affect sequencing of sections within a frame by the shell. Rather, they operate usually in conjunction with one of the other types of sections in which branching is controlled.

Again, a text-group code is set upon entering the section. A discussion of what is to follow may be printed here by calling the text-file print-out subroutine. A command, either in the program in its literal form or taken from the text-file, is issued to the data

SCHEMATIC of typical tutorial shell and subroutines.

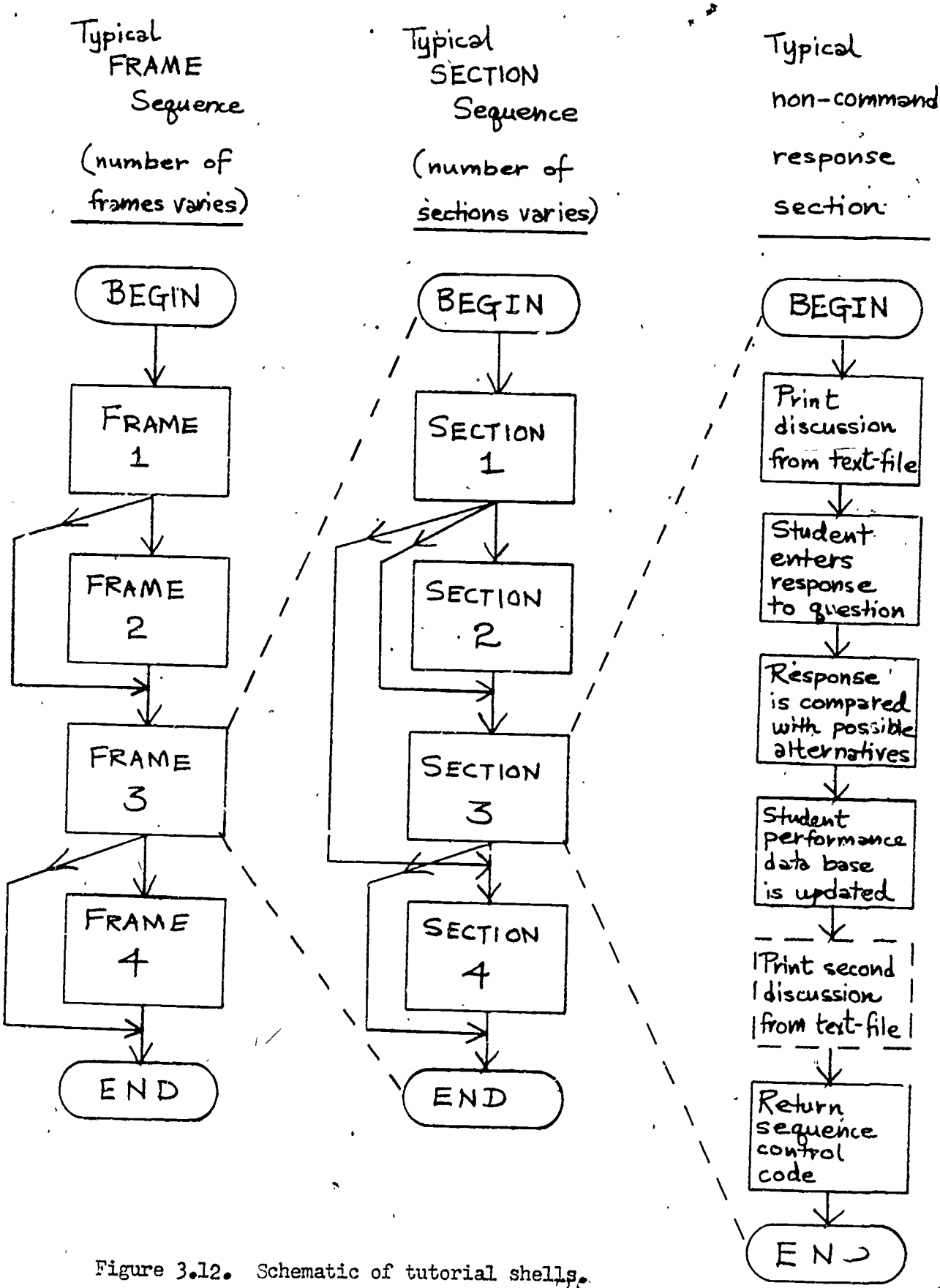
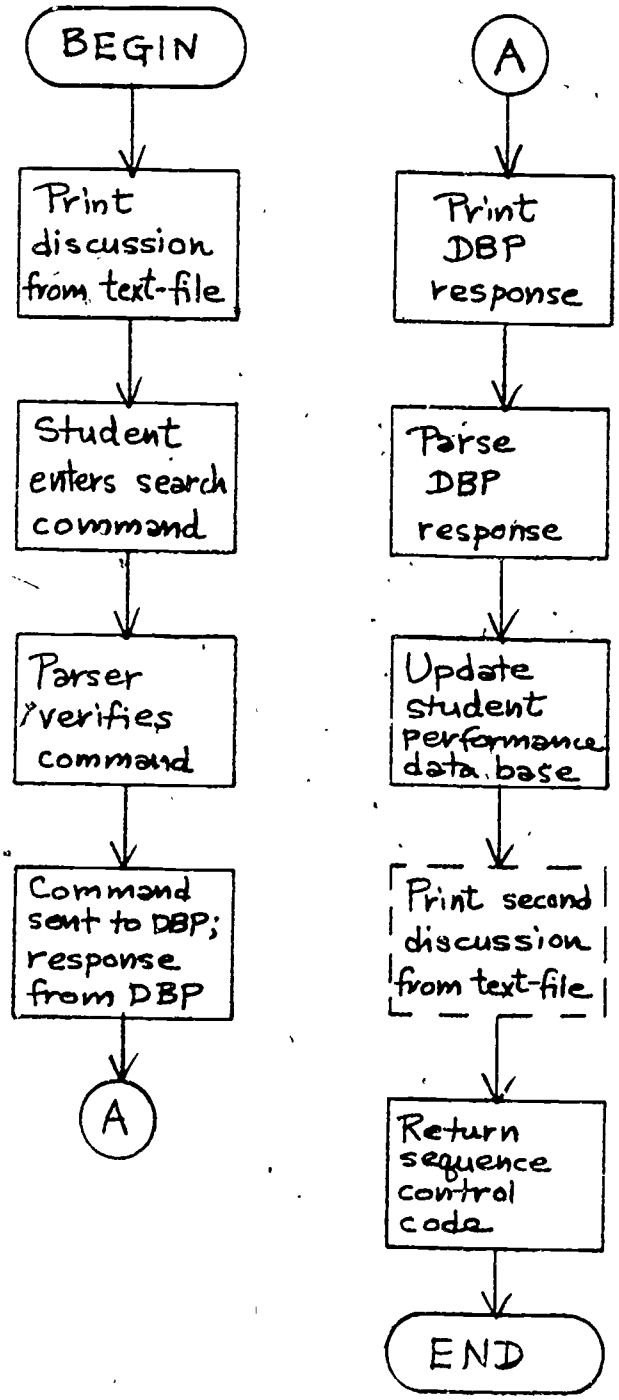


Figure 3.12. Schematic of tutorial shells.

SCHEMATIC of typical tutorial subroutines. (continued)

Typical command response section



Typical program-controlled command section

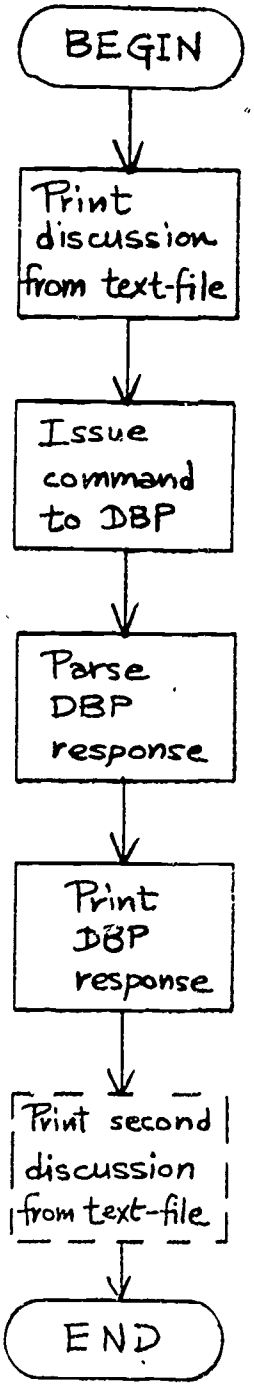


Figure 3.12 continued

base processor through the communications-logging file. The response is parsed and is printed or derived data is embedded in text. The text-group code may be reset for a post-response discussion.

3.4.4 Subroutines

1. Text-file print-out. The purpose of the text-file is to store data for the tutorial programs. This data includes the instructional text, expected responses, and program parameters. (See Section 3.7.4.) The text-file print-out subroutine uses two parameters set in the main program: the frame number and the text-group code. The frame number is used on the first block of the file to identify the frame's group directory on the file. The text-group code is used with the group directory to identify the location of the appropriate text. The text is extracted from the file and printed at the student terminal. After printing, the frame number, the text-group code, and the time since midnight are recorded on the communication-logging file. Any other applicable tagging data is also written. If the entry is to be communicated, a code received from the proctor will have indicated this. The tagging data in this case will indicate who is to receive the entry. For a discussion of the communications-logging files, see Sections 3.6.1 and 3.6.2.

2. Student input. This subroutine will accept input from the student, check for special system commands, and log all student input entries. Input from the student terminal is handled simply with the BASIC command INPUT. A small group of system commands are available to users of the tutorial programs. These enable messages to be sent to the proctor and quick log outs. Comparisons with permitted system commands will be made after each entry. If a system command is recognized, the appropriate action will be taken by coding in this subroutine, including calls to yet other subroutines. Student input entries will be logged as either communicating entries or simply logging entries, depending on authorization from the proctor. Entries are communicated to the concentrator and data base only after having passed through the command parser.

3. Expected response set up. This subroutine is similar to the text-file print-out in that it accesses text from the text-file using the frame number and text-group code. Instead of printing at the student terminal, however, the text-group is read into an array of variables, one line into each array entry. The course author is responsible for setting up these responses in some order, perhaps from the most to the least preferred. Generalized responses may also be encoded in this text-group, if the course author finds that desirable. The student input will be compared with entries in this array. If a match is found, the index of the array element could serve as a code indicating the "goodness" of the response. This code may then be used by the shell for calling in the next section or frame. If no match is found, a code indicating this will be used instead.

4. DBS interface. This subroutine will call both the command and response parsers (see Section 3.3.1) as subroutines and will manage posting data to the student data base and the logging file. After the command parser has elicited and recognized an acceptable

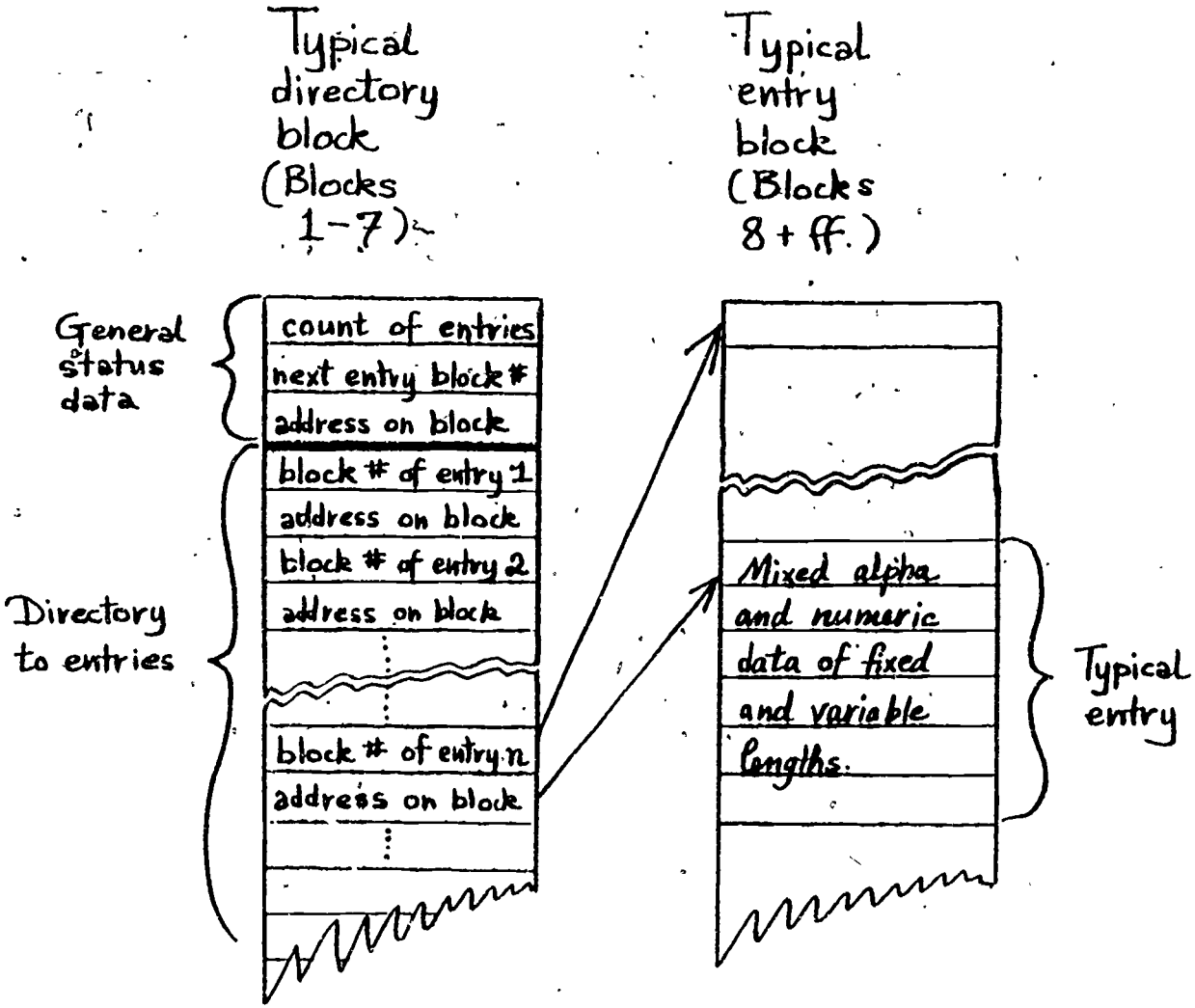


Figure 3.13. Typical directory block and typical entry block in the student performance data base.

command, this subroutine will send the command to the data base processor through the communications file. It will then print the response and call in the response parser. Data isolated by both parsers will be transferred to the student data base. The data base processor will return a code through this subroutine to the shell, indicating how many times input was requested before an acceptable command was entered.

5. Severe problem filter. Severe problems of several sorts can be identified and referred automatically to the proctor. These include: repetitive unrecognizable responses, exact repetition of responses, user-instigated infinite loops, and other "wise-guy" behavior. This subroutine follows the user input in most cases. Codes indicating severe problems can be generated in the command parser and picked up by the severe problem filter for transmission to the proctor. This subroutine will be able to send a message over the communications file to the proctor.

3.5 Student Performance Data Base

A file is maintained for each student which records each student's history of performance since beginning to use IIDA, with emphasis placed on the current or last search carried out. The file is attached to any program which the student runs, although data is largely gathered from activity with the exercise and the assistance modes in which the student performs real searches. The record I/O technique available under RSTS/E is used, with blocks of 512 bytes each.

The first seven blocks contain seven directories for the history of the student performance from seven different perspectives. The first block is a directory to a series of long-term data, covering the entire association of the student with the system. The next six blocks are directories to series of data on the current search. These may be copied as frequently as desirable to off-line storage media, which can be mounted at a later date for research on student searching technique and behavior.

Figure 3.13 shows how directory blocks are related to entry blocks containing data series. Each directory block begins with several bytes containing general data about the entire series. This includes, but is not limited to, the total number of entries in the directory and the block and address available for the next entry. The directory itself follows. It generally consists of the block number and address in that block of each entry. Entries in the data series are found beginning with the eighth block. See the figure below.

Directory Block	Data Series
1	Personal and long-term
2	Command and control
3	Sets created
4	Sets viewed
5	Records viewed
6	Descriptors used
7	Expanded index

Figure 3.14 Directory blocks and their associated data series.

3.5.1 Personal and long-term data

General series data found on the first block includes both permanent data and data updated with each new search. Permanent data includes: password to enable use of file; personal identification code; affiliation code; total number of log-ins; and, for each program available in the system, number of uses in which the program was begun and number of uses in which the program was completed. The personal identification code is used in lieu of a name against a list and is made available to only authorized personnel. The affiliation code, indicating class, company, or division, is similarly protected.

Data updated with each new search includes: total number of searches; and block number and address of next entry. The directory to all searches made with the IIDA system includes the block number and address of each entry. The long-term data for each search remains stored and updated in the program's core area. The file is updated when the search is completed or when the student otherwise logs out of the system.

Data on each search includes: date and time that the search began; time in seconds taken by the search; how many commands used, by type of command; the total number of sets formed, by type of command; total number of distinct descriptors used, by type of command; total number of records viewed, by format; average relevance of all records viewed in this search; total number of /HELP calls; total time spent in /HELP; total number of performance analysis calls (in-depth); and total time spent in the performance analysis program. See Section 2.5.1 for a more complete discussion of the use of this data.

3.5.2 Command and control data

General series data found on the second block includes: total number of commands indexed by the directory; and the block number and address of the next possible entry. The directory to commands used in the current search, also found on the second block, includes the block number and the address in that block of each command entry.

Data on each entry includes: a code for entry type (which search command, which control command, an error, or a PAR in-depth analysis call); clock time when entry was made, in seconds since midnight. The following three data items may use the same relative address in each entry: for valid search commands, phase of command; for invalid commands, the error code; and for /HELP, the clock time since midnight at which the student returned to the main program. In this last case, the first time entry is the time at which /HELP was called.

The entry will also include the set number, if the command was SELECT, COMBINE or LIMIT The length of the remaining data in the entry is then indicated. If the entry is a command or an error, this will be the text of the entry. If the entry is /HELP, this will be the sequence of selections made from the menus in /HELP. If the entry is an in-depth performance analysis, this remaining data will be an encoding of the activity within the PAR.

When histories of valid commands are displayed, all other commands in this series (errors, control commands, /HELP calls, and PAR calls) will be bypassed. For histories of errors, all other commands will again be bypassed. Integration of commands in this series allows for reviews in which the sequence and position of various types of entries is of interest

The command and control data series is updated with every search entry by the student. The entry type code comes from the command parser, if the entry is a command or an error; from the input routine, if it is a control command; or from the performance analysis program, if it is a PAR entry. The time in seconds since midnight is isolated by assigning the value of the BASIC-Plus function, TIME(0%), to a variable immediately following a student input. The phase of the command comes from the last search phase analysis. The set number is returned by the response parser. Data on PAR and /HELP activity is collected during the operation of those two programs.

3.5.3 Sets created data

General series data found on the third block includes: total number of sets indexed by the directory; and the block number and address of the next possible entry. The index to the directory is the set numbers themselves with entries in the directory corresponding one-to-one with the set numbers in the search. The directory to sets

created in the current search, also found on the third block, includes the block number and the address in that block of each set entry.

Data on each entry includes: the number of records in the set; the code for the type of command that created the set; the cluster to which the set belongs; the length of the original argument; the text of the argument. If the set was created with the COMBINE command, the length of the argument reduced to elementary set number is given, followed by that argument; and the length of the argument reduced to alphanumeric descriptors is given, followed by that argument.

The sets created data series is updated with every valid SELECT, COMBINE, or LIMIT command. The number of records in the set comes from the response parser. The command type code comes from the command parser. The cluster number comes from the last run of the cluster analysis routine. The text of the argument comes from the command parser. If the command is COMBINE, the argument reduction subroutine is run and the two reduced arguments are stored.

3.5.4 Sets viewed data

General series data found on the fourth block includes: total number of sets indexed by the directory; and the block number and the address of the next possible entry. The directory entries correspond one-to-one with the set numbers; the location of the directory data for any given set can be calculated from that set number. The directory to sets viewed in the current search, also found on the fourth block, includes the block number and the address on that block of each set entry.

Data on each entry is in the form of a linked list referring to all records in the set that have been viewed. Data on this list includes: the record number in the set; the relevance rating given the record; the format in which the record was seen; and the block and address of the next record data. In updating sets viewed data, each entry is inserted into the chain according to its place in the set. That is, if record 4 of a given set were viewed twice in two different formats, the two would be linked together by tracing through the chain, inserting the second viewing data, and rewriting the pointers on the link immediately preceeding the insertion.

The sets viewed data series is updated with every valid TYPE command. The set number and format number come from the command parser. The record number is initialized by the command parser and is incremented by '1' for every subsequent record seen under a given TYPE command. Relevance comes from the Phase 3 routine which asks the student for the relevance of each record.

3.5.5 Records viewed data

General series data found on the fifth block includes: total number of records in the directory; and the block number and address

of the next possible entry. The directory to records viewed, also found on the fifth block, includes the record's accession number, the block number, and the address in that block of each record entry. This directory is dynamic, that is, after each new entry it is reordered by inverted accession number sequence. This simplifies searching for the entry, which is necessary for each record viewed.

Data on each entry is in the form of a linked list referring to every time any given record has been viewed. Data on this list includes: the set number from which the record came; the record number in that set; the format in which the record was viewed; the relevance assigned to the record; and the block and address of the next entry of data on this record.

The records viewed data series is updated with every valid TYPE command. The accession number is isolated by the response parser. The set number and format number come from the command parser. The record number is initialized by the command parser and is incremented by '1' for every subsequent record seen under a given TYPE command. Relevance comes from the routine which asks the student for the relevance of each record. If a match of accession numbers occurs in the search, the chain of entries must be traced through to the end, at which point the new entry may be posted.

3.5.6 Descriptors used data

General series data on the sixth block includes: total number of descriptors indexed by the directory; and the block number and address of the next possible entry. The directory to descriptors found in the current search, also found on the sixth block, includes the block number and the address in that block of each descriptor entry. The directory may be alphabetized according to the descriptor entries.

Data on each entry includes: The length of the descriptor string; the descriptor string itself; the number of times the descriptor was used as an argument of each of the following commands: simple SELECT, free-text SELECT, EXPAND, second EXPAND (thesaurus access), and COMBINE (derived from the alphabetically reduced COMBINE); number of records indexed under the descriptor; and number of terms related to the descriptor.

The descriptor used data series is updated with every valid SELECT, EXPAND, or COMBINE command. The descriptor itself is isolated by the command parser. If the command is COMBINE, the argument reduction subroutine isolates each argument in the alphanumeric form. Data about records indexed and related terms comes from the response parser acting upon SELECT and EXPAND commands. For each descriptor, a match is sought in this series. If found, data on usage will simply be posted to the already existing entry. If not found, a new entry

will be placed at the end of the entry block, and the directory entry to the entry will be inverted in its alphabetic position.

3.5.7 Expanded index data

Expanded index data is essentially the first and last term found in every EXPAND-table. This data helps in determining if a descriptor would be seen on an EXPAND-table, if it is in the index. General series data found on the seventh block includes: total number of descriptor pairs pointed out by the directory; and the block number and address of the next possible entry. The directory to the pairs so derived in the current search, also found on the seventh block, includes the block number and the address in that block of each pair of descriptor entries.

Data on each entry includes: the length of the string representing the first descriptor; the first descriptor string itself; the length of the string representing the last descriptor in the EXPAND-table; and the last descriptor string itself. The directory can be reorganized with each entry so that the first descriptors are in alphabetical order. The expanded index data series is updated with every valid EXPAND command. The two descriptors stored are isolated by the response parser.

3.5.8 Transition from block to block

The last two bytes on each entry block is reserved as a next-block indicator. That is, the block number next used for the current series is stored in these two bytes. Before an entry is actually stored, a calculation is made to determine if enough room remains on the block for the entry. If there is, the data is placed in the block and the block is written onto the file. If there isn't, data is written through byte 510, the next-block indicator is filled with the number of the next empty block on the file, the remaining data is stored on the new block, and the block is written to the file.

3.6 The Communications Processor

The communications processor serves as a means of communication among the student, working through terminals, the instructional processor, the proctor, and the data base processor. Messages may be sent from any of these to any other one. The set of computer programs which physically control the movement of messages among these sources and destinations and perform some auxiliary functions is, collectively, the communications processor.

Message switching is the basic function performed. The others are: message logging, supporting the proctor, concentrating messages between the CP and the DBP, and billing and costing. Message logging provides information of use in analyzing message traffic patterns, needed to spot and correct bottlenecks and to enable the system to

expand without creating bottlenecks. The proctor has some functions which require IP-like programs, but in most cases, he is merely using communications facilities to send messages to students, receive them, or monitor a student's activities.

The concentrator, although only a part of the CP, plays an important role. We have noted previously that the actual percentage of time a line between a student and the DBP is in use is relatively small, expecially for inexperienced users. The concentrator provides a means to reduce the cost of this usage pattern by making fuller use of the CP-DBP line, shared among multiple student users.

The billing program monitors the amount of time a student is using the system and computes its cost.

3.6.1 Message switching

Communication between simultaneously operating programs is made possible by message switchers. These programs manage pairs of input-output files from all programs operating at any given time in the system. Any program operating under RSTS/E and written in BASIC-Plus may have at most twelve internal channels assigned to it. (See Section 3.1.4.) Files on these internal channels may be shared between programs, although only one program is authroized to write onto a file so as to extend it.

Each message switcher may have files assigned to as many as all of its twelve channels as follows: one to the proctor; one from the proctor; one to the concentrator; one from the concentrator; one to each student,) to four students; one from each student, up to four students. As many as four message switchers, accomodating sixteen students as a maximum, will be considered at this point in the design. Messages between each student and the proctor and the concentrator are always possible.

These communications files are handled by the record I/O technique available under the RSTS/E operating system. These files also serve logging functions, described in Section 3.6.2. Each file begins with a header record. This record indicates the following: the user code, the date, the login time, the specific program used (tutorial, exercise or assistance), and whether the file was input or output to the program. Each block on the file begins with a code which indicates if that block has been filled.

The following, referred to as tagging data, is stored with each message on both the input and output files:

- message type code (see Figure 3.15)
- total length of remaining data, including message
- number of destinations

. destination codes

. origin code

. the time since midnight, in seconds

. the message itself

Certain messages may be encoded, specifically, messages to the program and logging entries from the text-file. A time tag is assigned to a variable when the tagging data and message are stored on the block in the buffer. This usually takes place after the appearance or entry of the message itself.

Coming to the instructional program:

. message to be printed at the student's terminal

. message to be used internally by the program

Going from the instructional program:

. logging entry - an encoding of a text-file message

. communicating entry - an encoding of a text-file message

. logging entry - verbatim message from the student

. communicating entry - verbatim message from the student

. logging entry - verbatim message from the program

. communicating entry - verbatim message from the program

Figure 3.15 Types of messages on the communications-logging files, from the perspective of the instructional programs.

An "instructional program" will be defined for the following discussion as any tutorial, exercise, or assistance program which interacts directly with a student at a terminal. From the perspective of instructional programs, the first tagging datum, message-type code, indicates one of the conditions described in Figure 3.15. Communicating entries are those which are conveyed immediately to other programs operating in the IIDA system, e.g., a message to the proctor. Logging entries, further described in Section 3.6.2, are not conveyed to other programs, e.g., a notation about a message printed from the text-file in a tutorial program.

All programs have access to a 16 byte common memory area (See Section 3.1.4). Each message switcher uses two bits from each of six bytes. These bits correspond to the six input and the six output channels attached to the message switchers. One of the two bits indicates

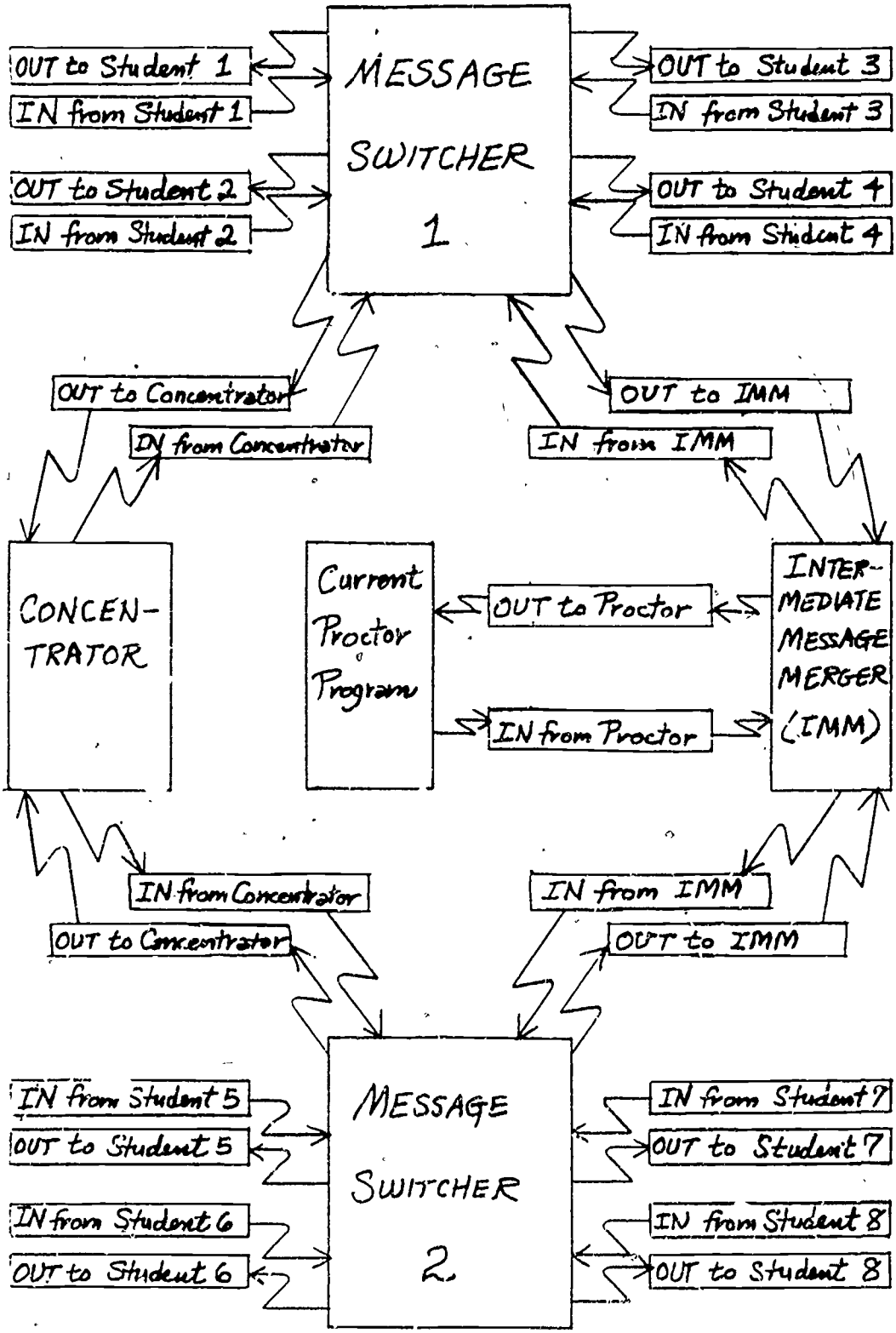


Figure 3.16. SYSTEM FLOWCHART showing two Message Switchers using Communications files.

that the file on the channel contains information for the associated instructional program; the other bit indicates that the file on the channel contains information from the associated instructional program.

Communication may be viewed from either the point of view of a typical instructional or proctor program or from the point of view of the message switcher. The output file of each instructional and proctor program is an input file to one of the message switchers while the output files of each of the message switchers are unique input files to instructional and proctor programs. The logical framework of the message switchers is shown in Figure 3.16.

1. Communication from the point of view of the instructional programs. When a program has a message to send, it proceeds as follows: It reads into the I/O buffer the block which contains the last message sent. The block number had been previously saved in a core variable. If the message is too large for the space remaining in the block, the next-block-indicator is turned on, the block is written out, and the next empty block is called in.

In any case, the following tagging data is written onto the block: the type of communicating entry; the total length of the remaining data and the message; the number of destinations; indications of the destinations; time since midnight; and the message itself. The block is written to the file and the appropriate bit in the common memory area is turned on to indicate that a message is to be transferred. The relative address of the next location and, if necessary, the block number of the output file are both updated in the core variables reserved for this information.

The program may either do some processing or may "wait around" for a return message. To "wait around," the program will enter a loop which checks the incoming message bit every few seconds. Otherwise, the program will check the incoming message bit when the other processing is completed. When the bit is turned on, a message is waiting for the program on the program's input file.

The block of the next input message and the location of the next message on that block are also stored as variables in the program. When the input bit is recognized by the program, the program turns it off. The appropriate block from the input file is then retrieved. If no data is found where it is expected, the next possible block is called in and checked for a message. More than one message may be waiting on the file. For each message, the tagging data is analyzed and the message is extracted and used appropriately. The variables indicating the block and the relative address in the block are updated accordingly.

2. Communication from the point of view of the message switcher. A typical message switcher will proceed as follows: It will check the first/next bit allocated indicating that an instructional or proctor program wants to send messages. After switching the message, the message switch - will check the next bit in the next byte to find out if the

next program wants to send a message, etc. After checking the last byte and switching any messages from the associated file, it will return to the first byte in the common memory area (See Section 3,1,4). The message switcher will iterate this loop the whole time the system is operating.

When a bit indicates that a message is to be switched, variables indicating the block and relative address of the next expected message on the file associated with the bit are used to access that message. The bit indicating incoming messages to be transferred is turned off and the block of the file corresponding to the bit is read into the buffer. Any purely logging data in the file, recognized by the message-type code, is skipped over. The next block of the file is called in whenever necessary. The variable indicating the block and relative address is updated as needed.

The destinations of the messages are determined from the messages' tagging data. Tables on block number and relative address are stored for each file coordinated by the message-switching program. This data enables calling in the appropriate block of the appropriate message switcher output file. Again, if the message and its tagging data exceeds the space remaining in the block, the next-block-indicator is turned on, the block is written out, and the next empty block is called in.

In either case, the tagging data is placed in the block and the block is written out to the file. The appropriate bit in the common memory area is turned on to indicate that a message is waiting for the associate instructional program. The block number and relative address of the next message on this file is updated in the message switcher's variables.

3. Housekeeping. Message switching files for each program are relatively temporary. To distinguish them, each will have a unique name. This name must be associated with a message switcher, a channel, and a common memory bit as long as the files exist. Housekeeping is necessary to identify this association of files with message switchers and to keep track of what channels are free on what message switchers.

A single block of a commonly accessible file, four bits of common memory, and a few lines of code in the message switcher program suffice for housekeeping. The common file block will be logically structured to associate the names of all current files with each of the message switchers, and the bit and channel devoted to the files on each of the message switchers.

When a program is logged on to the system, these areas are checked by the login or reception program. If a message switcher has an open pair of channels, the program's file names are written into the structure and a bit in common memory is turned on for the message switcher. Along with the twelve file bits, this bit is checked periodically by the message switcher. When recognized as being on, the switcher will call in the common file block, read off the names of the new pairs of files, and attach the new files to the open channels. The message

switcher then turns off the bit. Whenever a job is detached or logged off, the logoff program eliminates the file name from the common file block and turns on the message switcher's bit. When checked by the message switcher, the common file block will indicate that those channels are no longer required. The message switcher closes those files on those channels and turns off the bit.

4. Proctor intermediate message merger. Problems arise in sequencing messages to and from the proctor (see Section 3.6.3), since up to four message switchers are sending data to and receiving data from the proctor program over as many as eight files. If these were directly attached to the proctor, only four channels would then remain for all other files used in the proctor programs. The concentrator (see Section 3.6.4) is similarly limited. It, however, can tolerate eight channels tied up in communications. The proctor cannot tolerate this. Therefore, an intermediate message switcher is provided to merge the four input files into one input file for the proctor and split a single output file into four output files for the message switchers. This reduces to two the number of channels allocated to communications for the proctor.

The intermediate message merger operates as follows: The message-switcher-to-proctor bit is checked for each message switcher in sequence. If it is found on, messages are read off the file and written to the single output file to the proctor. (Input file from the proctor's perspective.) Added to the tagging data is a code indicating which message switcher sent the message.

Alternating with this process, the intermediate message merger checks a bit in common memory turned on by the proctor program when a message is written to the proctor's single output file. The message and its tagging data are read. The intermediate message merger determines from the tagging data which message switcher is to receive the message. The message is then written to the appropriate file. Common memory bits are turned off and on, as outlined above. Variables indicating blocks and relative addresses are maintained by the intermediate message merger, also as outlined above.

3.6.2 Message logging

Messages are logged on the same files as are used for communicating messages. (See Section 3.6.1.) To recapitulate, each program has associated with it an input file and an output file. These files are processed by the record I/O technique available with BASIC-Plus under the RSTS/E operating system. The program's input file is strictly reserved for messages coming from other programs via the message switcher. The program's output file sends messages to other programs via the message switcher and logs all activity that takes place at the student s terminal. The communications functions are discussed in Section 3.6.1;

the logging functions will be discussed here.

Messages are logged on the output file following every message entered at the terminal by the student and following the printing at the terminal of every message which originated with the program or with the text-file. Messages from the student and from the program are logged verbatim while messages whose source is the text file are encoded.

All messages logged are preceded by tagging data. For verbatim messages, this includes:

- code indicating that message is a verbatim logging entry, and indicating whether it originated with the program or student
- total length of remaining tagging data, including message itself
- the origin (student program) code
- time since midnight in seconds
- actual text of message itself

For encoded text-file messages, tagging data includes:

- code indicating that message is an encoded logging entry
- total length of remaining tagging data
- origin (student program) code
- time since midnight in seconds
- code indicating which text-file is involved
- frame number and text-group code

Since these simply are logging entries, they have no destination data. The above types of messages, however, can be encoded as communicating entries as well as simply logging entries. In this latter case, the proctor can monitor student activity, discussed below. If the entry is to be communicated, destination codes are included in their normal positions in the tagging data.

The logging-communication files can be used to analyze traffic flow, reposition students after a system failure, and monitor searches in real-time. Each of these functions, performed by programs or parts of programs, will be discussed below.

1. Traffic flow analysis. This program is run in batch mode after sessions for a day have been completed. Switches are set at the top of the program which indicate the population on which the analysis is to be run: one program, a group of programs, or all programs which have

operated during the day. Response times and activity demands can thus be evaluated from the point of view of a single user, a group of users, or the entire system.

The output of traffic analysis is a report tabulating the following data for each message: time in seconds since midnight, time in seconds since the immediately prior entry, user or program number to which this entry belonged, and type of activity this entry represents. Type of activity may be displayed as a check in one of several columns or as a code number. Types of activity include: a text-group from the text-file, text originating in the program, input from a student, communication from a student to the proctor, communication from the proctor to a student, message sent to the concentrator, or message received from the concentrator. This data is derived from the message's tagging data.

The traffic analysis program will process the logging files as follows: Up to five pairs of input-output files will be assigned ten channels in the analysis program. These now are the input files to the analysis program. A working output file will be assigned the program's eleventh channel. Timing, origin, destination, and function data will be read from the data tagging the first message on each of the input files into an array. Data from the oldest message will be reformatted and written to the working output file. Tagging data for the next message in the same input file will replace the data which has just been written. The oldest data now in the array will be reformatted and written and the process is repeated.

If ten or fewer logging-communication files are being considered in the analysis, the working output file becomes the only output, and then the only input, file in the analysis program. Otherwise, the process outlined above is repeated for up to ten more logging-communication files, creating a second working output file. The whole process may be repeated until a maximum of twelve working output files have been created. That is, as many as 120 logging-communication files may be reduced to as many as 12 working files. An analogous process is now used for taking data from these output files and printing the report. Data about messages is processed from the oldest to the latest. Time differences between the consecutive entries are calculated. Output is printed as described above.

2. System failure restart. The use of the logging file for repositioning students following a system failure is discussed in Section 3.8.3. The logging files are used in conjunction with the student performance data base to reestablish control to the right place in each program.

3. Monitoring searches in real-time. The proctor is able to monitor student activity in real time by sending a message to the student's program. This message throws a software switch which directs the program to communicate a copy of every message, except those generated

and sent by the program for internal purposes, to the proctor. The message switcher is responsible for sending to the proctor a copy of all messages going to the student's program from other programs such as the concentrator. The student is notified by a message at his terminal that this monitoring is taking place.

3.6.3 Proctor functions

The proctor has access to a number of programs and capabilities. These include: reviewing in depth the performance of one student at a time; communicating with any student at any terminal; monitoring in real-time all activity taking place at any student terminal; summarizing the performance of any group of students; accessing the logging-communications files of any group of students; and triggering an in-depth performance analysis to be run on any given student.

1. Reviewing performance. This option is based upon the /HELP option available to students. For a discussion of /HELP as a student option, see Section 3.2.4. The proctor has more options available here than do the students. The proctor may see the history of commands issued in three modes while students may see it in only one mode. These three modes are: valid commands; valid and invalid commands intermixed; and valid, invalid, and control commands intermixed.

The proctor can furthermore trace the entire history of any student's use of /HELP. This data is recorded on the student data base as detail on /HELP commands in the command and control data series. After seeing where each use of /HELP is located in the context of all the commands issued by the student, the proctor can trace the history of all options taken and responses made within each use of /HELP. These reviews can be selected by referring to the /HELP command's sequence number in the history table of commands given.

2. Communicating with students. In programs under proctor control, the function /SEND is available. With it, the proctor can send a message to any student program operating in the system. This message is transmitted via the proctor's output communication file. /SEND can take as its argument the names of all programs which are to receive the message and the types of messages they are to receive. See Section 3.6.1 for a discussion of message switching.

Two types of messages may be communicated to a program. The first is to appear at the student's terminal for the student to read and respond to; the second is a message to the program to set an internal software switch. The two functions are distinguished and conveyed by the tagging data that precedes the message on the communications file.

An instance of sending a message to the student's terminal may be found in the proctor's monitoring of student activity. If the proctor recognizes that the student needs a particular type of help, he can

send the student advice with /SEND. An instance of sending a message to a program may be found in the proctor's triggering of an in-depth performance analysis on a student's search. This authorization throws a software switch which routes control around the threshold checkers into the full analysis.

Periodically in the proctor's programs, a subroutine will check the proctor's input buffer for data. If that data is found to begin with /SEND, the subroutine will parse the input for further information: to whom is the message intended and what message is to be sent. Messages to programs will be encoded. A list of this encoding can be posted to a text file for the proctor to review if necessary. The subroutine which checks the proctor's input buffer will look like this in BASIC-Plus:

```
INPUT A$

WAIT n (a second or two)

ON ERR m RETURN (no input)

(process input found)

RETURN
```

3. Monitoring real-time activity. This function has been discussed already in Section 3.6.2, from the perspective of the logging-communications files. Student activity can be monitored by the proctor as that activity happens in real-time. Proctor monitoring may begin as a stand-alone program or it may be chained into the proctor's work area following a recapitulation of the student's current search for the proctor. Monitoring begins with the proctor's program sending a message to the student's instructional program. The message sets a software switch in the student's program which causes every message to be tagged, indicating that copies of the message are to be sent to the proctor. A message is also sent to the student at his terminal notifying him that monitoring of his activity is beginning.

The proctor's monitor program receives copies of all logging entries and communications. These are then printed at the proctor's terminal. The text-file used by the student's program is attached to the monitor program so that text-groups printed at the student's terminal are also printed at the proctor's terminal. The proctor may communicate from the monitoring program by his program control periodically passing through the input-buffer-checking subroutine described above.

4. Summarizing performance. Summaries of the performance of a group of students registered with the IIDA project can be made with the program under this option. Summaries may include or exclude the detail summaries of the last search done or the present search being done by members of the group. The proctor enters a group code to in-

dicate at the beginning of the program what group of students the report is to include, and which type of summary report is to be generated. The performance data bases of the group indicated are attached, one at a time, to extract data from the long-term series and print the requested information. No information about particular students, identified by name or number, is released. Data on the distribution of each characteristic is accumulated for a grand summary at the end of the report. Data presented in these reports includes calculations made from data extracted from the performance data bases. Data presented in the reports are listed in Figure 3.17.

5. Accessing the logging communications files. The proctor can run segments of the traffic analysis program on the activity of any group of students. See Section 3.6.2 for a complete description of the traffic analysis program. The report will include the type of entry, its origin and destination, the absolute time of the entry, and the elapsed time since the previous entry. Summaries of the time distribution of different types of entries are generated.

HEADER DATA

- identifier
- search or summary
- if detail, when was search made
- current search indicator

COMMANDS ISSUED

- total number of commands
- total number of command types
- total number of commands by type
- distribution of time between commands

SETS CREATED

- number of sets created
- number of sets created per command
- distribution of time between the creation of sets
- array of commands used to create sets

Figure 3.17. Data presented in detail and summary reports available to the proctor (continued on next page).

DESCRIPTORS USED

- number of descriptors used
- number of descriptors used per command
- number of descriptors used not found in EXPAND tables
- number of descriptors used in free-text sELECTS.

RECORDS VIEWED

- number of distinct records viewed
- number of records viewed, by format
- distribution of relevance ratings
- number of records viewed out of total records in all sets

ERRORS MADE

- number of errors made
- number of errors made per command
- number of errors made, by type of error

/HELP USAGE

- number of /HELP calls
- number of calls per command
- number of selections of each option
- absolute time in /HELP options
- total time in /HELP out of total time in search
- total time in each option of /HELP

PERFORMANCE ANALYSIS PROGRAM USAGE

- number of PAR calls
- number of calls per command
- number of calls for each type of threshold
- total time for in-depth PAR
- total time in PAR out of total time in search
- total time in-depth by each type of triggering threshold

(Figure 3.17 continued)

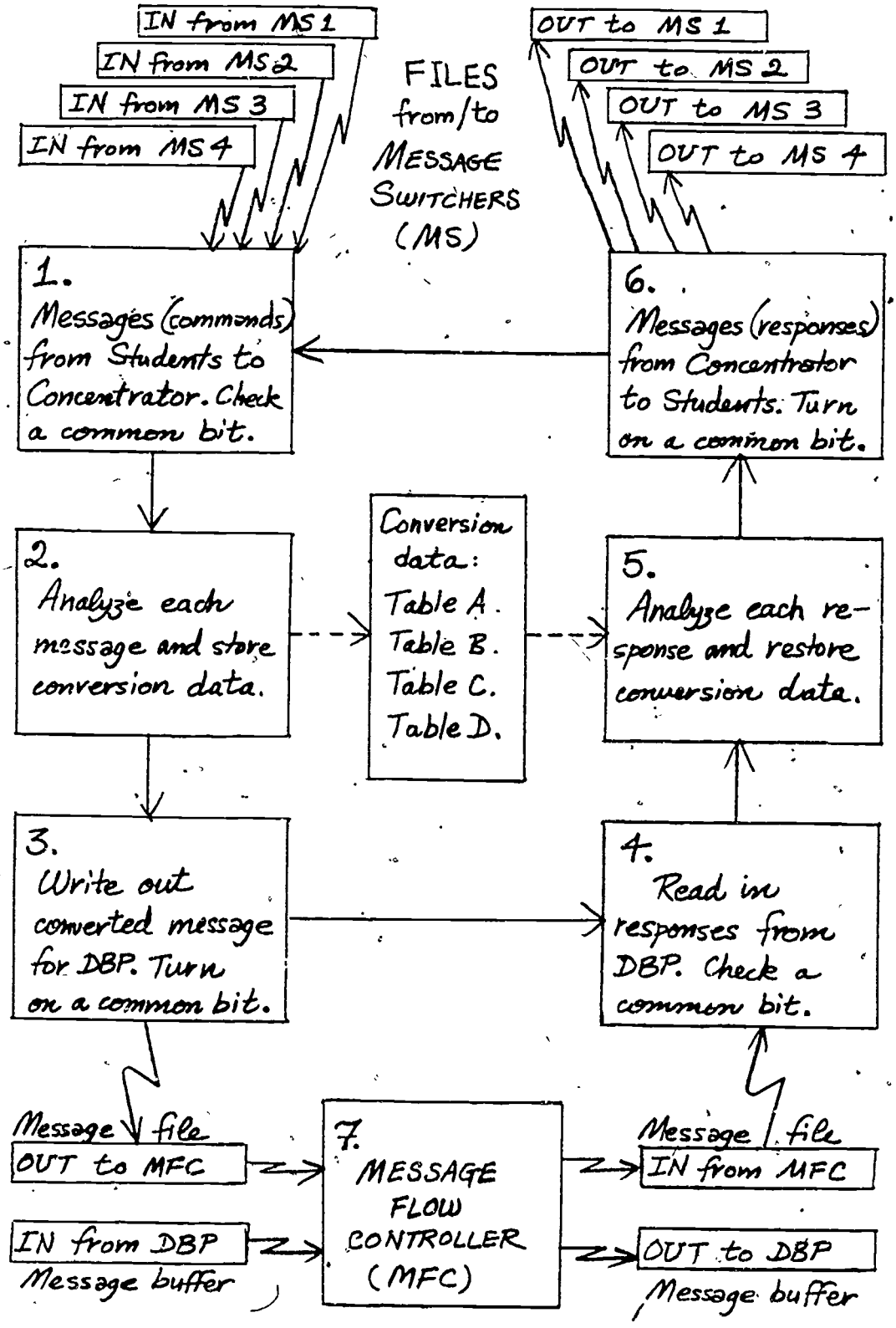


Figure 3.18. SYSTEM FLOWCHART of the Data Base Concentrator.

6. Triggering performance analysis. The proctor may at any time send a message to any current instructional program in order to activate the in-depth performance analysis routine for that program. The message will throw a software switch which bypasses the threshold checker, unconditionally activating the in-depth performance analysis program. The message is sent over the communications file. The proctor may want to do this whenever he observes a student with a problem which is not otherwise caught by the threshold checkers. See Section 3.3.4 for a discussion of the performance analysis routine.

3.6.4 Concentrator

The purpose of the concentrator is to allow several students to use a single line to the data base processor (DBP). Thus, the IIDA system will look like a single terminal to the DBP and each student will have the impression that he or she has a single direct line to the DBP. The general flow of the concentrator program is presented in schematic form in Figure 3.18.

From the point of view of the concentrator, there are three types of commands: those which need no set number conversion on the way out to the DBP or on the way back from the DBP (e.g., EXPAND); those which need set number conversion on the way back only (e.g., SELECT); and those which need set number conversion both on the way out and on the way back (e.g., COMBINE). Four data tables are maintained by the concentrator program for making these conversions. These are discussed in the following paragraphs as Tables A through D. Table D will be further amplified in terms of its constituent rows.

Table A. This is a list of user identification numbers, posted for each command in the order in which they are sent to the DBP. Two pointers are maintained. One is to the last command sent to the DBP. This is used for finding the place to update Table A. The other pointer is to the last response processed. This is used in assigning responses to the correct output files. A '0' is posted to Table A if the command originated with the concentrator. A minus sign on the user identification number indicates that set numbers in the response require conversion to user set numbers (i.e., the command was SELECT, COMBINE, or LIMIT).

Table B. This is a list of user set numbers, posted for each DBP set number in the order in which the DBP sets were created. The last DBP set number is held as a pointer to this list. Table B is used in converting set numbers in responses to SELECT, COMBINE, and LIMIT commands.

Table C. This is a matrix of DBP set numbers. Its rows correspond to user set numbers and its columns correspond to user identification numbers. Table C is used to convert user set numbers to DBP set numbers before commands like COMBINE and LIMIT are sent to the DBP. Pointers to the last set numbers are found in the third row of Table D.

Table D. This is a matrix whose columns correspond to user identification numbers and whose rows have the following different meanings:

Row 1. Output channel number. This may be an integer between '1' and '4', corresponding to the four message switchers. This data is picked up from the first message from the user on the corresponding input channel.

Row 2. File number. This is the DBP file number presently being searched by the user. This is the default file number until another file number is picked up from the argument of a BEGIN or .FILE command.

Row 3. Last user set number. This corresponds to the set number of the very last set created by the user. It is defined for each column in Table C by the last row with pertinent data. It is incremented by '1' for each user when the user creates a set.

Row 4. Count of response destinations. This is taken from the message tagging data and is checked with message tagging data that accompanies each new command. It is used in formatting the tagging data returned with each response. This number defines for the current column in Table D the last row with pertinent data in Table D.

Row 5 and following. Destinations. The remaining rows in Table D are codes for response destinations. As in the fourth row, this is taken from tagging data accompanying each command, is checked with tagging data for each subsequent command, and is sent with tagging data returned with each response.

Concentrator processing will be discussed below, closely following the flowchart of Figure 3.18. Each numbered paragraph below refers to the process box in the flow chart with the same number.

1. Students-to-DBP message handler. Each of the four message switchers handle a pair of communication files for the concentrator. These files with messages, or commands, destined for the data base processor (DBP) are continually checked for entries, using the common memory bit method discussed in Section 3.6.1. When an entry is recognized, its tagging data and the message itself are read by the concentrator.

Tagging data is reformatted, converting origin codes into destination codes and saving any additional destination codes in Table D. Experimentation will determine whether to read all messages before proceeding to the next step or whether to read only one message at a time. Here, as elsewhere, an on-bit in common memory will be checked and turned off before a block of data will actually be read into the buffer.

2. Command conversion for DBP. The origin of the message will be referred to as the user identification number. This will be used to

check Table D for the user's current DBP file number. If this differs from the current file number, a .FILE command with the user's file number will be sent to the DBP. This and all other concentrator-initiated commands will be noted by an entry of '0' in Table A.

The tables are updated as follows before the message is sent to the DBP: In every case, the user number is added to the end of Table A and the pointer to the last command is incremented by '1'. If a set is expected in response to the command, the user's last set number in Table D is incremented by '1'; the last DBP set number in Table B is incremented by '1'; and the last user set number in Table D is stored in Table B. If the command is of the third type, with set numbers that need changing before going to the DBP, the command is parsed and the set numbers from Table C are substituted for the set numbers in the command as given, making the command compatible with the DBP point-of-view on sets. For the last two types of commands, the sign of the entry in Table A is made negative to indicate that the set number in the response must be converted.

3. Set up commands for DBP. Each command will be written to a DBP queue file. This file contains no tagging data, although a directory on the file indexes the location and length of each command on the file. The DBP message flow control job learns from this bit that commands are waiting for it on the queue file. Action typically taken with respect to this bit is discussed in detail in Section 3.6.1.

4. Receive responses from DBP. The concentrator checks a designated bit in common memory to learn if any responses are waiting on the queue file coming in from the DBP. The DBP message flow controller posts the length and address of all such responses to the directory for the queue file. If responses are waiting, the directory is checked. Waiting messages are read off the file and then processed in the order in which they come off the file.

5. Response conversion for student. For each DBP response, Table A is checked for two things: First, if the sign of the entry in Table A is negative, the response must be parsed and converted back into terms of the user's set numbers. In this case, the response is information about a set. The set number returned by the DBP is used to point out the set number recognized by the user, stored in Table B. The user's set number is substituted for the set number returned by the DBP.

Second, the absolute value of the entry in Table A, beginning with the last response location, plus '1', indicates the destination of the response in terms of the user identification number. This user number points to the column in Table D in which the output channel number is stored in the first row.

6. DBP-to-students message handler. The appropriate message switcher is communicated the response, based upon the channel number in Table D. Tagging data, extracted from Table D, is written onto the appropriate output file, followed by the message from the DBP. Tagging data in-

cludes: message type code, length of remaining data and message, number of destinations, destination codes, origin code, and time since midnight. The response itself follows. When all messages have been written out, control goes back to Step 1, the student-to-DBP message handler.

7. DBP message flow controller. This job operates independently and functions exclusively to receive responses from and send commands to the data base processor (DBP). Commands for the DBP are recognized by a bit on in common memory. When it is recognized, the command is read off the file, using the length and address data directory, and sent to the DBP through the external communications buffer. This function alternates with the response receiving function. Here the content of the communications buffer is transferred to the response file, where it will be read by the concentrator. Again, a bit is turned on by the message flow controller to indicate to the concentrator that a response is waiting. If this job proves to be a bottleneck, it can be split into two semi-independent jobs: one for commands to the DBP and one for responses from the DBP.

Experiments will be performed to determine the most efficient way of sending commands to the DBP -- that is, whether to send one message at a time or several messages stacked together, and if stacked, an algorithm for getting the quickest response out of the stack each time.

The concentrator also acts as a filter for various housekeeping commands. In the DIALOG system these include: LOGON, LOGOFF, BEGIN, END, and .FILE. When a student sends any of these commands, he will get the standard response, not from the DBP but from the concentrator. The action will be totally simulated since actually conveying any of these commands to the real DBP could drastically affect the activity of all other users of the system. When BEGIN and .FILE are received with a file number as an argument, the contents of Table C and Table D are reset and standard Lockheed-like responses are transmitted back to the student at his terminal over the communications file. These standard responses can be read from a text-file associated with the concentrator, or they can be part of the program's code.

Following is an example typical of the effect two students might have on the concentrator's tables. In this example, Student 3 is attached to message switcher channel 2 while Student 4 is attached to message switcher channel 4. The tagging data in Student 3's commands indicate that only he is to receive DBP responses. The tagging data in Student 4's commands indicate that both he and the proctor are to receive his responses. The following commands are issued:

Command Sequence	Student Id no.	Command Issued
1	3	BEGIN 7
2	4	BEGIN 11
3	3	SELECT (arg)
4	3	EXPAND (arg)
5	4	SELECT (arg)
6	3	LIMIT (arg)
7	4	EXPAND (arg)
8	4	SELECT (arg)
9	3	SELECT (arg)
10	3	COMBINE (arg)

The four tables are updated after each of the above commands are read in. After all these commands are sent to the DBP through the concentrator, the tables appear as follows:

Table A	Table B		Table C (1)	(2)	(3)	(4)	Table D (1)	(2)	(3)	(4)
3	1	(1)	.	.	1	2	.	.	2	4
4										
0	1	(2)	.	.	3	4	.	.	7	11
-3										
3	2	(3)	.	.	5	.	.	.	4	2
0										
-4	2	(4)	.	.	6	.	.	.	1	2
0										
-3	3	(5)	.	.	.	.	.	.	3	4
0										
4	4	(6)	.	.	.	.	.	.	.	1
-4										
0										
-3										
-3										

Table A lists the sequence of inputs by student identification number. Since the students are on different data base files, 7 and 11, the concentrator must issue a .FILE command whenever the input changes from Student 3 to Student 4 or vice versa. This concentrator-issued command is indicated by a '0' in Table A. All commands which result in the creation of a set (SELECT, LIMIT, or COMBINE) cause the sign of the student identification number to be changed to a negative.

Table B lists the sequence of student set numbers. In the third command, Student 3 creates his set 1; in the fifth command, Student 4 creates his set 1; in the sixth command, Student 3 creates his set 2; etc.

The columns in Table C correspond to the student identification numbers. The first, third, fifth, and sixth sets created by the DBP are the first four sets created by Student 3. The second and fourth sets created by the DBP are the first two sets created by Student 4.

Table D accumulates miscellaneous data. The columns correspond to the student identification numbers. The first row indicates that DBP responses destined for Student 3 are transmitted over channel 2, and DBP responses destined for Student 4 are transmitted over channel 4. The second row indicates that Student 3 is searching file 7 while Student 4 is searching file 11.

The third row of Table D indicates that Student 3 has created four sets while Student 4 has created two sets. The fourth row indicates that responses to Student 3's commands have only one destination while responses to Student 4's commands have two destinations. The fifth and sixth rows indicate that responses for Student 3 are sent to him only while responses for Student 4 are sent both to him and the proctor, coded '1'.

3.6.5 Billing and costing programs

The purposes of these programs is to determine what to charge individuals and groups for their use of the IIDA system and to determine the cost of running the IIDA system. This is achieved by maintaining data on all usage of and costs incurred by the system. Because all durations of use must be monitored by the billing program, it is an appropriate place in which to initiate interprogram communications.

When a student logs in for the first time, the billing program will open a new student performance data base and will gather the basic data for that file. Every time thereafter that the student logs in, both his identification and the time he logs in will be recorded by this program. From the program the student chooses the instructional program he wants. The appropriate pair of logging-communications files will also be assigned by the billing program to the student's instructional program and to an available message switcher, i.e., one

with two channels available.

The names of these files will be recorded on a file commonly accessible to all the message switchers, as has been discussed in Section 3.6.1. When a student logs out, the billing program will record the time and will remove the names of logging-communications files from the commonly accessible file. This program will also record when the DBP is logged in, when it is logged out, and the duration of usage of each data base. The actual logging in and logging out will be done by the proctor, while the changing of data bases is done by the concentrator.

At first, charges will be based directly on connect time as a function of which PDP-11 program and which Lockheed data base is being used. A rate table will be set up for all the tutorial, exercise, and assistant programs. Also, a table of current Lockheed rates for each data base allowed by the IIDA system will be stored. With the exercise and assistant programs, at least the data base connect time rates will be charged.

Bills will be prepared on either an individual or group basis. Bills will include the date and time of each usage; the type and rate of each usage; the extension of each entry; and the total charge. These bills will probably begin as paper entries but will become real bills when the system becomes fully operational. At that time, billing will represent real charges. Income from it will finance the IIDA system. The paper entries, however, represent valuable data and experience which will serve as a basis for determining charges and rates when billing begins in earnest.

Costs to the IIDA system after it becomes fully operational will include at least the following: Lockheed and network charges; rent or amortization of the PDP-11 computer system; telephone charges; personnel salaries; rent, electricity, and other administrative overhead.

A cost summary report will be prepared on a regular basis. Cost per hour will vary considerably. A report on the distribution of the cost can be generated and can include a comparison with the system's income over the same time period. The break-even point in terms of the number of users and the type of use can be calculated for actual usage and alternative projections. Profitable periods can be determined and a sliding rate schedule can be established to attract more usage during less profitable periods.

3.7 Data Management Subsystem

A data management system is a set of computer programs that perform most of the functions of storing data on auxiliary memory

(such as disk), retrieving it on demand, making changes in content, and performing searches given appropriate specifications. In the context of IIDA, the data management subsystem is a set of programs that support other progr ms by providing some of these facilities. Reliance will be made to a great extent on the data management facilities provided with the computer operating system and some additional programs will be added to these. We have the advantage over many data management system users that, however complex the files in the DBP may be, the files within the instructional and communications processors are relatively small and simple in structure.

The IIDA data base consists of the communications log, the student performance history files and the text file.

3.7.1 File organization

The student performance history files are structured to allow for efficient data storage and dynamic growth. (See 3.5.1 through 3.5.7 for detailed descriptions of sub-files.) For reasons related to the compression of data and the indeterminate number of entries for certain fields, it is important to relieve the applications programmer from the task of unpacking elements from these files each time a data value is required.

The program which manages the unpacking of the history files is called whenever a student entry is a log on, request for /HELP, parsed command, or log off. The file is also unpacked during various update routines. When any of these initiating conditions are met, the entire performance history file for that user is addressed on disk and called into core. The blocks are unpacked so that each value resides in an addressable location (or array). This procedure consists of: (1) overlaying pseudo-structures on fixed length fields, (2) identifying and evaluating variable length tagged fields, and (3) manipulating entries in multiple occurring fields.

Each variable in the history file is then encoded and accessible by applications programs. If the calling condition is a valid command, the history file is maintained in memory until updated by the data base response parser. Otherwise the modified file is rewritten to disk.

The student performance data base will be organized in sequential manner, using an indexed sequential file organization. Records of this file will always be called either by student number, when retrieving the data for a particular student, or sequentially when performing an analysis across all student records.

The communications log is also an indexed sequential file. It is added to as a sequential file -- new data is added only at the end of the file. The file may be searched either sequentially or, by key based upon time. This would enable the log record for any given time to be located by a single command to the data manager program.

The text file is also an index sequential file. It is called for use or for updating always by frame number with which entries are associated.

3.7.2 Report generation

The standard IIDA reports are the student performance (discussed in section 3.6.3), traffic analysis (3.6.2) and costing and billing (3.6.5) summaries. These reports are generated by a series of specialized programs which use the student history data bases and the communications log as major sources of input. The reports are produced at regular intervals and upon request. The requestor may select from a menu of input options and output formats. The report print files are organized for flexible display; summaries may be directed to the line printer, hard-copy terminal or CRT screen.

1. Student performance report. The statistics appearing in this summary are useful for varied applications. A standard format for summarizing performance will be developed. Users of the report will have some control over which student records are included, but this will be limited both by the programming cost of generalization and privacy protection for students.

The student history report programs invoke the unpacking routine (discussed in section 3.7.1) which subsequently attaches the required student data bases. The contents of relevant variables are extracted and written to the preformatted print file on disk. Values are cumulated for later posting of summary data.

Some manipulation of raw data is done within the programs to provide more meaningful statistics. Similar data are combined, frequencies are tabulated, and means computed. The results are presented in matrices of varying degrees of complexity depending on the options selected.

2. Traffic analysis report. The traffic flow analysis programs compute all values needed for the traffic analysis report. Section 3.6.2 describes the data tabulated in great detail. The function of the report generator for this information is to build the print file and format the printed output.

3. Billing and costing reports. The billing and costing programs produce all data needed as input for the generation of reports. See section 3.6.5 for details. The report program draws these data from the billing/costing files and formats these data for output.

3.7.3 Ad hoc query

Ad hoc query is the capability for a user to ask a question of a data management system which has not been previously programmed.

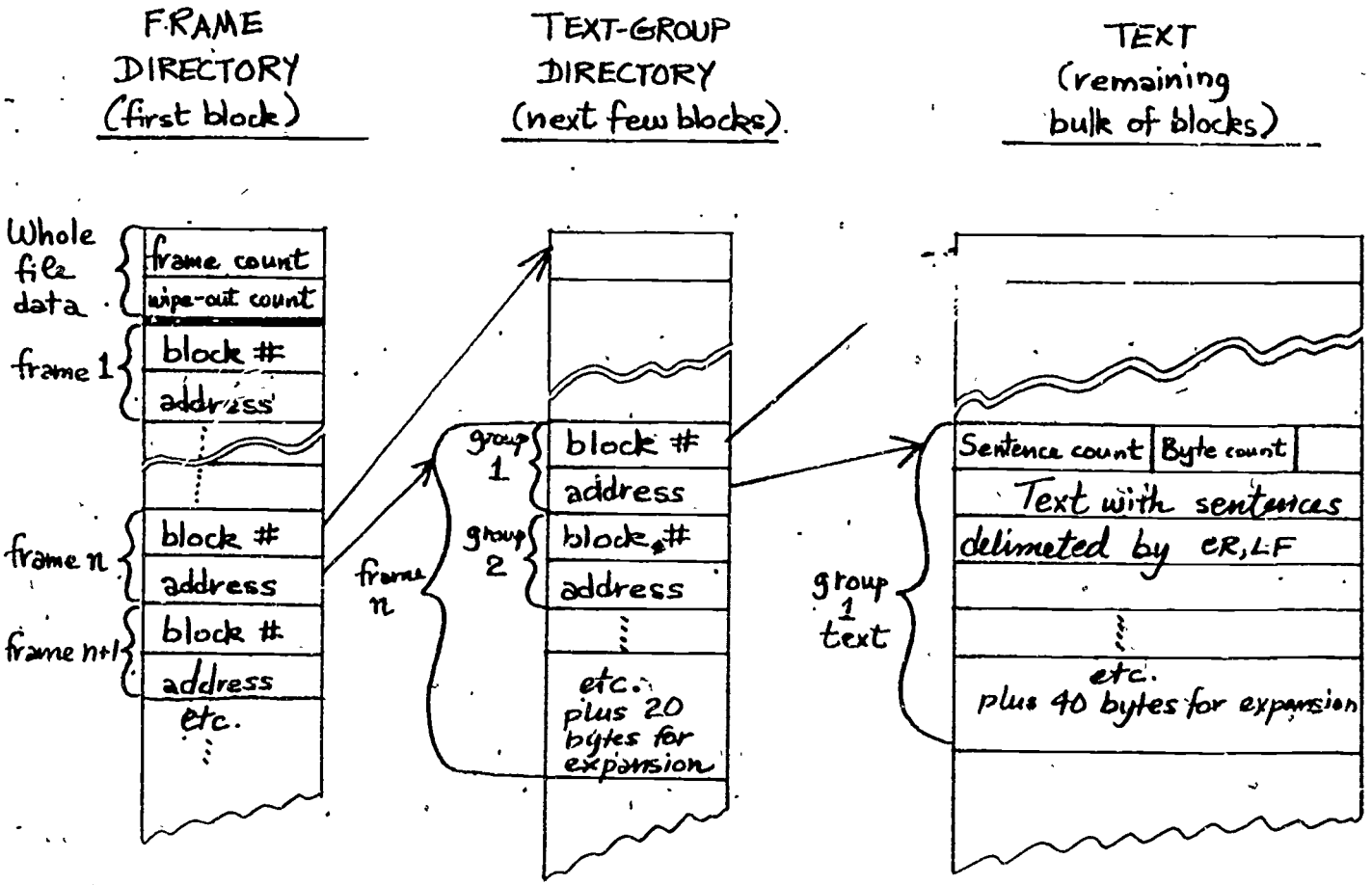


Figure 3.19. SCHEMATIC diagram of a typical text-file.

There will not be such a facility in IIDA, on the grounds of programming cost. Analyses of performance, communications activity and cost will be programmed as standard reports. These may, of course, be reprogrammed, if necessary. As new requirements arise, they may be programmed for analysis. But there will not be a program that permits IIDA users to define a new information requirement and have the resulting file search and formatting be done immediately.

3.7.4 Text files

A text file makes it possible to completely segregate the text used in each of the programs from the programs themselves. Changing and adding to the text become simple operations for the person who is writing the course. All textual entries from this file can be encoded into a few bytes on the logging file (see Section 3.6.2).

Each program in the IIDA system (the tutorials, the exercises, and the diagnostics) will have its own text file. For purposes of the text files, these programs can be described in terms of a number of frames, each of which is subdivided into a number of text-groups. A frame is a conceptually or structurally related unit of instructions or discussions. A text-group is any contiguous block of text within the frame.

A subroutine in each program will call for a single text-group from the file and print it at the terminal. This subroutine is also responsible for making a notation on the logging file that the text-group was printed. Two parameters are set before calling the subroutine. These are the frame number and the text-group code. The frame number may be set at the top of the frame so that only the text-group code need be set before every text-file subroutine call.

Data on the text-file need not be restricted to only information that will be printed. Data may be called by other subroutines to fill arrays which represent possible anticipated answers to a tutorial question. Branching to alternative routes through the tutorial may be based on the pointer to the array entry which matches the student entry. Similarly, various common misspellings may be set up in an array on the text-file and compared one at a time to the student response. Such alternatives will be set up as a text-group in the appropriate frame.

All files associated with programs in the IIDA system will be handled by the record I/O technique, the most versatile random access method under the RSTS/E operating system. Blocks using this method are 512 bytes (characters) each. Both fixed point numbers and alphanumeric strings can be found together on the same block. The very first block contains data that pertains to the entire file and a frame directory. The next few blocks contain text-group directories by frame. The remainder of the file, which makes up its bulk, consists of the text itself. Refer to Figure 3.19, a schematic diagram of a typical text file.

Two data items that pertain to the entire file are the frame count and the wiped-out byte count. The former contains the highest frame number for which there is data on the file. The latter contains the count of bytes on the file which have been discarded because of adjustments made to text-groups. The use of the wiped-out byte count will be discussed below under "reorganize."

The frame directory, also on the first block of the file, consists of pairs of numbers corresponding to, i.e., indexed by, the sequence of frames. The first element of the pair indicates the block number where that frame's text-group directory begins. The second element indicates the address on that block where the text-group directory begins. Given the frame number, the address of any pair in the frame directory can be calculated to the byte by : $d + 4*(f - 1)$, where d is the number of bytes devoted to data which applies to the entire file, found at the beginning of the first block, and f is the frame number. The number '4' in the formula derives from the four bytes allocated to each frame for the block and address data.

The text-group directory consists of pairs of numbers indexed by the sequence of text-groups. The first element of the pair indicates the block on which the text of the group begins. The second element indicates the address on that block on which the text of the group begins. The address of any pair of numbers in the text-group directory can be calculated by the formula: $a + 4*(g - 1)$, where a is the address of the text-group directory and g is the text-group code. If this code exceeds the limit of 512 bytes per block, the next block is read in and 512 is subtracted from the address calculated above, yielding the address on the next block. Twenty bytes are allocated at the end of each frame's text-group directory for up to five text-groups which may be added later to the frame.

The actual text is prefaced by two numbers: the sentence count and the byte count. Forty bytes are allocated at the end of each text-group for later expansion of the text. The byte count includes this expansion area. Text can flow from one block to the next. This is normal. It would be exceptional for a group of text to end with the end of the block. Lines or "sentences" are delimited by the two symbols: carriage return (CR) and line feed (LF). When these have been encountered the number of times indicated by the sentence count, processing of the text-group has been completed. Overflow from one block to the next can be handled by maintaining a pointer incremented by the length of each sentence. When this exceeds 512, the next block is called in.

Four programs are available for maintaining the text files. These programs are: Create, Add, Delete, and Reorganize. The first three make actual changes to the text on the file. The fourth, Reorganize, adjusts the file following a number of text changes in order to optimize disk utilization.

1. Create. This program allows whole frames and text-groups to be added to each of the files. The appropriate file is first indicated by the course writer and then established by the program. Then a frame number is entered and a text-group code is entered. Alternately, frame number or text-group code may be automatically assigned as an option. Line or "sentence" numbers will print on the left. A text-group is entered line by line by the course writer. The entire text-group is stored in core and the total length is calculated. The text-group directory is pointed out by the frame directory. For a new frame, a frame directory entry will be established. The next possible block and address will be picked up from the file data area on the first block. This information will be written in the text-group directory corresponding to the frame. The text itself will be written at the end of the file. Overflow from one block to the next will be managed automatically. Any attempt to write a text-group to an old frame whose text-group directory expansion area is full will be met with a message that the reorganization program should be run. This message will appear before any text is entered for the group.

2. Add. This program allows sentences to be added anywhere into already existing text-groups. The course writer indicates the file with which he is working. The program establishes the appropriate file. The course writer enters the frame number, the text-group code, and the line number following which additions are to be made. If possible, up to two lines immediately prior to the addition are printed with their line numbers for verification and reference. Additions up to several lines are entered. Line numbers are printed for each line added. An option set at the beginning of the program will allow the remaining lines in the text-group to print, with their new line numbers.

All added lines and all lines following the addition are stored temporarily in core. When adding lines is completed, this core-stored text is written back into the text-group area if there is enough room for it. If there is not enough room, the text of the entire group is rewritten at the very end of the text file. The appropriate corresponding changes are made to the text-group directory. The number of bytes thus freed are added to the wiped-out byte count.

3. Delete. This program allows sentences to be deleted from any text-group. The course writer indicates which file he is working with. The program establishes this file. The course writer enters the frame number, the text-group code, and the line numbers to be deleted. An option will allow an entire text-group to be deleted at one time, if that is necessary. The lines to be deleted are printed, with their line numbers, and the course writer is asked if he is sure he wants these lines deleted. Upon verification, the sentence count parameter at the top of the text-group is adjusted and the sentences following those deleted are moved up and over those which were deleted. If the entire text-group is deleted, the block number and address in the text-group directory are set to zero. The number of bytes thus

freed are added to the wiped-out byte count.

4. Change. There is no program which directly changes lines of text. Rather, changes are accomplished by deleting lines and then adding lines with those programs designed to do these functions, discussed above.

5. Reorganize. This program adjusts data on the file for optimal disk utilization. If a text-group directory is full and more text-groups are to be added to the frame, or if the count of wiped-out bytes exceeds a set percentage of total bytes in the file, a recommendation to run the reorganization program will be issued under program control in one of the other maintenance programs. Reorganization essentially involves copying the entire text file over into another file, by changing any directory parameters necessary in order to restore 20 bytes to each text-group directory expansion area, and to restore 40 bytes to each text-group area. All areas wiped-out in the old text file will be eliminated from the new text file. A first pass will determine how many blocks should be allocated to the text-group directories. A second pass will perform the actual copying of textual data. Directory data will be stored in core on the first pass and written out to the frame and text-group directories on the second pass.

3.8 Systems and Programming Support

In this section we briefly describe the mechanics of linking the IIDA computer with the data base processor. This is all done with standard commercial hardware and software and is made simpler by the fact that the IIDA computer appears to the DBP as just another terminal. No messages are exchanged between these computers that are different from messages between the DBP and any user terminal. Similarly, the DBP looks to the IIDA computer as a terminal, although IIDA expects different kinds of messages from the DBP than from a student terminal.

Both the establishment of the computer-computer link and writing its associated programs, and the restart-recovery program are tasks during the implementation phase. In addition to these, which have specific designs and goals, there is a third general support task which is to provide operating systems consultation to project programmers. One member of the staff must become expert in the operating system used and be in a position to advise the others, or to get advice from the manufacturer, on detailed use of the operating system and compiler.

3.8.1 Computer-to-Computer Linkage

The IIDA PDP11/34 and the DIALOG data base processor will be interconnected via a switched network (either TYMNET or TELENET). Functionally the network and host processors and the IIDA computer will view each other as ASCII terminals operating in a full duplex mode with odd parity. Communications will occur at a transmission rate of 120 cps.

The interconnection of the two computers will occur upon user demand, that is, when the first user in a day, or the first in any continuous session, issues a command to the DIALOG data base. (Only this user would experience a delay while the link is being established.) The following steps would be followed to establish the interface.

The proctor, as a privileged user, will run various utility programs under the RSTS/E operating system to effect communications. A port will have been designated solely for communications with the data base processor. This surrogate terminal will be configured during the system initialization phase causing the operating system to recognize pre-specified characteristics (such as baud rate) every time connection is made on that line.

The proctor will then manually dial the switched network via the local service network over ordinary telephone lines. If the response is a busy signal, a rare occurrence, the proctor will send a message to the user informing him that the system is unavailable and advising him to logoff. This and other proctor messages will be recorded on a text file and transmitted following proctor input of a transmission code.

A successful connection will be confirmed by an audible tone. Communication thus established, the proctor will place the receiver in the acoustic coupler. He will then process a file which calls for a series of commands that will carry out LOGIN procedures on behalf of the "pseudo" terminal. Other commands will call, run, and detach the message flow controller routine. An example of establishing such a command file, referred to as a control file by DEC, follows:

Command	Action
LOGIN KB1: 1,5	RTST/E will login terminal 1 under account number 1,5. The operating will look up the corresponding password.
FORCE KB1: RUN $TTYSET	Will process the command to set terminal characteristics for terminal 1
FORCE KB1: KB1:	Will cause OS to recognize that terminal characteristics are being set for keyboard 1
FORCE KB1: VT050B	RSTS/E will recognize "pseudo" terminal as a DECSCOPE for example.
FORCE KB1: EXIT	Signifies the end of the TTYSET routine
FORCE KB0: RUN $MSGCTL	Run message flow controller from the console
FORCE KB0: /DE	Detach message flow controller program
END	End of control file

The signal received from the network computer will be converted to an ASCII code through the modem, transmitted through the multiplexer, and recorded in the incoming message file. A bit in common memory is turned on to indicate that a DBP response is awaiting action. The message flow controller recognizes this bit as an unsolicited response indicating that the network machine is awaiting login data. The front end of this program then acts in the manner of a "network access machine" by sending appropriate access protocol messages which will establish network and host computer communications. Two sets of messages will be available for transmission depending on whether connection is attempted through TYMNET or TELENET. (A minimal number of commands are required for access; therefore they will be stored as literals within the context of the message flow program rather than as separate files to be opened and read.)

These are examples of network access messages to be stored and transmitted:

TYMNET	TELENET
terminal id	(CR) (CR)
LRS (CR)	terminal id
DIALOG	C 415 20 (CR)
Dialog password (CR)	Dialog password (CR)

The access sub-routine will scan network or data base responses as they arrive in the incoming message file to ascertain whether the computers have been successfully linked. It is possible that either the network or host processor will refuse the connection. The program will be matching messages for anticipated responses. Having detected an unexpected response, the program will send that message to the console to be examined by the proctor. Proctor input may supersede the programmed access procedures. The proctor may inform the message controller to retransmit all or part of the access messages, to wait n minutes and retry, or to discontinue attempts. He may also enter his own messages for transmission. If the connection cannot be made, the proctor will advise users that the network/DBP is temporarily "down".

When the connection has been made and Lockheed DIALOG is online, control for transmitting data to and capturing data from the DBP will then reside with the communication processor's applications programs.

3.8.2 Restart and Recovery

A system crash, power brown-out or other malfunction involving the PDP11/34 may result in disruption of student interaction with IIDA programs and the data base processor. The RSTS/E operating system will generally recover automatically with programs and data files intact; restart from the point of failure can theoretically be easily effected. This is accomplished by the operating system performing several tasks immediately upon failure detection such as closing all files and saving the contents of core memory and the general registers.

However, whatever data in the user's buffer that have not been put out to disk will not be preserved. There are likely to be, at any given moment, several simultaneous users whose history files will be in various stages of update and analysis. the RSTS/E automatic restart facility cannot guarantee that a user entry has been read from the input buffer and processed.

Therefore, a specialized IIDA restart program will be run upon recovery. The student will be advised that there will be a slight delay before he is once again conversant with IIDA.

The restart program will then reissue the sequence of valid user commands from the first to the latest completely updated item. The log file fill provide the necessary data for resubmission. Commands will be stacked and sent to the data base processor. Data base responses will not be transmitted to the user but utilized to restore the history files. Data from the traffic log such as relevance ratings will be maintained. Times accumulated during the restart run will not be recorded since they are not meaningful for performance analysis purposes; rather the original times will be transferred from the log file to the history file.

To aid in restarting after failure, certain procedures must be followed during update routines. A bit will be set when every update to the history file has been completed in response to a command. By examining this bit the restart program will know and can tell the student whether a command needs to be reissued.

Thresholds may be crossed while re-processing commands. The restart program will respond by bypassing PAR calls and resetting threshold counters. In effect, the routine will do the bookkeeping without actually performing the analysis.

A crash during a PAR run can be detected by examining the update completion flag. Here the user must reenter the command since history files may be in a partially updated state. Counters will be decremented accordingly.

At the conclusion of the restart run the student will be informed of the final command fully processed. He may continue the search from that point.

4. IMPLEMENTATION PLANNING AND EVALUATION

4.1 Overall Utilization Plan

It is important to distinguish between two types of planning for promoting the use of IIDA: (1) facilitating and promoting initial use in situations where the primary concerns are to prove the concepts, to obtain the feedback necessary to improve the design and operation of the system, and to gather support in the user community; and (2) marketing of the system where the objective is to m imize use and to achieve enough income to make a profit, support future research, or at least sustain the system on a self-paying basis. Our present concentration has been on ways of introducing the system into a user environment where it will be appreciated, from which we can learn, and which can lead to support when income-oriented marketing begins in the future. This is necessary in order to refine the system and develop a knowledgable and loyal user group. If the system is to be eventually sold on a broad scale it will be sold as a result of proven performance. This proof is best manifested as a result of gradual, steady, planned growth.

Inherent in the need for the feedback mentioned in the previous paragraph is the recognition that IIDA is still an experimental concept. Although IIDA's designers are convinced of its feasibility and utility they recognize that successful commercial exploitation of new products often requires experimenting with the ways in which users will accept the idea and possibily revising the original ideas.

While our major focus has been on planning ways to get started with initial use we have not neglected planning for the marketing of the system on a cost-effective basis. Our approach here has been to look for a situation which, following a diffusion-of-innovation model, will allow us to select an initial user group which will not only serve to provide useful feedback on the design of the system, but which will also provide us with an entry point into a broader user group. Thus the initial user group should also serve as a means of educating a broader potential user group about the system and as a means of diffusing knowledge about IIDA.

4.2 Possible Initial User Groups

When IIDA was first conceived, we assumed that users would be almost exclusively industrial. Our rationale was based upon two factors: (1) the enormity of the potential user group--somewhere between one and three million "scientists" in the country; and (2) the then-assumed cost. We felt that only industrial users would be able to afford the training cost, that those who could or would go to classroom training should do so, and that this form would be more cost-effective.

While ultimately this potential user group should make up a fairly high percentage of the users of IIDA, there are several factors which suggested to us a different group of potential users should be targeted as our initial user group. First, Arthur Elias at BIOSIS, and John Creps at Engineering Index have both suggested that their experiences indicate that industrial scientists are less receptive to innovative information programs than are faculty and student groups.

A second reason for change is that the cost of data-base search service is coming down as a result of competition, increased markets and improved technology.

A third reason is that our technical design work indicates that the concentrator will enable more than one user to share a line to a data-base computer. We hope to be able to negotiate lower rates for data base services when working through a concentrator.

Fourth, student users are also more attractive as an initial user group because, as they move into industry, if they are satisfied users as students they will be likely to continue to use the system in an industrial setting and to promote its use by colleagues who have not used it.

Finally, although it is nearly universal in universities today that conventional computational services are provided by a central service in ample quantities to academic users, search services are rarely made available in this way. Currently a break in this pattern is being negotiated at our own university and we believe that eventually bibliographic search services are going to be made available to academic departments, for both faculty and student use. In addition, we believe that this practice will be adopted elsewhere because the service is needed and will be eventually recognized as necessary. The only question is when it will happen.

As a result of these changes and a positive attitude on the part of educators, we now feel that academic users should be the first target of our attempts to test the system. An additional factor which suggests that academic users should be the initial target for preliminary use of IIDA is the existence of the Experimental Partnership for the Reorientation of Teaching (XPRT) program, a consortium of five engineering colleges, which is headquartered at Drexel and funded by NSF. The degree of interest expressed by the XPRT people at Drexel leads us to believe that this group on this campus provides an ideal initial user group.

When integrated into the ongoing activities of the students and faculty involved in the XPRT program, the use of IIDA by this group will provide us with the needed feedback in the design of the system. In addition it provides a natural mechanism for educating potential users about the system and for diffusing knowledge about IIDA. Expanding the use of IIDA into other aspects of the engineering curriculum at Drexel and expanding the use of IIDA into the entire

consortium of schools represented in the XPRT program would provide a widening of operational experience and act as the forerunner of a full-scale marketing program. This approach also allows for the possibility of the diffusion of IIDA use into industry. This could occur not only through active marketing, but also through the diffusion of satisfied users into various industrial activities.

4.2.1 Utilization at Drexel

The initial primary user group will be senior level and graduate level students in the various engineering curricula at Drexel. For this user group we will use the Engineering Index's data base Compendex. Entry into this user group will be gained in two ways: (1) contacts established with the faculty involved in the XPRT program; and (2) courses in technical writing for engineering students. Initial users will be approached via XPRT since this group on the Drexel compus provides a structure and set of on-going activities into which IIDA can be integrated with maximum impact and minimum disruption. The faculty selected for participation through this route will be those involved with the XPRT project and hence committed to innovative teaching methods. In addition the XPRT organization will provide a forum for discussion of how well the system works and how it can best be integrated into the overall curriculum. Discussion so far indicates that data base searching should probably not become a separate course. Rather it should be offered as a service to engineering students who would use it in connection with senior design projects or graduate theses. Students would be given assignments involving bibliographic searching and given the option to do searching through IIDA or completely through the library.

The second entry point involves courses in technical writing. A recent revision in Drexel's engineering curricula is the inclusion of a required course in technical writing for all engineering students. Discussion with the faculty presently on campus who are scheduled to teach the technical writing courses indicates enthusiasm for the IIDA system on their part and a willingness to work with us in incorporating IIDA into their courses as a service to the engineering students in writing term papers. We also have reason to believe that some of the engineering faculty not associated with the XPRT program would be receptive to this approach.

It will be possible to arrange workshops for faculty to provide a background in the project to a broader range than just those faculty who are active in initial participation. These workshops will provide feedback from IIDA users and help to build an extended network of users and potential users at Drexel who have shared the experience of IIDA use. Case histories of users can be collected and written for dissemination in the workshops, describing how others with similar backgrounds and needs have approached and used the system. These case histories can also be used in contacting students and faculty who are unable to participate in workshops.

4.2.2 Extension to other schools

The next step involves extending IIDA usage to the engineering classes and faculties in the XPRT member universities, where there is a need for bibliographic searching, using both workshops and case histories. The experience gained on the Drexel campus, and the already established liaison between the XPRT participant campuses, should greatly facilitate this extension. Once IIDA is in use on the XPRT program campuses the next step would be to extend IIDA usage to other schools in the mid-Atlantic region and then ever further.

4.2.3 Extension to industry

We also expect to be able to extend the usage of IIDA into industrial settings. Smith, Kline and French Laboratories have indicated a willingness to participate in tests of the system and other local industrial users may also be sought. Once the IIDA system has been put into use at Drexel and the other XPRT consortium schools an increasing number of engineering graduates going into industry will have had, we hope, positive experiences with IIDA and its capabilities. We expect that many of these users will be likely to want to continue to use the system in an industrial setting. These satisfied users will provide a potential word-of-mouth advertising network which should prove to be particularly valuable when potential industrial users are approached with information about the uses and benefits of the IIDA system. In addition the widening of operational experience and the marketing experience gained in extending the use of IIDA throughout the XPRT consortium and other schools, taken in conjunction with the already proven performance of the system, should prove invaluable in developing a marketing strategy for introducing IIDA to users in industry.

4.2.4 Pricing policy and support assumptions

The utilization and acceptability of IIDA depend upon price as well as on quality of performance. Price, in turn, depends upon costs, manner of distributing costs among users, financial support from NSF or other sources, and volume of use. In this section we discuss the variables affecting cost to the user. A specific pricing plan will be developed in the next development phase.

1. Costs. Costs may be divided into two broad categories: pre-operation or development costs, and operating costs. In all planning for a pricing policy, we are assuming that it will not be necessary to recover NSF-granted development costs out of operating income.

The operational costs exclusive of overhead are:

Computer (The IIDA minicomputer): lease or purchase, maintenance and spare parts if not leased

Programming support for the IIDA computer: maintenance programming and system modification

Tutorial and analytic support: the continued analysis of performance and improvement in design of the system

Evaluation: continued evaluation of performance

Communications: communication links between users and the IIDA computer and between the latter and the search service.

Search service: the costs of using the data base processor

2. Income sources. The following are the sources of income:

Student use fees, paid by themselves or their employers or educational institutions.

NSF subsidy, especially to assist in establishing the IIDA service and supporting it to the point where it is viable.

Vendor support: largely in kind, this can be realized through no-cost assistance in development and operation and, possibly, in reduced costs of operation for students or when using a concentrator.

3. Volume of use. Factors affecting volume of use are:

Venture capital. This system is not being developed as a profitmaking enterprise supported by venture capital. There are no funds for advertising and other promotional activities except through normal professional publication and discussion channels.

Developmental nature of the project: When first offered to test users, the system will not be complete and fully operational. Initial users will be test subjects and this will clearly affect their willingness to pay for services. The more the product is proven and publicized in service, presumably the greater the willingness of users to pay for it.

Initial use will be restricted to one data base and one search service. Expansion beyond that point will make the system more attractive to users but cost an added increment for each new data base or search service.

University central support. We have indicated previously the importance of university policies on central provision of computer services for the entire campus. The is common today for computational use of computers and, in the typical situation, users feel they are adequately supported in the computer needs, which are not charged to college or departmental operating budgets. Search service has not reached this condition yet and, typically, is charged to operating budgets of individual departments and colleges. This clearly reduced use of bibliographic search service, whether IIDA is involved or not. As universities are brought to recognize this use of computers as equal in importance with others, this situation will be alleviated and more funds will be available for bibliographic searching.

Laboratory fees. Many universities are beginning to drop the concept of laboratory fees, in effect paying these out of overhead which is then distributed among all students. Thus, in some universities, of which Drexel is an example, we do not have the option of charging a laboratory fee for use of the system, and this any affect level of use.

4. Price considerations. Aside from the need, eventually, to break even within all the considerations of cost, income and volume of use described above, the following are the major questions affecting the price that should be charged:

University or employer support. Will the university or employer agree to buy IIDA services for its students/employees?

NSF support. Will NSF support IIDA until it is able to become self-supporting?

Vendor price policy. To what extent will vendors of data bases and search services support IIDA through assistance in consulting, promotion and special usage rates?

When should IIDA try to break even financially?

Should the operational IID support continued research and development?

4.2.5 Specific goals to await more detailed study and development during the implementation phase.

During the implementation phase of IIDA development there are several goals which must be accomplished before beginning initial use of the system. Several of these are listed below:

- Survey of engineering and other faculty who work with engineering students and who might use this system in their courses.
- Selection of courses and faculty to work with.
- Joint planning with selected faculty for actual use.
- Development of a more detailed plan for extending use beyond initial test group.
- Preparation of training materials for use in connection with workshops, conferences, etc., aimed at encouraging new users.
- Development of a more detailed plan for formative evaluation at the end of initial use.
- Development of a more detailed plan for broadening the user base beyond the initial user group.
- Development of a more detailed plan for summative evaluation of IIDA impact on broad user group.

4.3 Preliminary Testing

It is our intention to perform mechanical testing of the system at all stages of the development of the IIDA system. Some testing of the model system has been conducted during the design phase and this is discussed in Section 5. Another system test will be conducted near the end of the implementation phase as the computer programs are brought together for integrated system testing. There are two kinds of testing which need to be performed--mechanical testing or de-bugging, and system testing.

4.3.1 Instructional programming testing

The educational programming aspects of IIDA involve the logic of course design and the actual programming of the tutorial, exercise, and assistance programs into the language of the instructional processor. During the implementation phase of the IIDA project the detailed course design must be examined for internal flaws or inconsistencies and for conformance with the original design. These programs must then be pilot tested on a small group of naive volunteers before attempting to use them with the initial users.

4.3.2 Computer programming

The computer programming aspect of the implementation phase involves systems programming (installing and testing the operating system, data communications packages, if needed, and any special data communications programs written specifically for this project), programming of message store and forward and concentrator functions of the communications processor, programming of a billing algorithm (for allocating the cost of the computer system use among users, depending on actual facilities used), and programming of diagnostic and student data base processors. Each of these systems must be carefully tested for mechanical flaws. Once again, the operation of at least some of these programs must also be pilot tested on a small group of naive volunteers before attempting to use them with the initial users.

4.3.3 System testing

Once each component of the educational and computer programming packages of IIDA has been de-bugged on an individual basis they must then be assembled and given a thorough system test. Once the mechanical aspects of system testing have been completed the total system must be pilot tested on a small group of naive volunteers before attempting to use it with the initial users. While pilot testing of the entire system must await completion of the total package of programs we expect that we can and will be able to do some of the pilot testing, as indicated above, with naive users on some of the components of the full system prior to actual completion of the system. In addition some of the aspects of the full system can be completed in advance of the date the full system is ready by doing simlulation testing with the model system completed during the first phase of this project.

4.4 Evaluation

Although we will eventually want to submit the IIDA system to both formative and summative evaluation, it does not make sense to implement both types of evaluation at this time. While the system is still in its early stages of development, formative evaluation aimed at monitoring the progress of system development and providing useful feedback seems most appropriate. We do, of course, also want to plan for the summative evaluation which should be performed at a later time.

As with any evaluation process, one should start with the goals for the system at this stage of development. There are basically two major goals which the IIDA system is designed to accomplish: (1) To provide an effective method of teaching and assisting in interactive bibliographic searching; and (2) To construct a workable, technically sound computer based system which can be successfully marketed. These are both long range goals which cannot be evaluated at this time. These are the goals, which a summative evaluation must address.

The dependent variable at this point was "user satisfaction"; an alternative dependent variable is "user skill" in operating the system. Measuring user skill in this type of experimental system is critical.

The role played by IIDA begins to change as users become more familiar with the characteristics of the system and the mechanics of searching. At some point the teaching role of IIDA will become subordinate to the assistance role. As searches get fully underway IIDA will be available for suggestions and help about how the system can be best used to accomplish search tasks. IIDA may be called upon to act as a translator of search strategy. Also should the searcher get into difficulties which we can identify in advance IIDA may suggest options to the user which have not yet been employed or exploited fully. Finally IIDA will continually monitor the performance of the user to see if he is having problems with the system, even volunteering help should the occasion warrant it.

As these developments take place and as the project moves up the ladders outlined in Figures 4.1 and 4.2, the dependent variable will change to impact upon the user. This impact measure will be described in Section 4.4.4, on plans for evaluation. At this stage given the state of project development, we have been most concerned about operationalizing user satisfaction and skill.

In order to operationalize the concept of "user skill", one must reflect carefully on the dynamics of interactive learning through IIDA. This dynamic can be quantified (operationalized) through several routes. One of these, for example, could be an analysis of the messages between IIDA and the user. Presumably the user's commands, errors, and requests for help can be classified into several categories such as the following: learning, procedural, requests for review, etc. While not specifically developed for this purpose, the ability to detect the skill or the user, and his errors, is, in part, being developed in the diagnostic routines. Skill error ratings can be accumulated and updated as a function of a number of variables (e.g., commands used, elapsed time, requests for information, pattern of commands, etc.). These indices can be used to describe the proficiency of the user in several aspects or different phases of system usage. Not only do the diagnostics allow IIDA to adapt its behavior to the demonstrated skill of the users, but they also will allow us to assess the development of learning. The adaptation of IIDA to the user is also a reflection of the user's proficiency with the system. Over time we expect to test for the extend to which users take advantage of the full capacity of the system.

While the dynamics of learning about interactive searching will be complicated to study they should be systematically related to the evolution of the search and to the reactions of searchers to the system. Although we will initially only be able to study users who have done interactive searching on a limited and short term basis, future analysis of long-term usage could reveal whether the dynamics of learning continue as people routinely search with IIDA.

It should be clear that these two major goals have a number of assumptions behind them which preclude evaluation now: (1) an effective teaching method must be evaluated relative to other methods available to users and potential users; and (2) no comparable assistance system exists; (3) a workable, marketable system must also be evaluated relative to other bibliographic search systems.

4.4.1 Evaluation at this Stage of Project Development

At this stage of project development several milestones have been met (See section 1.): (1) A model system has been built; (2) The system has been tested with a small group of users; (3) Plans have been made to implement an operational version of the system,; (4) Preliminary planning to market the system has begin and (5) Plans have been made to test the IIDA system with various potential user groups.

The measures which can be used to evaluate the success of our project team in teaching these goals fall into several categories: (1) Archival measures-- to what extent have we met our projected goals. This can be done through simple paper and pencil records; (2) Unobtrusive measures-- these measures will attempt to test for the initial effectiveness of our system with various user groups; (3) Technical measures-- those designed to measure actual system performance from a technical perspective; and (4) Impact measures-- the effect of the system on the users who have been exposed to it up to this point. Each of these formative evaluative measures is designed to help us in the further development of the system. In addition, they provide indicators to the National Science Foundation for the progress made to this date.

IIDA will need the benefit of formative evaluation during the early years of development and initial use: feedback on how well it is able to teach, on user acceptance, on the techniques developed to promote its extended use, and on user reaction independent of its actual success in affecting user behavior. Once the system has been developed and initial use begun, limited summative evaluative evaluation can be conducted on a number of bases which will be gradually expanded to include what will, we hope, be the long range evaluation of the impact of IIDA. Long term, summative evaluation will be based upon: (1) The effectiveness of teaching: Have users learned to search acceptably well? (2) Use of the system: Is IIDA actually being used? and (3) The user population: Have we, in fact, attracted new users to searching science bibliographic information rather than converted others to a new mode of searching? This section of the report discusses a number of considerations in the overall evaluation planning for IIDA and the specific plans for both formative evaluation prior to initial use and summative evaluation plans for the initial user group.

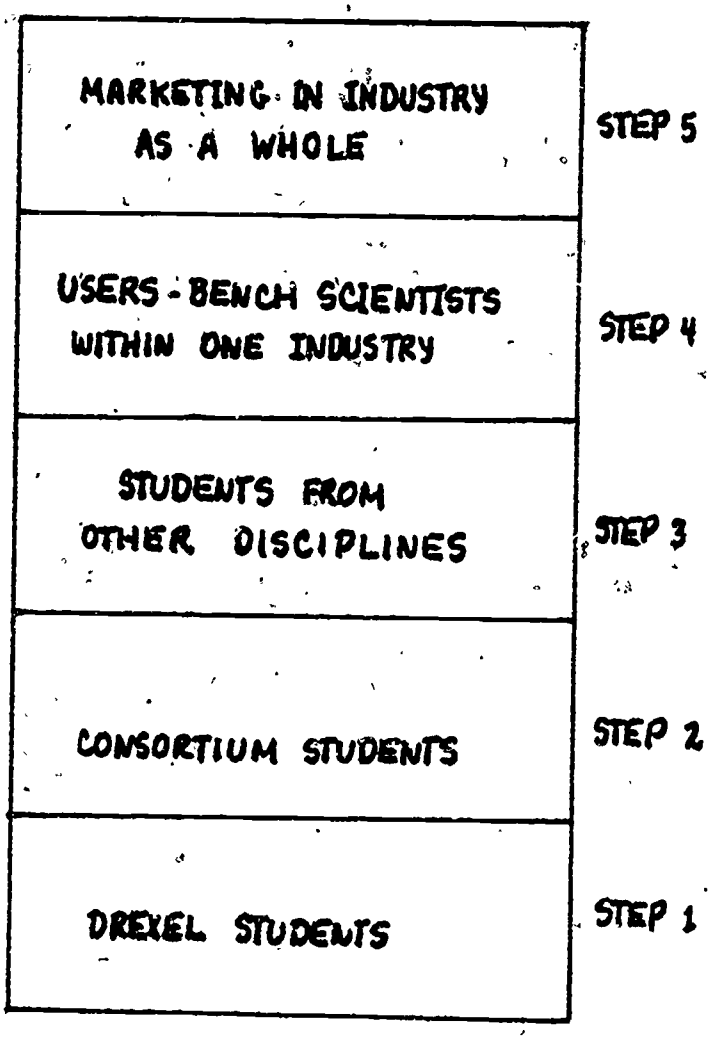


Figure 4.1. Ladder of Generalizability: Impact on Individuals

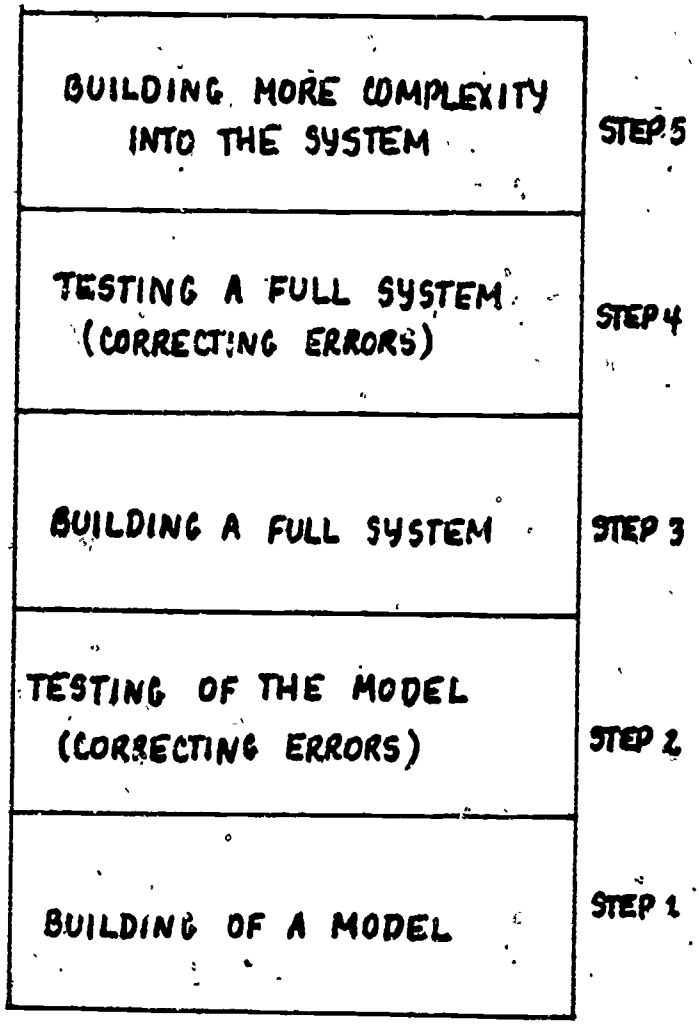


Figure 4.2. Ladder of Generalizability : Technical Development of IIDA

4.4.2 General Considerations

In general, we feel that it is helpful to think of evaluation as being tied to distinct levels of generalizability; i.e. the results can be applied (with confidence) to some level of generalizability. At this time, for example, prior to full-scale testing with any user group, we cannot generalize our results on teaching effectiveness beyond the limited number of searches made on a pilot basis by students at Drexel University. However, in terms of the technical development of the system we can make statements at a higher level of generalizability.

Thus, in terms of our own thinking, we distinquish between statements concerning the technical capabilities of the system, and statements concerning the impact on students and the effectiveness of IIDA as a teaching and consulting methods.

In terms of the overall development of this system, it may be useful to thinking of the interaction of system development and evaluation in terms of a ladder of generalizability (see Figure 4.1) As indicated in Section 4.2, we have shifted our initial emphasis on user groups from industry to the university. Indeed, in testing for the effectiveness of IIDA, we will use university students at Drexel from the engineering school as our first user group. We then hope to move to students from the five engineering schools who are part of the Experimental Partnership for the Reorientation of Teaching. From there, it will be possible to move to testing students from other disciplines. Then, we can begin large-scale testing the IIDA system on groups of industry users-- first one industry and then others. Preliminary testing with industrial groups will go on in parallel with academic testing.

Clearly, as we move from step one on the ladder to step five, we can have greater and greater confidence concerning the statements we are able to make with respect to the effectiveness of IIDA as a teaching method and the impact of IIDA on the productivity and efficiency of individual users. Similarly, as we move further up the ladder we can move to more complex and sophisticated methods of evaluation.

In terms of the technical development of the system, we can also think of levels of generalizability (see Figure 4.2). At the beginning of the project, IIDA started as a model (Section 5). We then tested the model, and are now ready to build a full system. With the full system in place, we must then begin to test the system with our designated "potential users". Finally, we can begin to perfect the system and add degrees of complexity into it.

In terms of the evaluation of this project, it should be clear that the IIDA system must be developed to the implementation phase (the building of a full system) before full scale evaluation (of a formative or summative type) can be launched.

4.4.3 Evaluation Variables

As the discussion to this point indicates, most of our evaluation efforts to this point have been in the form of planning for the future. There has been some evaluation of use of the IIDA model and this is discussed in Section 5.

It should be clear that our dependent variable in this case can be operationalized through the use of computer records. Some examples of the type of information which can be derived from the computer records kept by the IIDA system include:

i. Connect time
ii. Search time
iii. Connect time/search time
iv. Kinds of user errors
v. Number of user errors
vi. Kinds of commands used
vii. Number of commands used
viii. The way commands are used
ix. User familiarity with commands
x. Sequences or patterns of errors and commands
xi. Focusing of search (e.g., search directed commands/undirected commands)
xii. Repetitions of instructions
xiii. Repetitions of command descriptions
xiv. Frequency of HELP
xv. Kinds of HELP used
xvi. Patterns of HELP used
xvii. Changes in kinds of HELP over time
xviii. Prompts or indications by IIDA of availability of HELP
xix. Number of suggestions by IIDA
xx. Kinds of suggestions by IIDA
xxi. Patterns of suggestions by IIDA
xxii. Changes in kinds of suggestions over time
xxiii. Growth of completing of search strategy
xxiv. Costs of searching

4.4.4 Future Evaluation Plans

In general, both the formative and summative evaluations of IIDA at different stages of the development of the project can and should be conducted using several different types of techniques for measuring the variables of interest. The kinds of measures which seem applicable are (1) observation, (2) self-report, (3) and records of student and system performance over time. In a number of cases some of these kinds of measures can be collected unobtrusively so that the user is unaware of the fact that measurements are being made or so that while the user may know that measurements are being made he will not necessarily know what the measures are intended to measure.

1. The dependent variable. As was the case in testing the IIDA model, we plan to continue to test for user satisfication an skill. This will be done through the survey instrument

and through some interviews conducted by project staff with users concerning their judgments about the problems with the system. In addition, we will have project staff members monitor the interactions between users and IIDA. The staff will determine what aspects of the program users are having difficulty. However, as the user group becomes larger we will have to rely more upon the computer system itself to monitor interactions with the system.

However, with the full system in place, we plan to add an important dependent variable: impact upon the users bibliographic searching habits. Do users continue to use IIDA? Do they use other computer based bibliographic searching techniques? How do they compare the relative efficiency of IIDA versus other methods? Is their productivity and efficiency increased through (or influenced by) the use of IIDA?

This very important dependent variable will be operationalized through a combination of survey research techniques, observation, and labortatory experiments. The survey research techniques combined with interviewing will be used to ascertain user's attitudes with respect to the efficiency and desirability of IIDA versus other bibliographic searching techniques (both manual and other computer based systems. For a small sample (N=50) of users, we can also use this technique to ascertain information about their productivity. We would propose to interview this small sample several months after they have adopted IIDA into their regular research habits. These users would then be questioned concerning: (a) the number of hours they now spend on bibliographic searching; (b) the number of hours they used to spend on this activity; (c) the types of activities they were able to "take-on" as a by-product as a reallocation of time (it is also possible they wuld take on fewer activities); (d) in the case of researchers, the kinds of substantive activities they were able to perform (e.g., new articles, experiments, reports, etc.)

In the case of repeated uses of the IIDA system, we would rely upon observation of users' behavior--through the computer system and through monitoring by our project staff. This could give us an indicator of changes in user skill over time. We have discussed the specific measures for this in an earlier part of this section.

It would also be quite useful to conduct a laboratory experiment with the consortium of schools who have agreed to be potential users of this system. Similar experiments could be conducted in industry.

The experiments would be designed to give the same search assignment to a group of approximately 75 potential users. 25 would conduct the search through IIDA; 25 through some on-line bibliographic searching system; and 25 through a manual search. These three groups could be compared along many of the variables described above: length of time for conducting search, number of titles found; number of relevant titles found, satisfaction, skill in searching, etc.

We would then propose to follow-up with each group and test for the impact of this experience upon their overall searching procedures.

Such a laboratory experiment would be quite costly in terms of resources and staff. We are, therefore, simply outlining it at this point in time as one possibility for future evaluation.

4.4.5 Factors which may influence user's attitudes

In formative and summative evaluations conducted throughout the various stages of IIDA development and use extension we must concern ourselves with the effect of IIDA on the user. It is also necessary to recognize that a number of characteristics of the user, the system, and the interaction between the two are likely to make a difference, from one user to the next, in how the system influences the user's attitudes about and his ability to learn from and effectively use IIDA. So far we have been able to identify a number of variables which may influence user responses to IIDA. While not all of these independent variables will be of importance in all stages of formative and summative evaluation of IIDA it is desirable to have them listed in advance for use as a guide in planning both kinds of evaluations:

1. Physical characteristics of IIDA.
   - i. Characteristics of the terminal
   - ii. Ease of access
   - iii. Reliability
2. Operating characteristics of IIDA.
   - i. Content of tutorials and exercises
   - ii. Sequence of exposure to tutorials and exercises
   - iii. Form of messages
   - iv. Nature of feedback to users
   - v. Management and control of messages
   - vi. Time and speed of message handling
   - vii. State of computer and phone link
   - viii. Access procedures
   - ix. Costs
3. Interaction between IIDA and user.
   - i. Specificity and clarity of operating principles
   - ii. Frequency of searching
   - iii. Amount of pre-planning for search
   - iv. Training procedures for IIDA use
   - v. Ease of access to relevant literature via IIDA
   - vi. Ease of access to alternative search systems
   - vii. Quality of results with IIDA
4. User.
   - i. Physical characteristics (e.g., lack of typing skill, physical impairments, etc.)
   - ii. Familiarity with terminal, computers, interactive searching, etc.

iii. Attitudes towards learning and working with computer assistance
iv. Personal incentives
v. Personality characteristics
vi. Work and life style
vii. Perception of need or desirability of working with literature
viii. Strength of desire to be "in touch" with literature
ix. Attitudes toward other users (e.g., colleagues or peer group)
x. Perception of self with respect to other users
xi. Attitudes toward task
xii. Personal search patterns--i.e., preconceived or acquired notions of how searching "should" be done.

5. The nature of the task or search.
i. Divisible vs. unitary
ii. Maximizing vs. optimizing (most information vs. best information)
iii. Specificity of goal
iv. Number of alternate methods of goal attainment
v. Time constraints
vi. Need for closure or completeness

6. Social or group influences.
i. Colleague or peer group--as implicitly defined by the user--attitudes toward IIDA
ii. Colleague or peer attitudes towards searching and the need for it
iii. General organizational structure of the group
iv. Individual's role within the group
v. Organizational climate
vi. Group size
vii. Group definition of importance of the task
viii. Reasons for the existence of the groups
ix. Financial limitations imposed by the group.

Ultimately, a summative evaluation should be conducted to test for the impact of IIDA among diverse groups of users-- with different levels of system complexity (see Figures 4.1 and 4.2) For the moment, we have concentrated upon formative evaluation for purposes of system development.

4.5 Applications of IIDA

In developing IIDA it has been our belief that the concept is general in nature and is applicable to a wide variety of problems other than the teaching bibliographic searching. However, since IIDA is not the only project which impinges upon the possible uses of IIDA or IIDA-like systems discussed below, it is reasonable to believe that in time the various techniques will be fused and the resulting systems will not be exactly the same as any of the present day ones. In this sense we view IIDA as a precursor of a family of systems that will be implemented to help people use computers and devices which are attached to computers. They will save countless hours of dull and often ineffective oral instruction or the publishing and reading of ineffective texts.

4.5.1 Training of intermediaries

Although IIDA is being developed initially with a single primary user group in mind--the end-users of scientific and technological information--we believe it will also be of use in initial and refresher training of professional intermediaries. These intermediaries receive varying amounts of training and varying degrees of on-the-job experience in bibliographic searching. The small likelihood of the intermediaries' being thoroughly familiar with any given discipline is one of the reasons why some end-users of the services they can provide do not like to use intermediaries or libraries. If this lack of familiarity with the discipline of the end-user is coupled with inefficiency in searching owing to inadequacies in training or to lack of experience with searching the problem is made more severe. However, the fact that the intermediaries are information professionals and have some training should make them receptive to IIDA-like training and to the use of the IIDA diagnostics while performing routine searches for clients. It is our intent to continue the project as originally designed and, in addition, to bring in occasional professional intermediaries for testing. This will allow us to guage their receptivity to possible eventual use of IIDA. It will also enable us to begin building up performance data on experienced searchers relative to non-experienced searchers as part of our attempts to continually refine and improve the performance of IIDA in helping users learn not only the mechanics of searching, but also how to search more effectively.

4.5.2 Training of IIDA users in the use and appreciation of their professional literature

In general, science students receive little training in the structure of the literature of their fields and its use. Working scientists have probably picked up some of this on a hit-and-miss basis but do not, in general, have the time to fully investigate literature structure and use through brute force methods. While IIDA, as the design is presently projected, will provide training in one aspect of the use of professional literature, there is no coverage of how a literature is organized, how publication operates, or how a research person keeps up with a field (not, by any means entirely through searching or use of a library).

This leads us to believe that one reasonable extension of IIDA would be to develop course material with practical exercises on the structure of the literature or the simulation of the publication and diffusion of knowledge might be developed. At the undergraduate and graduate student level these could be added to the curriculum of various sciences as assignments that can be given for independent study. For the working scientist who wishes to learn more about these topics, the availability of these materials though IIDA would provide the opportunity, if desired, to learn or work through the material on an ad lib. basis at a relatively low cost in time and effort expended.

4.5.3 Coping with language differences

Differences in the languages used for cataloging and indexing have plagued all library automation efforts. Attempts have been made to 'translate' between classification schedules or thesauri, but not with any notable success. Currently attempts are being made to develop a universal search language. There seems to be no reason why this should not be successful since the functions of search services vary relatively

little even though the externals appear to vary widely. Given that a universal language can be developed, it could be implemented either by the individual search services, in lieu of or in addition to their own languages, or the universal language could be translated by an intermediary computer, such as that of IIDA. At the same time of course, the other IIDA services of teaching and assisting in bibliographic searching could be provided.

Regardless of how a universal language is brought into being, there is no conflict between such a language and the use of IIDA. IIDA can be used to teach and assist in the use of whatever language students and search services may wish to employ. Further, if the approach to a universal language is made by using a translator from it to a vendor language, then it can be expected that there will be areas of ambiguity in the translation. If there are, then the IIDA interactive techniques will be exactly what are needed to assist in the resolution of the ambiguity.

4.5.4 Teaching and assistance in other subjects

An IIDA-like system could be useful in teaching and assisting in the use of any highly structured activity performed in a computer or which is controlled by a computer. Examples of such activities are: use of statistical packages (sets of computer programs) on a computer, especially by non-experts in the field, who are using the package programs for output, not for learning how they function; performing control functions through a computer, such as laboratory experiments and analyses whereby the intermediary computer assists in deciding what to do; and in the use of large scale simulators. In many fields of research, statistical and simulation programs are used, much as were slide rules formerly. Similarly, many laboratory procedures can be either simulated or controlled in actual execution through a control computer. In all of these cases the user wants to process data and use the results as quickly as possible. Individualized instruction plus assistance on-line is an ideal approach. Using computer assisted teaching and mediating programs such as IIDA can save time and materials as well as reduce errors and turn around time while increasing capacity and flexibility for the individual user.

4.6 Dissemination of Information about IIDA

Some of our plans for disseminating information about IIDA have already been discussed under other headings in Section 4. These involve: (1) the development of user-oriented workshops and case history material; (2) the gradual expansion of the user base through the development of a loyal user group and a record of proven performance; and (3) the development of an industry-oriented marketing campaign. One aspect of this third route for dissemination of information not discussed above is continuing professional education. Drexel University has a long history of being industry oriented both through its cooperative education program and through the development of a broad spectrum of two or three day continuing education seminars to introduce a number of topics to people working in industry. It is likely that at some stage of IIDA development we might wish to make use of this mechanism to introduce IIDA and its capabilities to potential users in industry.

5 THE MODEL SYSTEM

5.1 Objectives of the Model

A small, working model of IIDA has been developed with two objectives: (1) to demonstrate that the techniques proposed for IIDA are feasible and (2) to aid in the design of the full system by requiring designers to face and solve operational problems which one might tend to gloss over if no operating model were required. It was not an objective of the model system to provide an educational service to any users. The model is not a complete teaching system and, as a result, it cannot be used to measure the ability of IIDA to teach.

For the sake of economy of operation and administrative simplicity, a data base processor written by a member of the Drexel staff was used. This program called IRSYS, emulates, in many ways, the DIALOG system of Lockheed. Anyone familiar with DIALOG can learn to use IRSYS in a very few minutes. The file to be used was a set of records taken from Air Pollution Index, prepared by the Franklin Institute Research Laboratories for the Environmental Protection Agency. Selection of this particular file was also on highly pragmatic grounds: ease of availability, knowledge of the file by an IIDA programmer who had to convert record structure to conform with IRSYS requirements, and the assumption that almost any test subject or user of the IIDA model system would be able to formulate a meaningful question to the file without special knowledge or background.

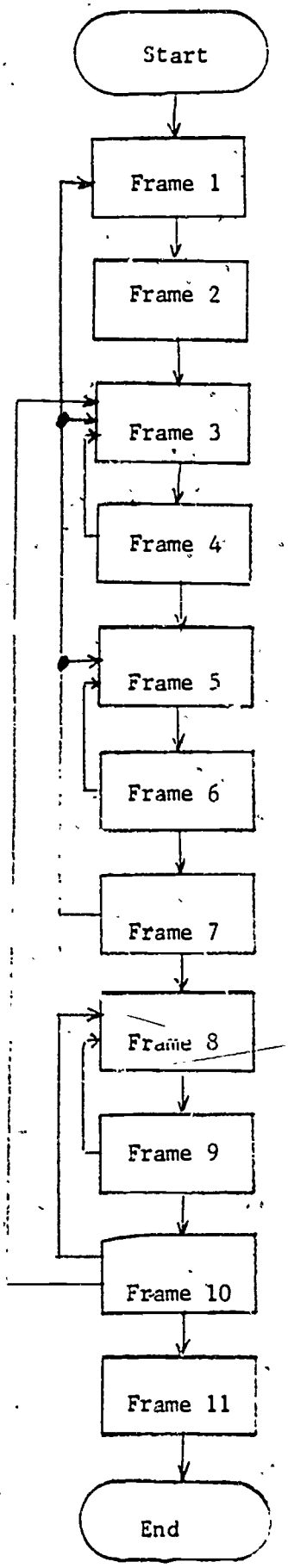
A PDP-10 computer was used for all three processors, with the interaction between them being simulated. Thus, CP, IP and DBP were each designed to operate as if the other processors were in a separate computer. This slowed response time somewhat, but timing was adequate for demonstration purposes.

5.2 The Model Curriculum

Two typical programs and an exercise were implemented. The tutorials were data base structure and search mechanics.

5.2.1 Tutorial programs

This portion of the model serves to illustrate how tutorial material in general will be designed in the full implementation. The subject matter in this case was restricted to explaining the commands and data base used with the model DBP. Some use was made of the IIDA capability for the IP to compose commands to the DBP based upon what the student has input or on what the course author needs. For example, if the course author wanted to display an example of a thesaurus display, rather than laboriously preprogram the entire display, he could cause the IP to issue an EXPAND command to the DBP and the display was gen-



Enter free text statement of search topic, to be stored for later analysis.

Record, not in computer, descriptors student plans to use.

Stage 1. Enter only the commands SELECT, EXPAND or RELATE. Descriptors chosen by the searcher.

Verify that at least one of each type of command called for in Stage 1 has been used. If so, pass control to Stage 2. If not, ask student to use the missing commands.

Stage 2. Enter only COMBINE commands, using sets created during Stage 1.

Verify that at least one new set has been created with a COMBINE command.

Display set history and choose from among these choices: restart, back to stage 1, back to stage 2, go on to Stage 3.

Stage 3. Enter only PRINT commands. A relevance judgment will be called for for every record printed.

Offer student option to review more sets or to proceed.

Student assesses his progress, elects to go back to an earlier point or go ahead.

Student assesses entire search

Figure 5.1. Schematic of model exercise program.

erated and transmitted from the DBP to IP to CP to student terminal. Normally, the kinds of student responses were so well restricted by the teaching material that the diagnostic programs are not needed.

5.2.2 The exercise

The goal of the exercise was to give the student the experience of conducting a search of his choice with tutorial guidance and assistance. For this reason definition of the search topic, descriptors used and depth of the search are under the control of the searcher. The system maintains control over the order of search stages and constrains the student to sample each type of search command at appropriate times. The vital HELP function is available on request, and analysis of ungrammatical or inappropriate commands is automatically performed.

The order of search commands follows the model of a search phases discussed in Section 1. These are: (I) exploration of descriptors available and choice of which to use, (II) combination of descriptors into more complex concepts, and (III) browsing in the resulting groups of items and printing of items of choice.

The student is allowed to leave Phase I when he considers himself ready, subject to the following conditions: (a) he may only pass from Phase I into Phase II and (b) he has used, at least in a cursory manner, each of the search commands available in Phase I. Since the model recognizes the iterative nature of interactive searching, when the student feels ready to leave Phase II, he is offered several options for which phase to enter next. After insuring that the searcher has sampled the commands of Phase II, the system offers options ranging from starting all over again (prior to Phase I) to continuing on without recycling (Phase III). In Phase III, the system requests revelence judgments for each of the records printed. If the searcher feels unsatisfied with Phase III, another opportunity to return to an earlier phase is presented. This option is presented in recognition of the increased information concerning the search available to the student on completion of reviewing some of the results of his search. It is also possible to decide the search is finished on completion of his first excursion into Phase III.

Using the guided exercise, the student is able to complete a search of his choosing to his own satisfaction. Help is available on request. The orderly progression of the search from one phase to another is insured without limiting the inherent flexibility of interactive searching. By requiring the student to try each command type in the context of a real search, we give him a better feel for their value than we can by out of context descriptions of function. A simplified flow chart of the guided exercise is shown in Figure 5.1. Figure 5.2 shows, in general, how diagnostic subroutines are used to monitor individual student commands.

An input, presumed to be a command, is received (a) from the student by the CP. It passes it to IP where the first check made (b) is to see whether it is a 'slash' command, i.e., one ordering a change in the sequence of instruction -- indicating that the student needs help or wishes to stop the course, for example. If it is not such a command, the input is sent (c) to the DBP.

The DBP response is received (d) by the IP (via CP) and is parsed. If the response indicated (e) that the command was invalid, which is denoted by a message beginning with ERROR, the command is parsed (f).

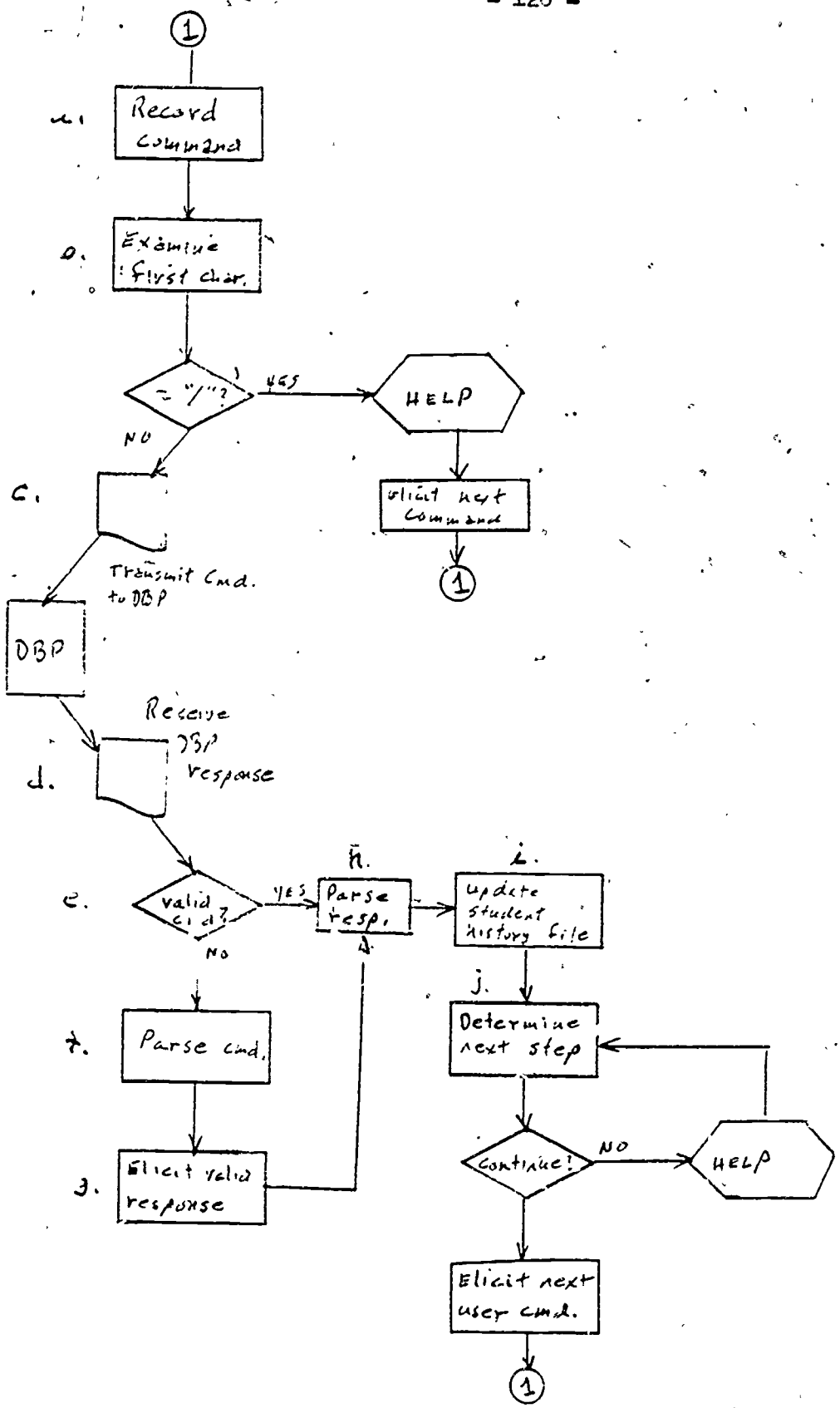


Figure 5.2. Schematic of model diagnostic logic.

Based upon the analysis of the command, the exact nature of the error is determined and the student is asked (g) to reenter it. This conversation may repeat several times until he finally sends a valid command or is referred to the human proctor and is terminated from the teaching system.

If the command parser program is able, finally, to get a valid command, it goes to the DBP and the latter's response is parsed, as above (h). Presumably, since the command this time was verified before it was sent, no error message will result.

When a valid command has been executed, the results of it are entered (i) into the student data base containing the record of his performance. Then, the IP determines (j) what it wants to do next -- where to branch to provide tutorial material or to elicit another student input. This branching decision may be complex. It may use any of the information in the student data base to assist in making this decision.

The guided exercise is composed of 11 frames. The first two frames require the student to define the search topic and put in writing the descriptors considered most likely to be useful in the search. In a non-teaching environment, these steps are usually accomplished before logging on to the search system. IIDA includes these steps to ensure some prior preparation before beginning a search.

Frame 3 presents the mechanism for Phase I, browsing the thesaurus, confirming the presence of desired descriptors, and selecting those descriptors which define the scope of the search. The commands to be used are SELECT, EXPAND, and RELATE. (RELATE is the equivalent of a Lockheed EXPAND following an earlier EXPAND -- calling for display of semantically related terms.)

Certain frames are present in the guided exercise for checking purposes. The student is unaware of these frames unless he fails to use one of the allowed commands in the proper frame. Frame 4 checks to see that the searcher has used all of the commands available in Phase I before allowing him to proceed to Phase II.

Frame 5 enables the operations of Phase II, the combining of descriptor sets formed in Phase I to create more complex search statements. The only search command to be used in this frame is COMBINE. Frame 6 is a checking frame to make sure the COMBINE command has been used at least once before allowing the searcher to proceed.

A branching node is found in Frame 7. The searcher is provided several options consonant with the nature of interactive searching. The search may be restarted or a new search initiated (return to Frame 1); more descriptors may be selected or more browsing through the thesaurus allowed (return to Frame 3); additional COMBINE commands may be entered (return to Frame 5); or the search may proceed into Phase III (Frame 8).

The only search command available in Phase III (Frame 8) is the PRINT command. The searcher requests printing of one of his sets. The system requests a relevance judgment on each record before the next record of the set is printed. Frame 9 allows the searcher to enter a PRINT command for another set, and switches control back to Frame 8. Frame 8 again requests relevence judgments of the records of the set. The searcher indicates when he is satisfied with the number of records viewed.

Another branching node occurs in Frame 10. In this frame the searcher may return to Phase I (Frame 3), Phase II (Frame 5), Phase III (Frame 8), or declare the search complete. When the search is declared

complete, Frame 11 requests a relevence judgment on the search as a whole. Following entry of the relevence judgment, the guided exercise is ended.

When the searcher requests help at a branching node or at another input time, a choice of areas for help is offered:

1. More instruction on what to do now.
2. Definitions of valid commands.
3. Reveiw of commands and search stages.
4. Review of sets created.
5. Review of records already viewed.
6. Review of descriptors used.

Option 1 gives meaningful comments on each frame of the guided exercise. Different assistance will be offered depending on what search stage the searcher is currently in and will take into account what progress has been made. Option 2 is tutorial in that valid search commands are explained on order.

The review of commands and search stages provides a listing of all the commands issued up to the point of requesting help. The display is in the following format:

Stage	Line	Command
1	1	EXPAND SMOKES
	2	SELECT COSTS
	3	RELATE ENGINES
	4	SELECT PARTICULATES
2	5	COMBINE 2 + 4
	6	COMBINE 2 * 4
3	7	SELECT POLLUTANTS
	8	SELECT REGULATION
4	9	PRINT 5/3/2
	10	PRINT 8/3/5

A stage is identified as a string of commands all belonging to the same search phase. For example, the commands of stage 3 are those appropriate to phase I. The stages are numbered consecutively. Using this table as a reference, the system discusses each stage. First, the stage is identified and a summary of the actions of the stage is given. For example, in stage 2, sets were manipulated with logical operators. Searcher input is requested as a free text comment on what was being accomplished in this stage and searcher satisfaction is queried. If the stage seemed unsatisfactory, the searcher is asked his opinion of why. After reviewing each stage, the system queries the searcher on his plans for continuing and asks to which point in the search he wishes to be returned.

In requesting the searcher to identify his goals in each of the stages, the searcher is presented with an opportunity to rethink his strategy and arrive at his own solution. The potential for system comments on misuse of certain commands is clear. Examples are trying to COMBINE sets with no items, or to SELECT descriptors that do not exist. However, system directives on which command or descriptor to use next are out of keeping with the philosophy of enabling the searcher to solve his own problems.

The review of sets created includes an analysis of sets created by the COMBINE command. Such sets are presented with the original des-

criptors, not

SET	COUNT	COMMAND	DESCRIPTION
9	2	COMBINE	3 * 8
			= (1 * 2) * (5 + 6 + 7)
			= (COSTS * POLLUTANTS) * (SMOKES + ENGINES + SAE AUSTRALAS)

The searcher picks out what it is about the sets formed that dissatisfies him.

1. Too many items.
2. Too many irrelevant items
3. Too few items
4. Expected items did not show up.

The suggestions from the system at this point are typefied by the response to dissatisfaction with a set with too many items. Suggested changes in strategy include the use of RELATE or EXPAND commands to find items more exactly on the search topic and the use of the logical AND(*) to eliminate extraneous items.

The review of records viewed first offers a short analysis of the command which generated the set viewed. A detailed definition of a COMBINE command much like that of the review of sets is included. If the searcher wishes to view the records again, he will be queried on relevence judgments and informed if his evaluation has changed since the last viewing. The average relevence for the set is recalculated and compared to the previous average for the searcher's information.

The review of descriptors used displays each of the descriptors used and how they were used.

DESCRIPTOR USED	USED IN SELECT	SEEN IN EXPAND	USED IN RELATE
COSTS	X		
PARTICULATES		X	X
POLLUTANTS	X		
SMOKE	X		
ENGINES	X		
SAE AUSTRALAS	X	X	

Since the system keeps records of what has been seen, meaningful comments can be made at this point regarding missteps such as selecting a descriptor which does not exist when the searcher has viewed that section of the thesaurus where the descriptor would appear if it existed. The need for verification of descriptors is stressed.

5.3 Model Software

The model has been implemented on a PDP-10 computer, with all processors in the same physical machine, but communicating among themselves in the manner they would if in separate hardware structures. The programming language has been BASIC which is not ideally suited to this form of programming, but was a common denominator among project staff members and is the language in which the model data base processor was previously written.

5.3.1 Instructional processor

No special system software has been used in the model. Rather, courseware has entirely been written in Extended BASIC. The instructional processor structurally consists of a short trunk program sup-

ported by an extensive group of diagnostic subroutines. The trunk's function is to cue and control command input and to call subroutines as they are needed. The most prominent subroutines are listed below.

SLASH SUBROUTINES	OTHER DIAGNOSTICS
SLASH DETECTOR	STUDENT FILE UPDATE
/DONE	COMMAND PARSER
/QUIT	DBP RESPONSE PARSER
/HELP	PRINT ROUTINE

A "slash command" is acceptable whenever a search language command is expected, and at other input times. Valid slash commands include: /DONE, /QUIT, /HALT, and /HELP. The slash detector subroutine follows each command input. If a valid slash command were entered, control transfers to the appropriate subroutine. A code indicating which slash command was entered is set before control returns to the instructional processor.

1. The /DONE command is used by the student to indicate the end of any intra-frame cycle. It is not a subroutine, strictly speaking, since its processing only involves the IP branching out of the frame when the /DONE code is recognized.

2. The /QUIT command clears all pointers and files accumulated on the student search and terminates the program. Thus the student begins a completely fresh search the next time he enters the system.

3. The /HELP command calls in the program's largest diagnostic sub-routine. It in turn depends on other subroutines for its operation. Five options are initially available with /HELP: (1) further instructions for the current frame; (2) definitions of all valid commands; (3) a review of commands and search stages; (4) a review of sets created; and (5) a review of records already viewed.

In more detail:

(1) Paragraphs corresponding to each frame of the trunk are stored in /HELP for its first option. These paragraphs describe in detail what is expected of the student in each frame. When this help is requested, the appropriate paragraph is sent to the student.

(2) Paragraphs describing each of the five commands valid in the model are stored for the second option. These paragraphs explain the commands' functions, abbreviations, argument formats, and expected data base responses. The student selects one or more command(s) whose paragraphs are then sent to the student. In the full system, all search commands will be described.

(3) All valid commands given in the search up to this time are printed in a table. They have been analyzed into three stages as described by Penniman, and into several transitional stages in which commands are mixed. The student is asked to state his objectives for the commands in each stage. If he feels that those objectives were not met, he is asked to state why not. When the review is completed, a plan for proceeding with the search is requested from the student. All prose statements are captured on the communications log for later analysis by the human proctor.

(4) Data about the created sets is printed in a table. This includes set number, count of records in set, command text, analyzed COMBINE arguments, and average relevance of viewed sets. The student is asked to indicate those sets which came closest to his

search objectives. It is assumed that if any sets were fully satisfactory, this /HELP option would not have been selected. The student is thus asked why each of the indicated sets is not satisfactory -- are there too many records? too few records? too many irrelevant records? etc. Advice on revising the search strategy is then given, based on this last set of responses.

(5) Each set viewed up to this point is described in terms of the command which generated the set. If that generating command were COMBINE it would be broken down into its basic textual components. For each such set, the student is given a choice to review or not to review records which have already been viewed from that set. If a review is chosen, it will be in the same format and in the same range as in the original PRINT command. For each record thus viewed, a relevance judgement is solicited. If it differs from the previous relevance judgement, this is brought to the attention of the student. If dissatisfaction with the set is indicated, the reason for this dissatisfaction is elicited and advice is given for proceeding.

Each of the five /HELP options described above are terminated by the student choosing to: (1) return to the courses; (2) sign off temporarily; (3) sign off permanently; and (4) get more help by returning to the top of the /HELP subroutine.

5. The student file update permanently stores data on the student's search activities. Data is stored for both valid commands and errors. This subroutine is generally called immediately following a DBP response. Data in this file is used in various diagnostic routines, both for analyzing and for describing the student's progress.

Data stored about valid commands include: text of command; command code; command stage code; set number created or viewed; command argument text; indication of multiple use of any argument; count of records in each set; average relevance of viewed sets; basic textual components of COMBINE commands; text of first and last EXPAND terms; count of valid commands; and a relevance-format chain for viewed records. Data stored on errors include: attempted command (if identifiable); error code; and where the error occurs in the command input sequence.

6. The command parser completely analyzes student commands. All command components are broken out and stored in temporary arrays. This subroutine can be called by parts of the IP to provide a basis for thoroughly discussing the command as given by the student. Errors that occur here are identified precisely and are coded for storage in the student file.

7. The DBP response processor completely analyzes responses from the DBP. All response components are broken out and stored in temporary arrays. Cryptic error messages from the DBP cause the command parser to be called. This parser provides a basis for a thorough discussion of the DBP response.

8. PRINT routine was developed because the system design requires the PRINT command not to be communicated verbatim to the DBP. This is because the student must make a relevance judgement after each record is viewed. Thus the PRINT command given by the student is broken down and PRINT commands requesting a single record at a time are issued to the DBP.

The routine begins with instructions that parse the PRINT command as it is given by the student. Any errors are explicitly pointed out; data about errors are recorded on the student file; and the command is entered again. Following the display of each record, individually

requested from the DBP, a relevance rating is solicited. This is recorded on the relevance-format chain. All other elements required by the student file are duly posted.

5.3.2 The data base processor

The data base processor used in the model is a program called IRSYS which was created by a Drexel graduate student for other purposes. It resembles the DIALOG system in its command structure, but has fewer commands. Some of them are more versatile than DIALOG's for example if a set number 3 has been previously defined, IRSYS permits the usage COMBINE 3* CAI to enable users to make use of both descriptor selection and set combination in one command. IRSYS also permits string searching, in the fashion of SDC's ORBIT. However, for the purposes of the IIDA model, only a minimal language is taught to the student. The major weakness of IRSYS for this application is that imposed on it by the PDP-10 operating system for BASIC programs -- that only one user terminal can make use of any file at one time. Thus, the model is a one-user-at-a-time system.

5.3.3 The communications processor

In the model, the CP performs a minimal role. Because all the programs are resident in the same computer, it is present only to make the model design conform with the full system design wherever possible. As implemented, all communications between the IP and the DBP pass through the CP and all communications with the users pass through this processor also. The CP also logs all messages it handles, which may be used for analysis of performance after a student has done a search.

Because all three processors are resident in the same computer, the CP imposes no significant delay in response by the IP to student input. In the full system, inter-computer communications will be a critical timing factor.

5.4 Preliminary Evaluation

In addition to the design and mechanical testing of a model system we have also done some limited pilot testing of IIDA through use of the model to simulate the full scale system. Our feeling was that it was necessary, as early as possible, to begin testing out the operation of the model system. This was done for both formative and summative evaluation purposes. Actually using the model with naive users offers us the possibility of detecting conceptual design flaws in the model which might be carried over into the full-scale system if not spotted in advance. In addition it makes it possible to begin to get a feel for user reactions to the system.

The basic procedure used in testing the system involved recruiting naive volunteers who were paid for their time. The volunteers were 4 graduate students from the Drexel Graduate School of Library Science and 8 undergraduates, 5 of whom were from various Drexel engineering programs. The remianing 3 undergraduates were enrolled in behavioral science programs on the Drexel campus. In all cases the volunteers were given an overview of the goals of the project and how the model system fits into long range development plans. Below are the instructions given to each user prior to actual use of the system.

The present IIDA system is a model of some of the component parts of a projected larger system and is designed to demonstrate

the feasibility of actually developing a full-scale system. The full system will provide a mechanism for teaching bibliographic interactive information retrieval. It will also serve as an on-line consultant for users. When completed IIDA will contain tutorials on such topics as search strategy, the structure of a data base, the data base command language, etc. It will also contain an exercise which requires the learner to go through the stages of an actual search while demonstrating a working knowledge of the data base commands. In addition IIDA will contain an assistance program which monitors and provides on-line consulting for the searcher.

Having done some preliminary testing of the model we would now like to get outside feedback about the model. As presently structured the system to be demonstrated consists of two parts, a tutorial on a command language developed for use with this model, and an exercise. The data base available for searching while using the system consists of eighty records and a micro-thesaurus from the Air Pollution Index. After you have seen or worked through IIDA please feel free to give us any comments or advice you may have on the model, or the projected full system, so that we can improve the design of the model, and ultimately, the full system.

In watching or working through a demonstration of the model we would appreciate it if you could pay particular attention to such questions as: How effective is the tutorial in teaching the basic commands of the model command language? How effective is the exercise in leading the student through the stages of a search? How effective is the exercise in requiring the user to demonstrate a working knowledge of the command language?

After reading these instructions, the users were asked to work through the material contained in the model -- the tutorial on the model command language and the exercise using the data base of records from the Air Pollution Index. This required, in almost all cases, less than two hours. Since at the time the users were run there were still mechanical and system bugs, a member of the IIDA project staff served both as observer and as proctor to assist any user in overcoming system difficulties of a mechanical nature. The observers also kept track of the difficulties encountered by the user so that changes and refinements in the model could be made.

There would seem to be little point in making a detailed report on the number of mechanical flaws, misspellings, or confusing wordings revealed by these observations. Suffice it to say that they existed in sufficient number to cause some chagrin on the part of some of the observers who had also been involved in the educational programming of the model. However, overall, the consensus of the observers was that the system worked as expected and seemed to work well.

This feeling on the part of the observers is supported by the responses given us by the users on a questionnaire administered at the conclusion of the training session. The purpose of the questionnaire was to give us some written feedback about the users' affective responses to the system and some of their impressions about working with IIDA. The questionnaire consisted of two sections. The first asked the users for their feelings and comments about IIDA as a way of learning about interactive searching. The second section asked the users for their feelings and comments about IIDA as an assistant or consultant in actually doing on-line searching. It should be noted that in the second

part of the questionnaire, the context for the comments made by the users is one in which their only real exposure to IIDA as a consultant is through the exercise. At that point IIDA is in a transition between the roles of teacher and consultant.

When asked to give their positive impressions of IIDA as a teacher, all of our users indicated that there was something that they liked. In general the aspects of IIDA most preferred had to do with the flexibility offered by an interactive system -- flexibility in terms of guiding their own learning by being able to choose search topics, work at their own speed, or do reviews as desired.

When asked to give their negative impressions of IIDA as a teacher, all but one of our users indicated that they disliked something. In general, however, what they reported disliking were certain physical or mechanical aspects of the system. Two users did, however, feel that the system needed more flexibility in terms of being able to explain something to the user in more than one way if he did not get it the first time.

When asked how their perceptions of learning to do interactive information retrieval had changed since using IIDA, inexperienced users tended to report either no change or that they had not realized before what was involved in learning about interactive searching. Experienced users almost unanimously felt that exposure to IIDA had helped them develop more confidence and a better feel for what was actually going on.

When asked to give their positive impressions of IIDA as a consultant in actually doing a search, almost all users responded with comments indicating that they liked the flexibility, consistency, and scope of search capacity made possible by IIDA.

When asked to give their negative impressions of IIDA as a consultant in searching, the users, in general, reported disliking some physical or mechanical aspects of the system (e.g., format of comments made by IIDA). Other negative impressions either focused on user lack of knowledge about the data base or the data base thesaurus. The major negative comments about aspects of IIDA per se came from the more experienced users who felt that there was too much repetition of information.

When asked how their perceptions of actually doing an interactive search had changed since using IIDA, most inexperienced users either reported no change or felt that they now had a feel for the commands and how they could be used. The more experienced users either felt it might not be necessary to have IIDA available or felt that IIDA would be of great help as a consultant.

When asked if they would like to sue IIDA as a consultant in doing a literature search on a topic relevant to their own interest, all 12 users but 1 indicated that they would. The one person who said "no" felt that IIDA would be unnecessary because everything she needs is already easily available.

The questionnaire used with test subjects follows.

Since you are part of the first group to have any extended experience with IIDA, we would like you to record your reactions and comments to help guide us in the future development of this system. At this point we are particularly interested in any suggestions you may have for improvement of the system and in what you see as being the potential benefits of a system such as IIDA.

When fully developed IIDA will in effect be able to serve two roles. The first role is that of teacher of the skills required to do interactive information retrieval. The second role is that of an on-line consultant to assist the searcher if problems are encountered. This questionnaire is divided into two major sections. In the first section you will find a series of questions asking you for your impressions of IIDA as a teacher. In the second, you will find questions asking you for your impressions of IIDA as a consultant.

On the following page you will find several scales designed to assess your feelings and attitudes toward IIDA as a tool for learning how to do interactive information retrieval. There are 18 scales altogether. Even if you find some of them strange or inappropriate it is important that you complete them all.

Please consider:

IIDA as a way of learning interactive information retrieval

You have conducted a search of a mini data base using IIDA. Please don't focus on the product of the search itself, but consider the system you used to learn about interactive searching.

Work rapidly through the scales without pausing for more than just a few seconds on each one and without returning to one you have already completed. Please place a check at the point on each scale which you feel to be most

ropriate.

IIDA as a way of learning interactive information retrieval

boring	:___:___:___:___:___:___:___:	interesting
complex	:___:___:___:___:___:___:___:	simple
free	:___:___:___:___:___:___:___:	constrained
good	:___:___:___:___:___:___:___:	bad
important	:___:___:___:___:___:___:___:	unimportant
meaningless	:___:___:___:___:___:___:___:	meaningful
passive	:___:___:___:___:___:___:___:	active
pleasurable	:___:___:___:___:___:___:___:	painful
sensitive	:___:___:___:___:___:___:___:	insensitive
stabl	:___:___:___:___:___:___:___:	changeable
strong	:___:___:___:___:___:___:___:	weak
tenacious	:___:___:___:___:___:___:___:	yielding
ugly	:___:___:___:___:___:___:___:	beautiful
unsuccessful	:___:___:___:___:___:___:___:	successful
technical	:___:___:___:___:___:___:___:	non-technical
relaxed	:___:___:___:___:___:___:___:	tense
informal	:___:___:___:___:___:___:___:	formal
valuable	:___:___:___:___:___:___:___:	worthless

Next we would like to have you answer a series of questions about IIDA as a teacher or a system for learning how to do interactive information retrieval. We would appreciate it if you could help us by answering them as fully as you can.

1. How good are your typing skills? (Please describe.)

2. How familiar are you with computers and their uses? (Please describe.)

3. How much experience have you had in working with computers using remote terminals? (Please describe.)

4. How much experience have you had with computer based interactive information retrieval? (Please describe.)

5. Have you had any previous experience with other methods of teaching computer based interactive information retrieval? (If "yes" continue with 6, if "no" go to 8.)

6. Which methods? (Please describe.)

7. Given the way you first learned to search, what do you think about this method?

8. Please describe your impressions of learning to do computer based interactive information retrieval using IIDA as a teacher or system for learning.

Positive Impressions:

Negative Impressions:

9. How has your perception of learning to do computer based interactive information retrieval changed since using IIDA? (Please describe.)

Now that you have given us your reactions to IIDA as a teacher or a system for learning how to do computer based interactive information retrieval we would like to have you give us your impressions of IIDA as a consultant in actually doing interactive information retrieval.

On the following page you will find several scales designed to assess your feelings and attitudes toward IIDA as a consultant to assist in doing interactive information retrieval. There are 18 scales altogether. Even if you find some of them strange or inappropriate it is important that you complete them all.

Please consider:

IIDA as a consultant to assist in interactive information retrieval

You have conducted a search of a mini data base using IIDA. Please don't focus on the product of the search itself, but consider the system as a consultant.

Work rapidly through the scales without pausing for more than just a few seconds on each one and without returning to one you have already completed. Please place a check at the point on each scale which you feel to be most appropriate.

IIDA as a consultant to assist in interactive information retrieval

boring	: : : : : : :	interesting
complex	: : : : : : :	simple
free	: : : : : : :	constrained
good	: : : : : : :	bad
important	: : : : : : :	unimportant
meaningless	: : : : : : :	meaningful
passive	: : : : : : :	active
pleasurable	: : : : : : :	painful
sensitive	: : : : : : :	insensitive
stable	: : : : : : :	changeable
strong	: : : : : : :	weak
tenacious	: : : : : : :	yielding
ugly	: : : : : : :	beautiful
unsuccessful	: : : : : : :	successful
technical	: : : : : : :	non-technical
relaxed	: : : : : : :	tense
informal	: : : : : : :	formal
valuable	: : : : : : :	worthless

Next we would like to have you answer a series of questions about IIDA as a consultant in doing interactive information retrieval. We would appreciate it if you could help us by answering them as fully as you can.

1. Please describe your overall impressions of doing computer based interactive information retrieval with IIDA serving as an assistant or consultant.

   Positive Impressions:

   Negative Impressions:

2. How has your perception of doing computer based interactive information retrieval changed since using IIDA? (Please describe.)

3. If IIDA were available for you to use as a consultant in doing a literature search on a topic of relevance to you, do you think you would make use of it? Why?

THANK YOU FOR YOUR HELP

(If you have any further thoughts about IIDA, either as teacher or as consultant, please use the space below. We are particularly interested in any suggestions you may have for improvement of the system and in what you see as being the potential benefits of a system such as IIDA.)

APPENDIX A: TUTORIAL PROGRAMS

1. STRUCTURE OF COMPENDEX

1.1 Objectives of the Course

The purpose of this course is to teach students: (1) about the material the comprises the Compendex file, (2) the structure of the file in the sense of the information elements that make up a record, and (3) the use of bibliographic tools such as subject headings. No prior background in the use of Compendex or the Engineering Index is assumed.

1.2 General Characteristics of Compendex

This section will identify the publisher of Compendex, the materials covered by it, size of the file, date span of contents, and frequency of updating.

1.3 Organization of the File

This section will cover the specific fields of information present in a Compendex record and their interrelationships. It will describe the formats available for record printing or display and will show the student one or more sample recrods, annotated as to element structure.

1.4 Abstracting in Compendex

This section will describe how abstracting is done and by whom. It will cover the standard content of an abstract, what material is reviewed for abstracting, and who the abstractors are.

1.5 Subject Description

Here will be described the subject divisions and subject headings used in Compendex. Mention will be made, in anticipation of later course material, of the possibility of searching titles and abstracts for occurence and co-occurences of words.

1.6 Timing

This course should be completed in 20 - 30 minutes by the average student.

2. SEARCH MECHANICS

2.1 Objectives and Organization of the Course

This course introduces the student to the tools used in conducting a search. Some mention is made of the terminal, but concentration is on software tools -- the search or command language. The course will be taught in two segments. The first introduces a minimal subset of the command language, a small number of commands which can be easily learned but which can enable a student to begin doing useful searches almost immediately. In some cases, the many variations actually possible in the use of a command will not be taught, and only a limited range of uses described. In the second segment, the full command set will be presented.

The intent is that a typical student would take only the first segment initially, then go on to the exercises and possibly even an assisted search. After he begins to feel comfortable with the basic commands, he can return to the second segment of this course and the corresponding exercise. For some users, it will never be necessary to take the full language segment or exercise.

2.1 Access to the System

Ir the first section of the course, the mechanics of accessing DIALOG are described: use of the telephone, dialling into the network, the DIALOG password, the use of the terminal, mechanical differences between DIALOG AND IIDA.

2.2 The Minimal Subset Segment

Commands covered in this segment are: BEGIN, EXPAND, SELECT, COMBINE, PRINT and LOGOFF. They will be taught with the following restrictions:

1. BEGIN. Always used in the form BEGIN8 which calls for the Compendex file and reinitializes set numbers.
2. EXPAND. The argument of EXPAND may be a descriptor or a line number (En or Rn).
3. SELECT. The argument of a SELECT command in this segment is restricted to a descriptor or a single line number (En or Rn). Truncated descriptors, multiple line numbers or full text searches are not taught at this level.
4. COMBINE. The argument of COMBINE will be taught as an expression of set numbers with boolean operators. Parentheses may be used.
5. PRINT. Printing will be restricted to either of two formats: one showing title only and one the complete citation without abstract. The abstract is omitted in order to reduce the volume of printing.
6. LOGOFF. This command has no variations or argument.

This segment of the course should take a typical student about 15 minutes to complete.

2.3 The Full Language Segment

In this segment the full range of each command is explained to the student. He need not take the entire course in sequence, but will be permitted either to do that or to request instruction on a particular command only. Thus, this segment of the course can be used in a reference mode as well as a conventional tutorial mode.

If presented sequentially, the commands will be described in phase order, using the five phases listed in Section 1.2.2 (index search, logic formation, record display, procedural and diagnostic). The diagnostic portion will concentrate on diagnostic information available through DIALOG but will briefly introduce the extended set of diagnostics available through IIDA.

IIDA instruction will not cover the symbolic form of commands accepted by DIALOG, except for the ? representing EXPLAIN.

The second segment of the search mechanics course, if taken in its entirety, would probably consume an hour.

3. SEARCH STRATEGY

3.1 Objectives

By now the student has learned something about the composition of Compendex and the commands available to him to search it. This course will provide instruction on some general principles for conducting a search -- how to plan and approach the overall search. We have previously pointed out that there is no accepted theory of search strategy. Nonetheless, we hope to teach students from the beginning that they should have some plan when they begin a search.

3.2 Statement of Objective

We will ask students to state a search objective. This is intended to be a guide not a limitaiton on their performance. They might elect to search for a few high grade references, a bibliography of a dozen or so citations, or a fairly complete bibliography that may run to dozens, even hundreds of citations. The point is that search behavior will be different if the searcher is looking for one or two best references rather than a hundred or so somewhat relevant ones. The student may at any time change his objective, but he should be conscious of the fact that he is doing this.

3.3 Starting Point and Direction

The student should begin his search with a few descriptors in mind. He should also decide early in the search whether he will try to find one or two articles of known relevance and use bibliographic descriptors found with them or work from descriptors to documents; whether he will try to define a small set and then gradually enlarge it to suit his needs, or define a large set and gradually reduce it until it satisfies him.

3.4 Specific Techniques.

Here, we will discuss a few techniques a user may use to get a search started, such as

Locating a known article and using its descriptors to look for more.

Locating articles in a journal or journal issues of known relevance.

Retrieving articles by a well-known author in a field, or originating in a specific laboratory or institution.

Using date or other limits to reduce the size of exploratory sets until an adequate-appearing set description is reached, then expanding the date range.

There are others. There will be no attempt to be exhaustive, rather to implant in the student's mind the idea that there is or can be strategy to his searching.

3.5 Timing

This course should take no more than 15 minutes.

4. DIAGNOSTICS

4.1 Purpose of IIDA Diagnostic Instruction

Diagnostics are the heart of the IIDA teaching technique. Their purpose is to help the student identify the source of his own problems by providing analyses from various points of view of his commands, results obtained and errors made. The diagnostics cannot interpret what the student intended. Except for readily-detected procedural or syntactic errors, they try to give the student data from which he can determine that he is doing something 'wrong' (i.e., unproductive) and change his approach.

Initially, a student might be overwhelmed with diagnoses of a process which he does not yet understand well. Use of the diagnostics also requries skill, as does searching, and both forms of skill require instruction and practice. Hence, the student must be taught what diagnostics are available and how to use them.

4.2 Diagnostic Techniques

4.2.1 Parser. This is the first diagnostic program the student is likely to encounter and it is one over which he has no control. Whenever, in exercise or assistance mode, a student enters a command that is either procedurally restricted at that time or syntactically incorrect, the parser will converse with him. In this course, the student will be told the circumstances under which this happens, given some examples of parser messages, and instructed how to respond.

4.2.2 /HELP. The /HELP program may be voluntarily involved by a student or called by the performance analysis routine. /HELP operates on the concept of a tree of menus, through which the student selects a general, than ever more specific diagnostic or instructional assistance. He will be shown examples of the /HELP menus and the logic of their construction will be explained. He has options, which will be explained, for returning to the porgram from whence he came or to other program safter he has invoked /HELP. The use of /HELP as a tutorial program will be explained.

4.2.3 Performance analysis routine. The course will explain to the student the kinds of checks made by the PAR and the circumstances under which it invokes /HELP.

4.2.4 Proctor. The role of the proctor will be explained, as will the mechanics of how a student may contact him or be contacted by him.

4.2.5 Restart. In the event of machine failure during a search, the student will have the option of continuing his previously begun search or starting over, when the computer is eventually restored to service. The nature of this option and its implications will be explained to him both in this diagnostic course and more briefly, each time a restart occurs.

4.3 Timing.

This course should require about 15-20 minutes.

APPENDIX B: TRANSCRIPT OF A TEST USE

On the following pages is a transcript of the record of a search done by a student during the testing of the IIDA model. It is presented here to illustrate the kinds of messages a student receives and how he might react to them. It is not an "ideal" use -- the transcript was selected at random from the complete set of trial uses.

In a few cases the right hand portion of a line of text has been truncated to accommodate the wider margins of this report. No serious lessening of comprehensibility has resulted.

The material on the last two pages is a diagnostic summary of the search. It is printed out, not for student use, but in simulation of the kind of material a proctor would want who was reviewing a student's performance.

```
PEADY
RUNSAV GUIDED.SAV

STUDENT NAME: ?[redacted]
GOOD DAY.  YOU HAVE LOGGED INTO THE GUIDED EXERCISE.
A SMALL DATA BASE ON AIR POLLUTION IS AVAILABLE.
SOME GUIDANCE IN SEARCHING WILL BE AUTOMATIC.  IF AT
ANY TIME YOU REQUIRE FURTHER ASSISTANCE, TYPE IN
/HELP  FOLLOWED BY A CARRIAGE RETURN.
     REMEMBER TO MAKE AN ENTRY ONLY AFTER GETTING A QUESTIONMARK (?).
REMEMBER ALSO TO END EVERY ENTRY WITH A CARRIAGE RETURN <CR>.
PLEASE TYPE IN A SHORT SUMMARY OF YOUR SEARCH TOPIC.
 ?

MEASURING EMISSION SOURCES

USING PENCIL AND PAPER, JOT DOWN A FEW DESCRIPTORS
YOU FEEL MAY BE HELPFUL.  WHEN YOU ARE THROUGH WRITING,
HIT THE CARRIAGE RETURN.

 ?

              FILE AIRP IS ON-LINE

              SET   COUNT   DESCRIPTION
              ===   =====   ===========

SEARCHING A DATA BASE CAN BE DIVIDED INTO THREE PHASES.
     PHASE 1 IS THE TIME FOR BROWSING IN THE THESAURUS
WITH THE EXPAND AND RELATE COMMANDS.
     PHASE 1 IS ALSO THE TIME FOR USING THE SELECT
COMMAND TO FORM SETS OF ITEMS INDEXED BY THE DESCRIPTORS
OF YOUR CHOICE.
     PHASE 1 COMMANDS ARE EXPAND, RELATE, AND SELECT.
IT IS USEFUL TO KNOW ALL PHASE 1 COMMANDS.
YOU MUST TRY ALL PHASE 1 COMMANDS BEFORE
MOVING ON TO PHASE 2 BY ENTERING /DONE.

ENTER PHASE 1 COMMAND OR /DONE.
 ?SELECT EMISSION FACTORS

        1      6    EMISSION FACTORS
```

```
ENTER PHASE 1 COMMAND OR /DONE.
 ?SELECT POLLUTANTS

                2        8     POLLUTANTS

ENTER PHASE 1 COMMAND OR /DONE.
 ?SELECT AEROSOLS

                3       14     AEROSOLS

ENTER PHASE 1 COMMAND OR /DONE.
 ?EXPAND AEROSOLS
```

REF	DESCRIPTION	CNT	RET
12	19=AEROSOL ATOMIZATION	1	2
13	19=AEROSOL ELECTRICAL PHENOMENA	3	2
14	19=AEROSOL FORMATION	3	3
15	19=AEROSOL PHENOMENA	1	6
16	19=AEROSOL SPECTROMETERS	1	2
	-05=AEROSOLS		
17	19=AEROSOLS	14	4
18	09=AIR POLLUTION CONTROL ASSOC ANNU M	17	0
19	09=AIR POLLUTION CONTROL ASSOC SPEC S	1	0
20	19=AIR POLLUTION FORECASTING	1	5
21	19=AIR PREHEAT	1	1
22	16=AIR QUALITY MEAS	3	0

```
ENTER PHASE 1 COMMAND OR /DONE.
 ?RELATE AEROSOLS
```

REF	DESCRIPTION	REL	CNT	RET
17	19=AEROSOLS		14	4
322	19=SETTLING PARTICLES	0	1	5
348	19=SUSPENDED PARTICULATES	0	3	6
277	19=PARTICULATES	1	26	13
289	19=POLLUTANTS	1	8	17

```
ENTER PHASE 1 COMMAND OR /DONE.
 ?DONE

USER INPUT: DONE
ERROR # 14 : CANNOT RECOGNIZE D AS A VALID COMMAND CODE

ENTER PHASE 1 COMMAND OR /DONE.
 ?/DONE

YOU HAVE NOW ENTERED PHASE 2.
THIS IS A DISPLAY OF THE HISTORY OF THE SETS YOU HAVE
CREATED SO FAR.
```

SET	COUNT	COMMAND	DESCRIPTION	RELEVANCE
1	6	SELECT	EMISSION FACTORS	
2	8	SELECT	POLLUTANTS	
3	14	SELECT	AEROSOLS	

IN PHASE 2 THE COMBINE COMMAND IS AVAILABLE TO USE WITH THE SET NUMBERS FROM THE TABLE SHOWN ABOVE.
ENTER A COMBINE COMMAND.
?COMBINE 1 + 2

4 0 1 + 2

ENTER COMBINE COMMAND OR /DONE.
?COMBINE 2 + 3

5 22 2 + 3

ENTER COMBINE COMMAND OR /DONE.
?DONE

USER INPUT: DONE
ERROR # 1 : INCOPRECT COMMAND

ENTER COMBINE COMMAND OR /DONE.
?/DONE

YOU HAVE COMPLETED, AT LEAST TEMPORARILY, PHASES 1 AND 2. BEFORE PROCEEDING, REVIEW YOUR SEARCH SO FAR. THIS IS A DISPLAY OF THE SETS YOU HAVE CREATED SO FAR.

SET	COUNT	COMMAND	DESCRIPTION	RELEVANCE
1	6	SELECT	EMISSION FACTORS	
2	8	SELECT	POLLUTANTS	
3	14	SELECT	AEROSOLS	
4	0	COMBINE	1 + 2 =EMISSION FACTORS + POLLUTANTS	
5	22	COMBINE	2 + 3 =POLLUTANTS + AEROSOLS.	

BRANCHING TO OTHER PHASES IS NOW POSSIBLE.
ENTER ONE OF THESE OPTIONS BY NUMBER:
1. SATISFIED, WANT TO PRINT SOME ITEMS TO VIEW BY GOING ON TO PHASE 3.
2. NEED MORE DESCRIPTORS, WANT TO RETURN TO PHASE 1 AND SELECT, EXPAND, OR RELATE MORE TERMS.
3. NEED MORE LIMITED SETS, WANT TO RETURN TO PHASE 2 TO COMBINE MORE SETS.
4. DISSATISFIED, WANT TO START SAME SEARCH OVER BY RETURNING TO BEGINNING.
5. UNHAPPY, WANT TO START NEW SEARCH BY RETURNING TO THE BEGINNING.
6. UNSURE, NEED HELP.
?3

ENTER COMBINE COMMAND OR /DONE.
?COMBINE 1 + 3

6 20 1 + 3

ENTER COMBINE COMMAND OR /DONE.
?COMBINE 1 + 2

7 14 1 + 2

ENTER COMBINE COMMAND OR /DONE.
?/DONE

YOU HAVE COMPLETED, AT LEAST TEMPORARILY, PHASES 1 AND 2. BEFORE PROCEEDING, REVIEW YOUR SEARCH SO FAR. THIS IS A DISPLAY OF THE SETS YOU HAVE CREATED SO FAR.

SET	COUNT	COMMAND	DESCRIPTION	RELEVANCE
1	6	SELECT	EMISSION FACTORS	
2	8	SELECT	POLLUTANTS	
3	14	SELECT	AEROSOLS	
4	0	COMBINE	1 * 2 =EMISSION FACTORS * POLLUTANTS	
5	22	COMBINE	2 + 3 =POLLUTANTS + AEROSOLS	
6	20	COMBINE	1 + 3 =EMISSION FACTOR + AEROSOLS	
7	14	COMBINE	1 + 2 =EMISSION FACTORS + POLLUTANTS	

```
 * BRANCHING TO OTHER PHASES IS NOW POSSIBLE.
  ENTER ONE OF THESE OPTIONS BY NUMBER:
     1. SATISFIED, WANT TO PRINT SOME ITEMS TO VIEW BY
GOING ON TO PHASE 3.
     2. NEED MORE DESCRIPTORS, WANT TO RETURN TO PHASE 1
AND SELECT, EXPAND, OR RELATE MORE TERMS.
     3. NEED MORE LIMITED SETS, WANT TO RETURN TO PHASE 2
TO COMBINE MORE SETS.
     4. DISSATISFIED, WANT TO START SAME SEARCH OVER BY
RETURNING TO BEGINNING.
     5. UNHAPPY, WANT TO START NEW SEARCH BY RETURNING
TO THE BEGINNING.
     6. UNSURE, NEED HELP.
 ?1

     YOU HAVE NOW ENTERED PHASE 3.
     IN PHASE 3, THE PRINT COMMAND IS AVAILABLE.
FROM THE SET HISTORY JUST PRINTED, CHOOSE A SET TO VIEW.
ENTER A PRINT COMMAND FOR THAT SET.
 ?PRINT 7/3/1-3


           ACCESS NO. = 58
05=NONE
06=JACKO R B
06=NEUENDORFT D W
06=BLANDFORD J R
09=AIR POLLUTION CONTROL ASSOC ANNU MEET 69TH PORTLAND OREG 1976
10=1976
11*PREPRINT
16=EMISSION SOURCES
18=LAB
19=PARTICULATES
19=YEAR 1976
19=COKE OVENS
19=EMISSION FACTORS
35*84765
35*JACKO ROBERT B., DAVID W. NEUENDORF, AND JOHN R. BLANDFORD
36*THE BY-PRODUCT COKE OVEN PUSHING OPERATION: TOTAL AND TRACE METAL
36*PARTICULATE EMISSIONS.# AIR POLLUTION CONTROL ASSOC., PITTSBURGH, PA
36*15P., 1976.  (PRESENTED AT THE AIR POLLUTION CONTROL ASSOCIATION ANN

JUDGE THE RELEVANCE OF THIS RECORD BY ENTERING
AN INTEGER BETWEEN 1 AND 5 WHERE:
   1 = IRRELEVANT , 5 = MOST RELEVANT
 ?1
```

ACCESS NO. = 40
05=CONTRACT
06=DANA M T
06=LEE R N
06=SCHULZ E J
10=1975
16=EMISSION SOURCES
18=FLD
19=PARTICULATES
19=YEAR 1975
19=PETROLEUM REFINING
19=EMISSION FACTORS
19=PARTICLE SIZE DISTRIBUTION
19=SULFUR DIOXIDE
35◆84742
35◆Dana, M. Terry, R. N. Lee, and E. J. Schulz
36◆PARTICULATE AND SO2 EMISSIONS FROM PROCESS HEATERS AT SHELL AND TEXA
36◆REFINERIES, ANACORTES, WASHINGTON.# Battelle, Pacific Northwest Labo

JUDGE THE RELEVANCE OF THIS RECORD BY ENTERING
AN INTEGER BETWEEN 1 AND 5 WHERE:
1 = IRRELEVANT , 5 = MOST RELEVANT
?3

ACCESS NO. = 32
05=NONE
06=WARD D E
06=MCMAHON C K
06=JOHANSEN R W
09=AIR POLLUTION CONTROL ASSOC ANNU MEET 69TH PORTLAND OREG 1976
10=1976
11◆PREPRINT
16=EMISSION SOURCES
18=THEO
18=BIB
19=PARTICULATES
19=FORESTS
19=YEAR 1976
19=BURNING
19=EMISSION FACTORS
35◆84717
35◆Ward, Darold E., Charles K. McMahon, and Ragnar W. Johansen
36◆AN UPDATE ON PARTICULATE EMISSIONS FROM FOREST FIRES.# PREPRINT, THE
36◆Air Pollution Control Assoc., Pittsburgh, Pa., 15p., 1976. 20 refs.
36◆(Presented at the Air Pollution Control Association Annual Meeting, 1

JUDGE THE RELEVANCE OF THIS RECORD BY ENTERING
AN INTEGER BETWEEN 1 AND 5 WHERE:
1 = IRRELEVANT , 5 = MOST RELEVANT
?5

```
DO YOU WISH TO VIEW MORE SETS?
IF SO, ENTER ANOTHER PRINT COMMAND.
IF NOT, ENTER /DONE.
 ?PRINT 6/3/1


            ACCESS NO. = 58
05=NONE
06=JACKO R B
06=NEUENDORFT D W
06=BLANDFORD J R
08=AIR POLLUTION CONTROL ASSOC ANNU MEET 69TH PORTLAND OREG 1976
10=1976
11*PREPRINT
16=EMISSION SOURCES
18=LAB
19=PARTICULATES
19=YEAR 1976
19=COKE OVENS
19=EMISSION FACTORS
35*84765
35*JACKO ROBERT B., DAVID W. NEUENDORF, ANI JOHN R. BLANDFORD
36*THE BY-PRODUCT COKE OVEN PUSHING OPERATION: TOTAL AND TRACE METAL
36*PARTICULATE EMISSIONS.# AIR POLLUTION CONTROL ASSOC., PITTSBURGH, P
36*15P., 1976. (PRESENTED AT THE AIR POLLUTION CONTROL ASSOCIATION ANN

JUDGE THE RELEVANCE OF THIS RECORD BY ENTERING
AN INTEGER BETWEEN 1 AND 5 WHERE:
   1 = IRRELEVANT , 5 = MOST RELEVANT
 ?5

DO YOU WISH TO VIEW MORE SETS?
IF SO, ENTER ANOTHER PRINT COMMAND.
IF NOT, ENTER /DONE.
 ?/DONE

THE FOLLOWING CHOICES ARE NOW AVAILABLE.
1. CHOOSE MORE DESCRIPTORS, EITHER FROM THE MATERIAL JUST VIEWED, OR NI
2. COMBINE DESCRIPTORS ALREADY CHOSEN IN NEW WAYS.
3. PRINT MORE RECORDS IF YOU HAVE NOT SEEN THEM ALL.
4. DECIDE YOU ARE SATISFIED AND YOUR SEARCH IS COMPLETE.
5. REQUEST HELP.
MAKE YOUR CHOICE AND ENTER ITS NUMBER.
 ?4
```

```
NOW THAT YOU CONSIDER YOUR SEARCH COMPLETE,
WOULD YOU CHOOSE ONE OF THE FOLLOWING
THAT MOST NEARLY FITS YOUR EVALUATION OF
THE RECORDS RETRIEVED FOR YOU.
OF THE RECORDS I SAW, I CONSIDER
1. LESS THAN 10% TO HAVE BEEN RELEVANT.
2. 10-25% WERE RELEVANT.
3. 25-50% WERE RELEVANT.
4. 50-75% WERE RELEVANT.
5. 75-90% WERE RELEVANT.
6. 90-100% WERE RELEVANT.
 ?3

SINCE YOU FEEL YOU DIDN'T FIND MANY RELEVANT ITEMS
DO YOU SUPPOSE THERE IS NOTHING IN THE LITERATURE
OR DO YOU THINK A REVIEW OF THE TERMS IN THE
AUTHORITY LIST WOULD BE HELPFUL?
THESE QUESTIONS ARE FOR YOUR USE IN THE FUTURE.

THANK YOU, I HOPE YOU ENJOYED YOUR SEARCH.
```

REVIEW OF COMMANDS AND SEARCH STAGES

STAGE	LINE	COMMAND
	1	SELECT EMISSION FACTORS
	2	SELECT POLLUTANTS
	3	SELECT AEROSOLS
	4	EXPAND AEROSOLS
	5	RELATE AEROSOLS
	6	COMBINE 1 * 2
	7	COMBINE 2 + 3
	8	COMBINE 1 + 3
	9	COMBINE 1 + 2
	10	PRINT 7/3/1-3
	11	PRINT 6/3/1

REVIEW OF SETS CREATED

SET	COUNT	COMMAND	DESCRIPTION	RELEVANCE
1	6	SELECT	EMISSION FACTORS	
2	8	SELECT	POLLUTANTS	
3	14	SELECT	AEROSOLS	
4	0	COMBINE	1 + 2 =EMISSION FACTORS + POLLUTANTS	
5	22	COMBINE	2 + 3 =POLLUTANTS + AEROSOLS	
6	20	COMBINE	1 + 3 =EMISSION FACTORS + AEROSOLS	5
7	14	COMBINE	1 + 2 =EMISSION FACTORS + POLLUTANTS	3

REVIEW OF DESCRIPTORS USED

DESCRIPTOR USED	USED IN SELECT	SEEN IN EXPAND	USED IN RELATE
EMISSION FACTORS	X		
POLLUTANTS	X		
AEROSOLS	X	X	X

REVIEW OF ERRORS MADE

FOLLOWS LINE #	TEXT OF ERROR	COMMAND ATTEMPTED	ERROR CODE (TO BE TEXT)
5	DONE		
7	DONE		
		PRINT	14
		COMBINE	1

HIT CARRIAGE RETURN, PLEASE ?

TIME: 11.31 SECS.