ED 052 500	24	CG 006 510

AUTHOR	Roman, Richard Allan
TITLE	Developing and Implementing Materials for Computer Assisted Instruction. Information System for Vocational Decisions.
INSTITUTION	Harvard Univ., Cambridge, Mass. Graduate School of Education.
SPONS AGENCY	Office of Education (DHEW), Washington, D.C.
REPORT NO	PR-26
BUREAU NO	BR-6-1819
PUB DATE	Dec 69
GRANT	OEG-1-6-061819-2240
NOTE	69p.
EDRS PRICE	EDRS Price MF-$0.65 HC-$3.29
DESCRIPTORS	*Computer Assisted Instruction, Computer Programs, *Educational Technology, Guidance, Information Networks, *Information Systems, *Man Machine Systems, Occupational Guidance, Programing Languages, Vocational Counseling, *Vocational Development
IDENTIFIERS	Information System for Vocational Decisions

ABSTRACT

This final report discusses certain parts of the successes and failures, strengths and weaknesses of the development of the Information System for Vocational Decisions (ISVD) in terms of their relevance to the issues within the area of computer-assisted instruction. A major focus is on the kinds of computer-assisted instruction that promote interactive learning and access to data. In this regard, the scripts which assist inquirers to become monitors of their own decision processes are presented in some detail. A second major focus emphasizes the interaction between subject matter specialists and computer programmers which facilitates the creation, evolution and effectiveness of a computer language for information systems such as ISVD. The third major aspect of this report is a consideration of the setting and constraints surrounding the development of a computer-based system for guidance. (Author/TL)

INFORMATION SYSTEM FOR VOCATIONAL DECISIONS

Project Report No. 26

DEVELOPING AND IMPLEMENTING MATERIALS FOR COMPUTER ASSISTED INSTRUCTION

Richard Allan Roman

This paper was supported in part by Grant No. OEG-1-6-061819-2240 of the United States Office of Education under terms of the Vocational Education Act of 1963.

Graduate School of Education
Harvard University

December 1969

ABSTRACT

In June 1966 the United States Office of Education started a grant to Harvard University to conduct a project (contract number 1-061819-2240) called the Information System for Vocational Decisions Project (ISVD). The responsibility of Harvard in this project was to create a working model of a computer-based information system that can become part of the vocational and educational efforts of school systems. In the development of the first model, the Newton, Massachusetts school system has been the main focus. This working model is to be delivered on or before the first of July 1969, which is three years and one month from the day the ISVD project began.

Central to ISVD is a guidance theory which asserts that decision-making is more than an important component of career development. Indeed, to the principal guidance theorists of ISVD, vocational decision-making is the process through which career becomes definable at all. One premise related to this theory is that people grow in their ability to make decisions only to the extent they are aware of the process of their own decision-making. Thus one must be both a decider and an observer, they call it a monitor, of himself as a decider.

A second notion important to ISVD is that the individual must develop a "sense of agency," that is a sense that he is the principal agent in his own development. Part of this sense comes from his self-monitoring and part from the decision process itself in which the individual turns data into information.

Since data in the world are never complete, people must actively assimilate data rather than passively accept them. In this way, people create information from data and take responsibility for their decisions. Thus awareness of self, self-determination, and direct access to facts about the world are the three central concepts on which ISVD is based.

These same notions are relevant to instruction as well. In fact it has been my habit during the past three years of work on the development of the ISVD guidance project to think of my activities in a wider educational context than guidance. Specifically, I have considered many problems common to ISVD and its instructional counterpart, computer assisted instruction (CAI).

The purpose of this final report, therefore, is to consider and discuss certain important parts of the successes and failures, strengths and weaknesses of the development of ISVD in terms of their relevance to issues within the area of CAI. In this spirit the report has a major focus on the kinds of CAI materials that provide interactive learning and access to data. The scripts that assist inquirers to become monitors of their own decisions processes, we call them Monitoring scripts, are presented in some detail.

A second major focus is the creation and evolution of a computer language designed within ISVD to promote effective CAI. The discussion emphasizes the interaction between subject matter specialists and computer programmers necessary for the language to grow. Several techniques for promoting the necessary communication are discussed and then the current version of the language is evaluated on ten criteria with suggestions for improvements to both the language itself and its utilization.

A third major aspect of the report is a consideration of the setting and the constraints surrounding the development of a computer-based system for guidance. Particular attention is paid in this account to my role in the project as intermediary between guidance and computers. The unusual difficulties associated with this role stem in part from the nature of the project itself and in part from the different languages guidance people and computer people use to express themselves. The necessity for such an intermediary and the difficulties of the role should be of interest to anyone engaged in the effort to meaningfully apply technology to education.

INTRODUCTION

The guidance theory of the Information System for Vocational Decisions (ISVD) states that people grow in their ability to make decisions when they are aware of the process of their decision-making. Operationalizing this concept, which we call Monitoring, was the main reason for creating a computer system with natural language capabilities. Chapters One and Two present examples of Monitoring and protocols from the type of interactive computer assisted instruction toward which we worked.

The first two chapters are necessary background to Chapter Three, THE EVOLUTION OF A LANGUAGE, since the development of Monitoring and the production of some interactive computer instruction (CAI) materials shaped the capabilities of GLURP, the CAI language of the ISVD. While the ISVD is concerned with guidance for vocational decisions, the computer staff developed a general purpose CAI language. We believe that our language allows authors to write responsive materials more easily than other CAI languages.

GLURP grew through interaction of the computer staff and the needs of the guidance staff. Educational considerations suggested functions of the language; the language changed to provide those functions.

The final chapter discusses my role in the ISVD as an interface between guidance specialists and computer programmers. In it I try to show that a person who understands both instructional concerns and computer technology was necessary for the ISVD to develop and would be important in other projects that join subject matter specialists with computer programmers.

CHAPTER ONE

MONITORING AND DATA FILES

Confronting an inquirer with his own performance requires a system with objective data about the world, subjective impressions about the inquirer, and a network of materials to connect the two. The ISVD currently distinguishes these three logically different types of data files: primary data, inquirers' secondary data, and scripts.

The primary data files contain the system's knowledge of the external world. Currently the ISVD has files about occupations, military careers, colleges, trade schools, and school records of the inquirers.

The script network, a second kind of data file, presents material and collects responses from the inquirers. Only through the scripts does the inquirer communicate with the system. Through these scripts the system gathers the data of the "secondary data files," which are the "history vectors" of the inquirers, used by the system to make instructional decisions dynamically. This use of the inquirer's history vector is commonplace in computer assisted instruction (CAI) literature, though less common in practice (Rigney, 1962). The ISVD is not unique in recognizing the importance of history vectors as the fundamental difference between textbook writing, programmed instruction and the more powerful CAI.

The secondary files in the ISVD are not used merely to report results to the inquirer: "You tried 9 problems and made 3 errors for a 66.66% average. See you next time." They are not simply bookkeeping devices. "Inquirer 1143 has used the system three times. He has taken 21 scripts." The system does do both of these things, but they are by themselves theoretically unimportant.

The system also uses the history vectors to individualize instruction. The system calls the inquirer by name, recognizes that he has used certain scripts before and skips introductory material. The system reminds the inquirers of their previous activities and choices. All of these are traditional features of a CAI system.

The ISVD goes beyond these traditional uses of secondary files to confront the inquirer with his own performance. The inquirer is forced to consider and evaluate his performance, to be conscious of how he learns. This confrontation is Monitoring.

The guidance theory of the ISVD maintains that people grow in their capacity to make decisions as they become aware of the decisions they make and their decision-making process. Monitoring arises naturally from the theory of the ISVD; however, the idea of Monitoring belongs to all CAI. Any attempt to make a machine teach could be enriched by confronting the inquirer with his performance.

Monitoring at its simplest is recalling the inquirer's behavior in a similar situation in the past and reporting it to him. Such straightforward information retrieval at the appropriate time can be quite effective; the system uses this technique in the preference scripts when the inquirer's focus is on stating his criteria for jobs or colleges. Rather than distract attention from the task at hand by introducing lengthy discussion, the system simply makes the inquirer aware that his criteria and self evaluation may be changing.

More complex forms of Monitoring require extensive planning and detailed communication among authors. While it seems obvious in retrospect, it took two years for the ISVD to realize that Monitoring is not something the machine does; it is a consequence of exhausting work by the system planners. In order to say, "Last time you said A1, B1, and C2" to an inquirer, it is necessary to anticipate that you *will* want to say it and make provision to save the response.

Since planners did not anticipate this need, much of the secondary data the system collects is unavailable. Data that might be available is not because those who stored data did not communicate that fact to other authors. Names of stored data items were not recorded systematically. On the other hand much data that is desirable is not available because the script network does not contain appropriate store commands.

Anyone intending to use Monitoring in a large scale CAI project must provide authors with a way to communicate what they are saving, how it was elicited, what it is called, and the form of the datum.

They must also provide a way for authors to receive this information from others. Equally important, an author who wants to include Monitoring in his script, but finds that the appropriate data are not available must have a way to ask other authors to store the information.

Monitoring does not take place in terms of concepts like self-esteem, sense of agency or self process. While these are useful organizing concepts for observing and summarizing behavior, the computer does not understand them. The machine operates on one response at a time: humans form judgments about self-esteem only after integrating many responses; the machine can hardly be expected to do better.

Self-esteem, or any other Monitored psychological concept must be defined in terms of a set of responses to particular situations. If you believe certain words indicate self-esteem, you must list all of the words and detect responses that use those words. Even that is not enough. You must write script material that provides the stimuli for those words. Then if the inquirer obliges with one of the self-esteem words, the script can store away that fact for later use. Monitoring decisions are not made in the overview, but in the tedious particular.

The computer does not process English easily; there are too many ways to say the same thing. If the machine is to treat responses in categories, the English must be concisely encoded. Treatment by category is essential to any Monitoring more complex than simply reporting the inquirer's words in new context. Collection of data in encoded form is, therefore, essential to Monitoring.

Once the ISVD realized the necessity for concise encoding, we had to devise a way to achieve it. The obvious device of multiple choice questions makes encoding simple. The ISVD uses it as a convenient expedient; however, suggesting multiple choice options restricts the set of responses, forces conformity of thought, and imposes the author's language for certain concepts onto the inquirer. In an area as laden with values as career guidance, these faults particularly threaten the inquirer, so the ISVD has tried to process the inquirer's own words to achieve the desired encoding.

To get an idea of what is involved in this more complicated Monitoring let us consider the encoding of factors relevant to choosing a college. In order to recognize and encode the facts inquirers mention, the ISVD developed a set of standard factors an inquirer might mention. The list follows:

1. Location
2. Type of college (public, private, or religious)
3. Sex of student body
4. Size of student body
5. Types of programs (liberal arts, professional)
6. Financial aid
7. Urban or rural setting
8. Special courses
9. Activities
10. Admission requirements
11. Cost
12. Housing

Having named the twelve factors and established an encoding system, we prepared a dictionary of phrases equivalent to each factor. Then we created the ANALYZE script. ANALYZE examines an inquirer's statement and recognizes any word or phrase on the synonym list as the equivalent of the factor name. When it recognizes a factor, it makes the appropriate binary switch 1 and creates a form containing the word or phrase the inquirer used as a synonym. Thus ANALYZE accepts the inquirer's response in his own terms but stores away for retrieval a binary switch corresponding to the factor. ANALYZE is a complicated script providing a service to other scripts. In effect any script can ask for many logical steps by simply writing #(XQ:ANALYZE). Let us look at some of the possible applications.

The first of the Monitoring routines, EXPLOREONE, uses ANALYZE to determine what factors the inquirer considers important when thinking about college (or jobs). It explicitly reports the results of ANALYZE to the inquirer and confirms the accuracy of the analysis. For more details, see the flow chart of EXPLOREONE in Appendix A.

An interaction, typical of that generated by EXPLOREONE for the college data base follows:

MACHINE:	What kinds of things are important to you in thinking about colleges?
INQUIRER:	I want to know how much it will cost, how far from home the school is and if I can get in with my Board scores. Also about the girls there.

MACHINE: (Uses ANALYZE, then says)
We recognized words referring to location, admission requirements, and cost in your response. It's almost certain that this list doesn't include all of what's important to you in choosing a college. The list may include some meanings that aren't in your statement as you see it.

INQUIRER: Yes it does.

MACHINE: You may have some feelings and ideas about colleges that aren't easy to find words for, and the words you use to describe your decisions could be very different from the words we know. No one but you can ever really know what's important to you about colleges or any decision you make. Looking at the words you use to describe what's important to you is one way to get to know more about yourself and your deciding.

INQUIRER: (pushes send block)

MACHINE: You may have some questions about why we recognized these meanings in what you said. Location, admission requirements, and cost. If you want to know how we got any of these meanings out of what you said, type in the meaning you'd like explained, or type go if you don't want any more explanation.

INQUIRER: Admission requirements

MACHINE: It seemed you were talking about admission requirements when you said "get in". Is "get in" a word that means admission requirements for you?

INQUIRER: Of course.

MACHINE: Are there any others you want explained? Location, admission requirements, and cost?

INQUIRER: No, that is plenty.

Another script which does complex processing for use in Monitoring is GOOD OR BAD ITEM. Immediately after data is accessed for any item in the college (or occupational) data base the inquirer is asked whether he thinks the school (or job) is good or bad for him. He then is asked to give his reasons which are ANALYZE'd for keywords. The abstracted keywords are recorded as a set of binary switches. After classifying three schools as good or bad, the inquirer's statements are recalled for him, and he is asked to make a review statement of his criteria for goodness or badness. This statement too is ANALYZE'd for keywords. It would be interesting simply to present this result to the inquirer as EXPLOREONE does. The system goes further here, comparing the keyword switches in the three items with those for the later review statement. Any keywords appearing on all three specific statements are expected in the review statement. Consistency across the lists is noted and commented upon; the inquirer is asked to explain differences, using the COLLEGE CHANGE FACTOR script.

The COLLEGE CHANGE FACTOR script points out differences in several statements, forcing the inquirer to focus on reasons for his progress. Any two sets of binary switches can be compared by the COLLEGE CHANGE FACTOR script. This routine detects differences between one set of switches and another, and asks the inquirer to explain each change.

After the inquirer gives his reasons in free form, he is asked to compare his answer to some given previously by the ISVD staff and other inquirers. If he thinks his is included among the alternatives, no further interaction takes place. However, if he considers his answer unique, it is added to the list of answers that the next inquirer will see. This is one way the system grows with experience.

COLLEGE CHANGE FACTOR is also a service script. Any author who has used ANALYZE and wants to compare the resulting set of switches to any other set can provide COLLEGE CHANGE FACTOR with the two sets of switches and the names of each set and then instruct the COLLEGE CHANGE FACTOR script to compare the sets for equality.

Monitoring depends on systematic encoding of information. Once an encoding is achieved, Monitoring routines build upon each other. For example, GOOD OR BAD ITEM depends on ANALYZE, and COLLEGE CHANGE FACTOR depends on GOOD OR BAD ITEM. Just as these routines build on each other, more and more sophisticated uses of Monitoring can be built from simple observations.

To enable the system to gather data on decision-making behavior, rather than simply on verbal reports of ongoing process, the inquirer must be engaged in dialogue with constructed responses. Full sentences contain more information than single words and therefore are more desirable for Monitoring.

In order to reinforce the typing of full sentences, the machine must maintain a simulation of normal conversation. It must appear to understand the inquirer's language. While most of the ISVD is multiple choice oriented, the orientation routines do maintain the illusion of conversation. The second chapter illustrates some of these routines.

CHAPTER TWO

CONVERSATIONAL SCRIPTS

Orientation scripts route the inquirer to appropriate responses from the script network. Their central position in the ISVD is illustrated in Chart 1, page 8. The simplest model for an orientation would be a set of multiple choice questions. The particular set of questions an inquirer sees depends on his previous responses. Eventually the orientation determines which script materials are most appropriate for the inquirer. While such a set of multiple choice questions would be easy to write and would rapidly locate the appropriate scripts of the ISVD, we have chosen to process the inquirer's natural language questions to give the illusion that the inquirer is controlling the machine in his own language. This decision is based in part on the theory of the ISVD which says to begin with the inquirer's natural use of language leading him toward the machine's language. Another reason to avoid multiple choice questions in the central routines of the system is that they encourage the student to give short answers which directly conflict with the discursive behavior the ISVD tries to elicit in the Monitoring routines.

Having decided that the ISVD must try to process natural language questions, we must ask "How is it possible to recognize what an inquirer wants to do?" The literature of natural language processing assures us that recognizing arbitrary sentences is impossible. To make analysis possible, a number of assumptions are necessary. For the system to work, the inquirers must implicitly share these assumptions. If they do not, the field supervisors must impose them as restrictions.

A. The inquirer will want to play the game -- he will ask about something ISVD is programmed to understand.

B. The inquirer will try to make the machine understand rather than lapsing into perjurative comments.

C. The inquirer will use words in the usually understood sense.

D. The inquirer will use normal English syntax, generally simple sentences.

E. In most cases, the inquirer will not change the subject abruptly.

F. When the inquirer does not understand, he will type @HELP, a system command for further explanation.

Given these assumptions, it is possible to design a series of syntax rules that narrows the range of possible questions. The MJOR ORIENTATION determines the data base with which the inquirer is concerned by examining his input text for key words and phrases. For example, it would recognize an inquirer's interest in education if he used any derivatives of the following root words: college, school, university, educate, scholar, test, professor, or teach. [illegible] the inquirer has typed in a statement containing one or more of these [illegible], he is branched to the EDUCATION MINOR ORIENTATION. Similar word lists recognize each of the data bases.

There is always the possibility that the inquirer's input will not contain any of the words anticipated on the lists. Chart 2 contains a generalized orientation flow chart, which applies to the major, minor, and overview orientations. The third page of the chart (page 12) describes the procedures used when an inquirer has not typed in any recognized word. The procedures are based on a model of the way conversations proceed when the vocabularies of the two participants are inconsistent. The first time no match is found, the computer asks the inquirer to rephrase the question. The second time, the machine "hazards a guess" that the inquirer is referring to some arbitrary area of the data base. The second response is designed in the hope of stimulating a clearer statement when the inquirer disagrees. After a third misunderstanding, the computer will explain what kind of questions it can answer, hoping that the inquirer will then mimic some of the keywords. The last resort is to present a multiple choice list of possible activities.

Each orientation follows this flow in general, but varies in its specifics. The following protocol, from MAJOR ORIENTATION, illustrates this flow:

MACHINE: What would you like to do?

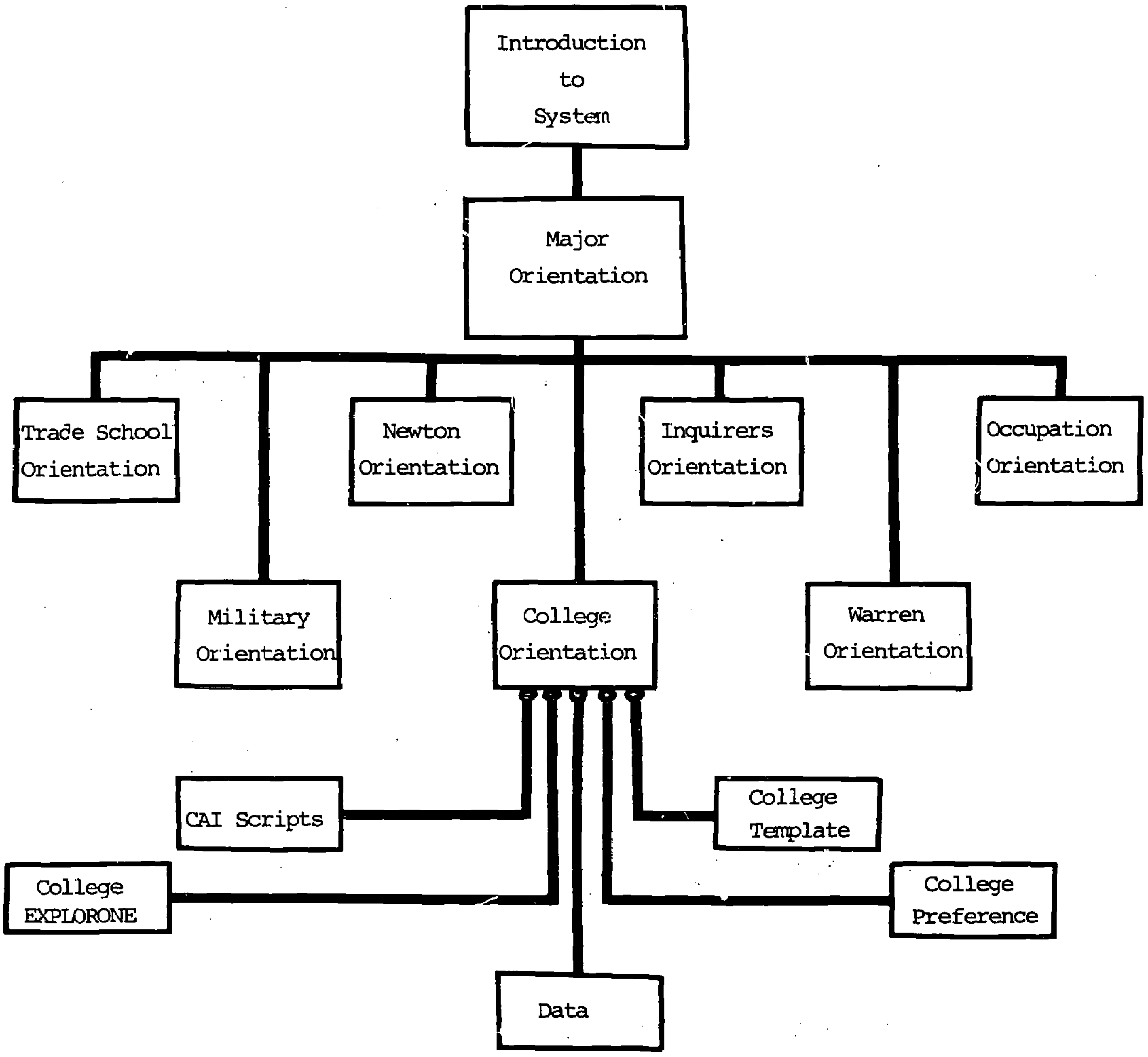
INQUIRER: Tell me about tuition at Harvard.

MACHINE: We have not understood your request. It would help us if you use general, simple words to tell us what you want to do.

INQUIRER: How much must I pay to go to Harvard.

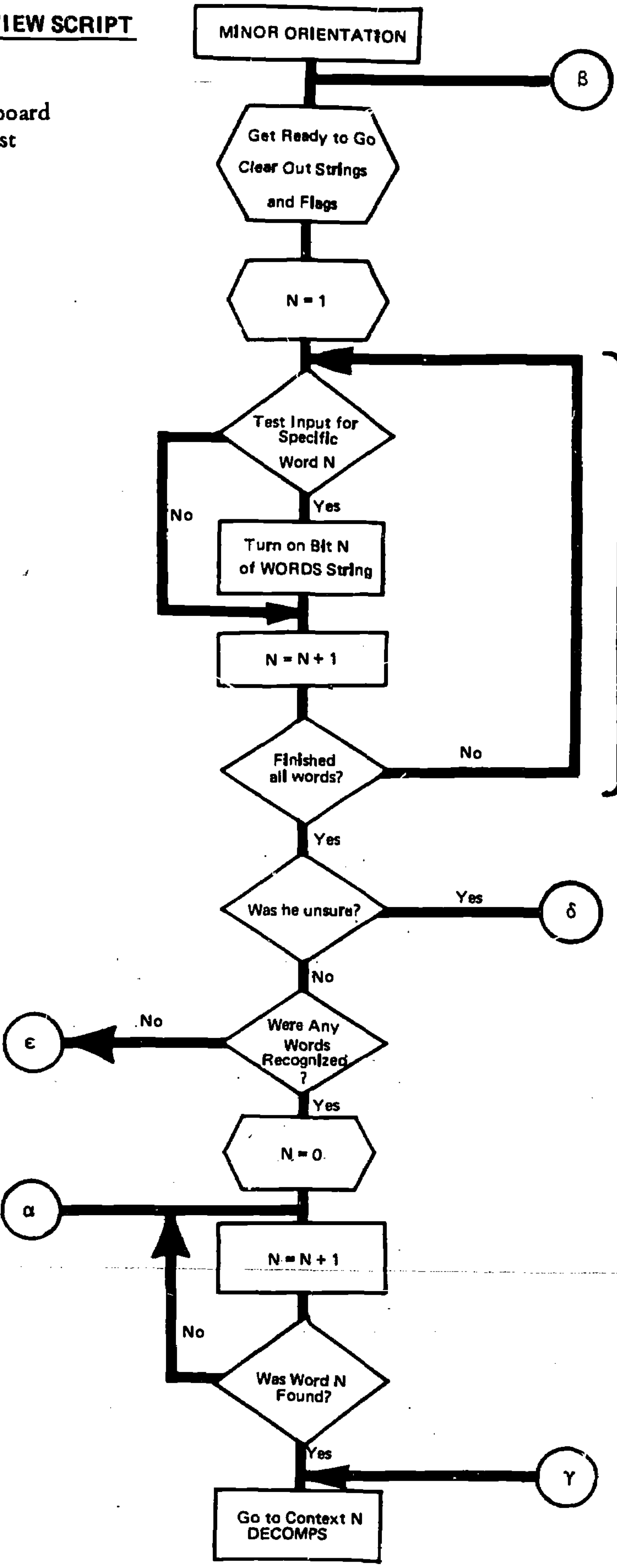
PROTOTYPE 1

Script Network of ISVD as Planned December 1, 1968



A TYPICAL OVERVIEW SCRIPT

Assume at ⓑ that the keyboard buffer has the inquirer's last statement.



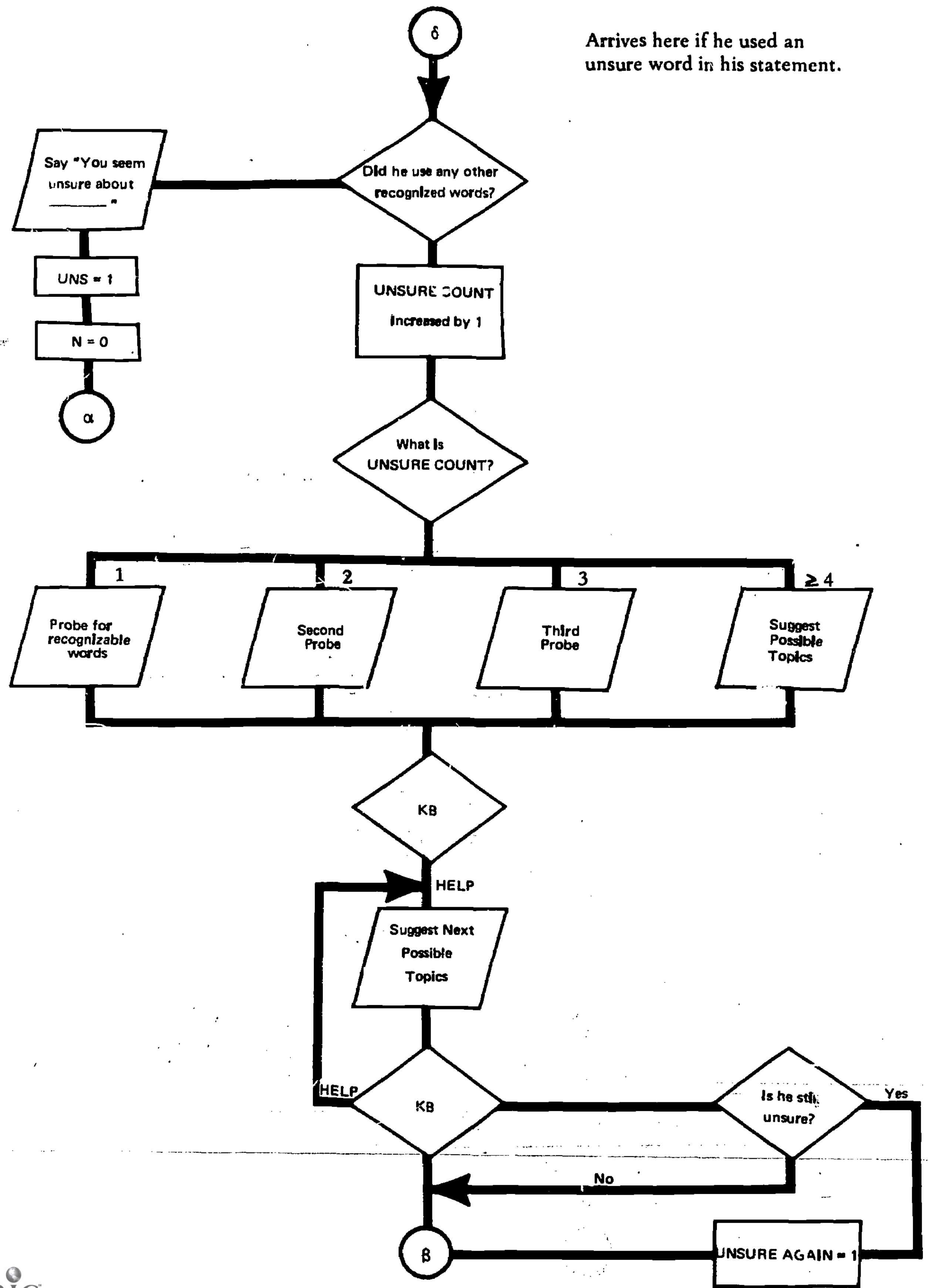
This section of code will test the inquirer's statement for each anticipated word in succession and remember which are there.

Let M be the total number of anticipated words.

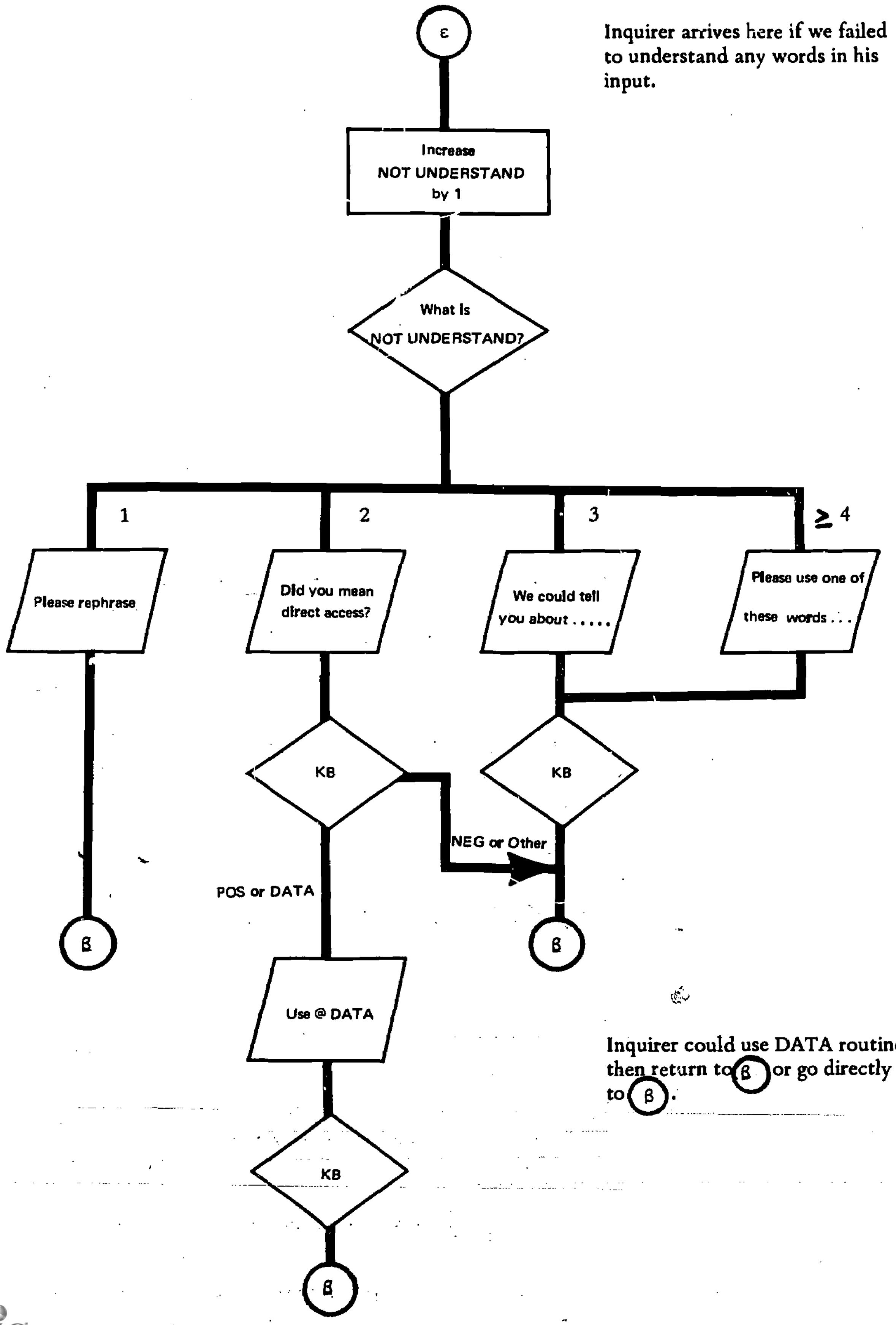
δ will handle inquirers who have expressed uncertainty.

ε will handle inquirers who weren't understood.

There are M subsections with similar structure.


δ
Did he use any other recognized words?
Say "You seem unsure about ______"
UNS = 1
N = 0
α
UNSURE COUNT increased by 1
What is UNSURE COUNT?
1
2
3
≥ 4
Probe for recognizable words
Second Probe
Third Probe
Suggest Possible Topics
KB
HELP
Suggest Next Possible Topics
HELP
KB
Is he still unsure?
Yes
No
UNSURE AGAIN = 1
β

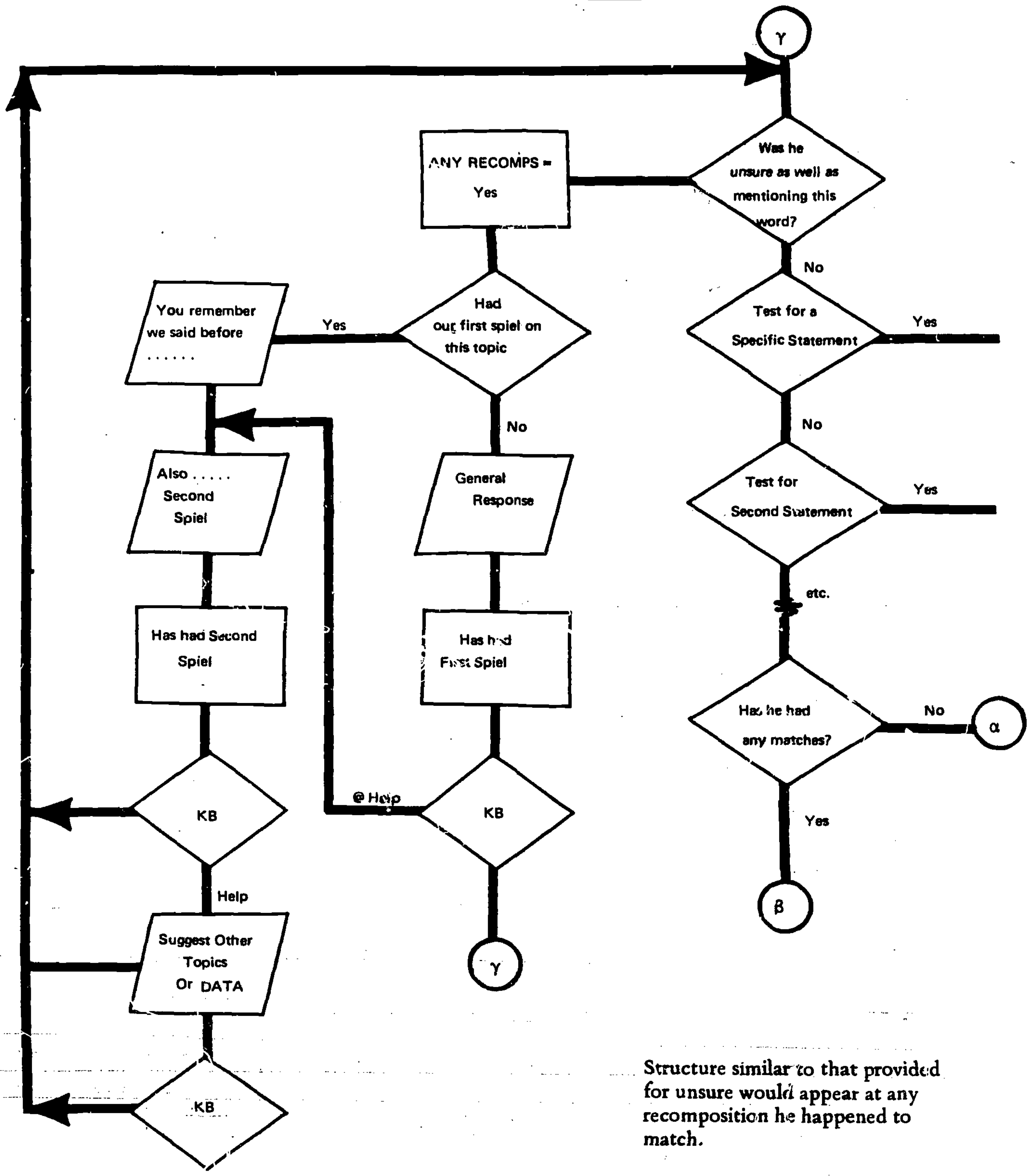
Arrives here if he used an unsure word in his statement.


ε
Increase NOT UNDERSTAND by 1
What is NOT UNDERSTAND?
1
2
3
≥ 4
Please rephrase
Did you mean direct access?
We could tell you about .....
Please use one of these words . . .
KB
KB
NEG or Other
POS or DATA
Use @ DATA
KB
β
β
β

Inquirer arrives here if we failed to understand any words in his input.

Inquirer could use DATA routines then return to (β) or go directly to (β).

A TYPICAL RECOMPOSITION SECTION


γ
ANY RECOMPS = Yes
Was he unsure as well as mentioning this word?
No
Had our first spiel on this topic
Yes
You remember we said before ......
No
Test for a Specific Statement
Yes
No
Also ..... Second Spiel
General Response
Test for Second Statement
Yes
etc.
Has had Second Spiel
Has had First Spiel
Has he had any matches?
No
α
KB
@ Help
KB
Yes
Help
β
Suggest Other Topics Or DATA
γ
KB

Structure similar to that provided for unsure would appear at any recomposition he happened to match.

MACHINE: We are still having trouble understanding your request. Since we would like to make the machine better at understanding people, please push the hard copy button to make a printout of your next request.

INQUIRER: I am simply trying to ask for information about the cost of going to Harvard.

MACHINE: The machine's limited vocabulary includes the words -- data, military, sc'ool, game, self, education, job, summary, and "don't know.' If what you want to do seems related to any of these words, please type the word you think is closest to your request.

INQUIRER: Thank heavens. I want data about Harvard.

MACHINE: You have requested data access.

(Inquirer is linked to the DATA routines.)

In the COLLEGE ORIENTATION, more specific root words like tuition, fees, expenses, bills, cost, and pr' es will be recognized. Any of these particular words will send the inquirer to the unit on college costs. It is possible that, although the inquirer mentions a college word, he is uncertain what he wants to know about college. If he expresses that uncertainty in his statement, he will be branched into that part of the college orientation designed for inquirers who are uncertain. This part of a generalized orientation is diagrammed on page 2 of Chart 2 (page 11).

As the flow chart shows, it is possible for an inquirer to state that he is unsure about some specific topic. If he does so, the script will reflect his uncertainty back to him, and remember that he was uncertain (by setting UNS to 1). This fact can be used later in talking about the specific topic. If the inquirer did not use any specific, recognizable words, we help him by stimulating him to make another statement. The COLLEGE ORIENTATION illustrates the technique.

MACHINE: What would you like to do? (From MAJOR ORIENTATION)

INQUIRER: I am uncertain about college.

MACHINE: (The major orientation recognizes the word college and sends inquirer to the COLLEGE ORIENTATION.)

The activities you can request are: select a school, data about certain schools, exploring your interests, factors colleges use in admission, tests, size of school, college life, transfer, or scholarships. Try to be specific, but use simple words. What would you like to do now.

INQUIRER: It is difficult to say. What can I do.

MACHINE:
1. Choose a college
2. Direct data access
3. Explore colleges
4. Tests
5. Cooperative programs
6. Cost of college
7. Living arrangements
8. Paying for college
9. Size of schools

INQUIRER: I still do not know which thing I want to do.

MACHINE: A list of the more interesting things to do in the ISVD follows. Each suggestion will be followed by a chance for you to type yes and try that or no and get another suggestion.

Would you like to select a college by listing your preference.

INQUIRER: I am not ready for that yet.

MACHINE: (Recognizes the word not)
Would you like to ask for facts about a specific school.

INQUIRER: That is much too specific.

MACHINE: (Machine recognizes neither a positive nor negative word, and so continues.)
Are you interested in factors colleges use in selecting the freshman class.

INQUIRER: Yes, that is exactly what I need.

MACHINE: (Links to the FACTORS IN COLLEGE ADMISSION script)

The overview orientation assumes the inquirer is talking about one college (or job) factor, so it contains very specific vocabulary. Most overview orientations in the ISVD are multiple choice and lecture style. One, the cost factor for college, has been written to conform to Chart 2 (pages 10 to 13). The following protocol is from the COSTS OVERVIEW ORIENTATION.

INQUIRER: Isn't there a special government loan for teachers.

MACHINE: There is an education loan that is specially tailored for people who want to teach. That is the NDEA loan. For every year you serve as a full-time teacher in a public elementary or secondary school (or as a school librarian, psychologist, guidance counselor, or administrator) ten percent of your NDEA loan will be cancelled. This is good up to a maximum of 5 years or 50 percent of the loan.

INQUIRER: How can I get an NDEA loan?

MACHINE: You are eligible for an NDEA loan if you are:

A. Enrolled for at least half the normal full-time academic workload
B. Capable, in the opinion of your school, of maintaining good academic standing
C. In need of a loan to continue your course of study at the institution.

Your chances of getting a loan are improved if your academic record is good since the law specifies that special consideration be given to students who have superior academic backgrounds.

INQUIRER: Do you have to apply to the government to get an NDEA loan.

MACHINE: If you want to apply for an NDEA loan, apply to the school you wish to attend, not the government. Consult your guidance counselor or principal about the loan and contact the appropriate official of the college (usually the director of admissions). Students who are already enrolled in a school should contact the financial aid officer of the school.

INQUIRER: What other kinds of loans can I get.

MACHINE: Do you have any (other) questions about NDEA loans.

INQUIRER: No. What about some other loans.

MACHINE: How about bank loans, loans given by the State of Massachusetts, government loans, or loans intended for vocational education. Do any of these interest you.

INQUIRER: Yes, bank loans.

MACHINE: If you need money for college, a loan may be one way to get it. However, a straight loan turns out to be quite a bit more expensive than either straight savings, or a combination of savings and loan.

MACHINE: Also, there are requirements for borrowing, which vary with the lending company. One may be residence: your parents may have to live in the trading area of the bank. Another requirement may be age. Borrowers generally must be under 60 or 65.

Often loan programs include insurance provisions so that if the borrower (your parent) dies, your education will be taken care of. A loan program may have other provisions that you'd want to check into.

INQUIRER: That's enough about bank loans. How can I get a scholarship.

MACHINE: We have information on some specific loans: NDEA (National Defense Education Act), guaranteed government loans, and vocational training loans. Would you like to see information on any of these.

INQUIRER: No.

MACHINE: What are you interested in now.

INQUIRER: How do people, who can't afford it, pay for college.

MACHINE: You've asked how to pay for college. There are lots of different sources of funds. Here's a list which is probably not exhaustive, not quite complete.

1. Scholarships
2. Loans
3. College work-study programs
4. College assistantships
5. Part-time jobs
6. Tuition payment plans
7. Co-operative plans
8. Veterans benefits
9. Financial help from parents

INQUIRER: What are assistantships.

MACHINE: Colleges grant assistantships to students (particularly juniors and seniors) to help professors and administrators in the college. These are usually based on need, ability, and potential. The financial aid office can probably tell you about the availability of assistantships.

INQUIRER: Yes, but what do assistants do.

MACHINE: You've already seen all the information we have on that subject. What other questions do you have pertaining to college cost.

INQUIRER: Well, what do you do if your parents can't afford to help you very much.

MACHINE: The biggest burden of college cost falls on your parents. They are expected to pay what money they can for your education.

How much money a college considers them able to pay depends on their income and savings, and also on any special circumstances affecting your family. For example, the number of children, recent illness, or support of a grandparent are the kind of thing that would be considered.

If your parents can afford but don't want to send you to college, you have an unusual, but real problem. (We don't know what to suggest. Perhaps your counselor can be of help.)

INQUIRER: Well, I have the USUAL and real problem that my parents can't afford to pay.

MACHINE: We don't quite understand what you mean, would you say it again differently.

INQUIRER: Never mind. Tell me something about co-op programs.

MACHINE: Are you interested in cooperative work programs.

INQUIRER: Yes.

MACHINE: Co-op programs are a special type of work-study program. Instead of holding part-time jobs all year long, students spend

MACHINE: semesters (or trimesters) alternately working and studying. Part of each year is spent working full time, ideally on a job related to the student's major field.

INQUIRER: How much money can they make that way.

MACHINE: How much money you make on a co-op job is variable. It can be very little if you have a low skill job like clerking. However, if you have a skill, or gain one during college (like programming or key punching) there's a possibility of making quite a bit of money.

INQUIRER: What schools have co-op programs.

MACHINE: Are you interested in cooperative programs.

INQUIRER: Yes. What schools have cooperative programs.

MACHINE: Antioch in Ohio and Northeastern in Boston are 2 schools that have co-op programs. Either your counselor or these 2 schools might be sources of further information.

CHAPTER THREE

THE EVOLUTION OF A LANGUAGE

The development of Monitoring in the ISVD from a vague notion to some concrete examples took several years. Before this was possible, the ISVD needed to develop a language which enabled authors to express complex decisions simply. The remainder of this paper presents the evolution of GLURP, the instructional language of the ISVD.*

Like other computer languages, GLURP has grown to fill explicit and implicit needs of potential users. At ISVD this growth process has occurred over the span of the project, as the needs of the guidance staff became clearer to computer personnel. Discussion of this growth process and the interaction between guidance and computer staffs will give a clear picture of the nature of the language. This chapter begins by describing the process by which the language evolved to its present state and concludes with recommendations for its future development.

In the first months of ISVD, the guidance staff articulated the theory of the project. Unfortunately, the language of guidance is not operational enough to serve as a computer language. Terms like self concept, Monitoring, and Access Routine expressed early notions of how the system was to perform, but the computer staff could not translate these concepts into machine instruction without more concrete statements.

It became apparent that the guidance staff would continue to make the non-operational statements, natural to their way of thinking, unles confronted with the limitations of the machine. It was not enough to say "You tell us what you want and we'll do it;" the computer staff has to provide a language through which the guidance staff could communicate with computer specialists. From this immediate need, MYNORCA was born.

The purpose of the original MYNORCA was to provide a way for the guidance staff to specify procedures for computer implementation, to allow them to communicate among themselves at the level of detail necessary for computerizing the project theory, to suggest some concrete behavior for

*This author has operated between the computer staff and the guidance staff. For purposes of exposition, a role as member of the computer staff was adopted.

the guidance staff, and finally, to see what procedures the guidance staff developed that could not be expressed in MYNORCA. These procedures would lead to new directions for language development. The computer staff did not intend to implement either a compiler or interpreter for the original MYNORCA.

The original MYNORCA, with its multiple choice questions, was particularly suited for implementing branched programmed instruction. The organizing concept was the frame, containing text, an indication that a response was expected (the KEYBOARD), and facility to accept answers, give differential feedback, and branch to the next appropriate frame.

Analysis of the answer was in the ACTION step, which looked like this:

```
ACTION
A/Good/*160.00/
B/No, doctors make more than teachers./*160.00/
```

The previous statement is to be interpreted as follows: If the inquirer responds A, say "GOOD" and continue at frame 160.00. If the inquirer says B, say "No, doctors make more than teachers." and continue at frame 160.00. For any other response, continue with the next frame.

There were only six statements in MYNORCA: CRT, AUDIO, SLIDE, KEYBOARD, GOTO, and ACTION. Users had no problem learning these statements, and using them in their original sense. Most authors could handle the statements after an hour and a half presentation and several hours practice.

The computer staff realized that multiple choice was not adequate even to teach the concepts of the ISVD; that it certainly would not be adequate for Monitoring. By proposing an inadequate language, we hoped to stimulate conversation about the features and functions necessary in a guidance language. Instead we forced the authors into the rigid mold of the multiple choice frame. They complained, but could not overcome the restriction. The guidance staff, unfamiliar with computer functions, could not make concrete suggestions. The computer staff unrealistically expected the guidance staff to function as computer specialists. As time passed, it became clear that a programmer needed to be in intimate touch with script authors in order to sense the need for new functions and describe them concretely enough for programmers.

To implement materials written by the permanent guidance staff in MYNORCA, certain functions were needed. These were defined as a model for a way to expand the language. The expanded language was called Summer MYNORCA. It included logical conditions and some convenient ways to examine the inquirer's response.

Typical of the new functions were these:

#(KW:DOG)

> The keyword function caused a match in an ACTION step whenever the word DOG appeared anywhere in the inquirer's response.

#(PHONETIC:DOG)

> The phonetic function causes the answer to match if there was any reasonable misspelling of DOG.

.I.F. ON JOBLIST

> This statement was true if the inquirer typed any job appearing on the JOBLIST. JOBLIST is presumably defined by the author or a standard system list.

AND, OR and EQUAL were used in their usual logical connective sense.

Again the computer staff hoped that the expansion of MYNORCA would stimulate the guidance staff to provide formal definitions of functions they required. Instead the guidance staff sought highly specific solutions to specific problems, rarely recognizing the generality implied. Only when a programmer who understood the instructional and guidance goals of the authors looked at the mass of specific problems did new functional definitions emerge. By the end of the summer of 1967, a substantial number of functions had been added. The Summer MYNORCA users' guide is included in Appendix B.

From these attempts to get the guidance staff to behave as computer specialists, it finally became clear that neither the computer staff nor the guidance staff would change roles. Yet some communication channel was needed -- someone in the middle, who understood both computers and instruction.

Soon after Summer MYNORCA was created, the ISVD hired thirty practicing guidance counselors and social studies teachers for eight weeks of work. Their task was to write scripts in MYNORCA, expressing "Whatever they thought important" about topics in the script-taxonomy for the ISVD.

Formal training in Summer MYNORCA took only one week. Scripts about MYNORCA written in MYNORCA served as the training text. The scripts required active response; readers had to understand the concepts before they could follow the scripts. For a few of the thirty authors the scripts were effective. For the rest, the scripts assumed greater logical ability than they possessed. The language is teachable using self descriptive scripts, but a wider range of expected response should be included.

The scripts on MYNORCA, one of which is included as Appendix C, tried to model good script writing. They anticipated different responses and tried to find the appropriate feedback. Unfortunately, the techniques were not explicitly pointed out, only included. Readers did not pick them up for their own use. When they did recognize a technique, they felt it was only appropriate for teaching about a language or at best for teaching factual materials. They were unable to generalize the techniques to new applications.

Most authors began work without benefit of feedback on the instructional or technical merits of their work. No standards were set; in fact, for the last four weeks the directive of "quantity, not quality" was in force. The result was that material was not revised. Many authors never saw a typed copy of their scripts. No author tested his material on any potential inquirer. One hundred and sixty two scripts resulted from this effort.

Multiple choice was restricting, and these authors did not have the perseverance or time to produce good questions genuinely testing the inquirer's understanding and inclinations. Therefore, the CAI scripts in the ISVD, with minor exceptions, lecture to the inquirers, give occasional tests of comprehension, and provide infrequent concrete feedback. The inquirer is a passive receiver of these lectures, probably bored unless he is intensely interested in the content.

FACTORS IN COLLEGE ADMISSION, one of the scripts written that summer, illustrates these script writing problems.

MACHINE: When a student thinks about a college he generally asks himself, "Would I like to go to this college?" If the answer is yes, or maybe, then the question, "Could I get in?" is asked. Practically all colleges are selective; that is, they have more applications than the number of students they accept. This script is about the factors that make some colleges more "selective" than others.
Do you wish to continue?

INQUIRER: You bet I do (presses send block).*

MACHINE: Good. Let's assume you have asked this question about a college, "Can I get in?"

INQUIRER: (Presses send block.)**

MACHINE: Who will make the decision on whether or not you are accepted by the college? Type the letter of your answer.

A: You	C: Both
B: College	D: Neither

INQUIRER: Me (presses send block).

MACHINE: Type A, B, C, or D.

INQUIRER: A (presses send block).

MACHINE: Yes, to a certain extent you do have something to say in this decision, but the college will also have to pick its students from a group of applicants that are greater in number than the spaces available. This is what makes colleges selective. Yes, part of the decision depends on your own desire to get accepted and part on the college's selection factors.

INQUIRER: (Presses send block.)

MACHINE: What are the factors that are determined by you. Here are some.

A: Desirable Geographic Location

B: Special Course Offerings

C: Financial Standing

D: Reputation of the School

INQUIRER: A B C D (presses send block).

MACHINE: If you have already selected one or more particular colleges to which to apply to you may wish to go on to the factors that colleges use in making their decisions about applicants. If you have not decided which colleges you will apply to, you may look into the factors to consider in this decision now. Some of the answers to the question "Can I get in?" are based on your decision to apply to a college. The rest of the decision is made by the other people, namely college admission staffs.

INQUIRER: (Presses send block).

MACHINE: Colleges collect information about applicants from a variety of sources. Information about student's high school record is sent by the high school, reference letters are received from people who have known the applicants, and test scores

*The machine recognizes "I do" as a positive response.

**There is no question, so no response is expected. However, to make more room on the CRT, it is necessary to press send block.

MACHINE: are sent by the College Entrance Examination Board. The application form itself contains a great deal of information about the applicants. After collecting information from all of these sources the colleges decide on which applicants to accept.

Some of the scripts written in MYNORCA did force major advances in the language. One of these was the OCCUPATIONAL PREFERENCE script. The preference script offers the inquirers a series of choices and prepares a list of jobs that fulfill all his requirements. Two of the questions it asks are: "Are you above average, average, below average in clerical ability?" and "Do you want to work indoors or outdoors?"

To specify a list of appropriate jobs, the preference script describes rules such as these (Durstine and Fitzhugh, 1967):

Begin with all jobs.

If the inquirer answers C1, REJECT any job that has a 1 or 3 or 5 in column 1236 of the DOT tape and does not have 9 in column 1237.

If the inquirer answers C2, INCLUDE any job that has 2 or 3 or 4 in column 1236 of the DOT tape.

Data retrieval and logical processing of this complexity was not specified in the ISVD prior to the OCCUPATIONAL PREFERENCE script.

The other script which forced expansion of the language was the OCCUPATIONAL TEMPLATE. Templates are pattern sentences into which appropriate words are substituted. For example, the OCCUPATIONAL TEMPLATE contains the following sentence and accompanying instructions (Wolff, 1967):

____________ must at least ______________ and should preferably ____________ before entering his occupation.

For any job, print the sentence, filling in the first blank with a "<u>A JOB NAME</u>" or "<u>AN JOB NAME</u>" as appropriate.

Fill the second blank as follows. If <u>Education Required</u> on DOT tape is

<u>Number</u>	<u>Insert Meaning Below</u>
1	Complete junior high school
2	Graduate from an academic high school
3	Graduate from a vocational high school
4	Attend a business school

Number	Insert Meaning Below
5	Attend a technical school
6	Attend a junior college
7	Graduate from college
8	Complete graduate or professional school

Fill the third blank as follows: If Education Preferred on DOT tape is any number in the table above, insert corresponding meaning. However, observe following additional rules: If Education Required is blank, omit sentence. If Education Required is 2 or 3 and Education Preferred is 3 or 2, then use "MUST graduate from either an academic or a vocational high school." If Education Preferred is blank, omit "and should also ________." If Education Preferred is the same as Education Required, omit "and should also."

In current GLURP, the logic above is written:

```
#(A OR AN:#(FFR:RECORD NAME))
#(EQ:#(FFR:EDUCATION REQUIRED)::(#(DS:A:L)):(
MUST AT LEAST #(FFR:EDUCATION REQUIRED)))
#(EQ:#(FFR:EDUCATION PREFERRED)::(#(EQ:#(A):L:AND)
SHOULD PREFERABLY #(FFR:EDUCATION PREFERRED)))BEFORE ENTERING
HIS OCCUPATION.*
```

While this expression is indisputably complex, it is far more concise than the cumbersome English equivalent. A non-programmer is not expected to write the GLURP expression for this idea; he actually writes, #(GET:SCHOOLING), as part of his script. The GET function finds the expression above and interprets it. The author never needs to know it exists. We call GET a service function. The OCCUPATIONAL TEMPLATE created the need for GET, and alerted us to other possible uses of service functions. These are discussed in more detail later.

Someone acquainted with the literature of natural language processing and familiar with the general capability of the computer was needed to make the connection between the needs of the guidance staff and the capabilities

*#(FFR:X) means to look X up in the data file and translate it according to the appropriate table entry.

of the computer group. Since no one at ISVD had the appropriate background, the solution was approached from another direction. The syntax matching techniques of ELIZA* were added to GLURP. These provided most of the desired language processing power. In fact, no author at ISVD has yet suggested processing beyond the capabilities of ELIZA. However, the rules and lists required are tedious to prepare; the syntax is difficult to read.

From its various stages, GLURP has emerged as a collection of languages merged into one. It includes TRAC,** MYNORCA, and ELIZA capabilities. These combine to give logical power, string manipulation, syntax matching, and simple input-output control statements. The combination lacks simplicity of syntax rules, algebraic powers, and logical readability. (The textual portions are easy to follow, but logical flow is obscured by TRAC notation in GLURP.) In its present form GLURP is not far from the computer language outlined by the ISVD in the project's first year.

At that time, ten goals for a CAI language were established. The goals, and the present state of fulfillment are:

1. The language should allow an author to begin to write scripts after one to two hours of instruction.

Most authors can write simple scripts, involving only the MYNORCA functions of GLURP, after only an hour's instruction. Increasingly sophisticated use of the language takes longer to develop. Total mastery of the subtleties of TRAC is beyond those without special aptitude and/or training in computer languages. Fortunately, the subtle distinctions are unnecessary in most instructional applications.

2. The language should be readable by laymen with minimal training.

Here, too, a half hour of training can get most people started. To read the complex TRAC code in GLURP requires as much experience as writing it.

3. The language should allow a group of authors who are not

*ELIZA is a language designed for conversational interaction by Joseph Weizenbaum.

**TRAC is a procedure describing language for interactive consoles designed by Calvin N. Mooers.

computer programmers to create a set of scripts
with the aid of one computer programmer.

GLURP meets this condition well. The ISVD has had thirty short term authors and several authors employed for over a year. One computer programmer has worked together with the guidance staff on scripts.

4. The language should allow an author to use the full
logical power of a computer to express instructional
decisions.

TRAC, in GLURP, meets this goal fully.

5. The language must be capable of growing.

TRAC, an interpretive language for handling macro procedures, grows painlessly since the coding of the interpreter is modular. Functions can be added or deleted at will.

6. The language must be implementable.

GLURP is implemented on the RCA 70/45. The coding requires approximately 16K characters of memory. Storage requirements are 12K characters. In addition, the system needs physical input-output coding, some external disc memory, and a supervisor if the system is time shared.

7. The language must allow authors to modify their
scripts easily.

Since material is not compiled, the scripts can be changed and placed on tape easily. With three to four man-weeks of programming effort, the ISVD computer support system could be modified to allow authors to change their scripts dynamically on-line. This would allow immediate feedback on the corrected text and corrected logic.

8. The language should encourage authors to write
good scripts.

Here GLURP falls down. The functional power and facilities are there, but combining them to express decisions is too difficult. Service routines and a properly constructed manual would help. These ideas are expanded below.

9. The language must allow storage and retrieval
of data.

Both core storage and disc storage are available to the language. References are symbolic. Three of the six members of the permanent guidance staff involved with scripts have used disc storage and retrieval. Both storage and retrieval should be simplified for recurrent situations. Service routines for this improvement are discussed following this review.

10. The language must be general enough to handle applications other than guidance.

Nothing in the language specification limits it to guidance. GLURP is a general purpose instructional language.

In summary, GLURP meets all the requirements for power, but it is less successful on convenience. GLURP is a procedure-oriented language while the ideal language for computer assisted instruction would be problem oriented. Conceptualizing it this way proposed a solution; GLURP needs to be restructured. The ideal language is GLURP with complex procedures readily available to authors. If an author wants to know if an inquirer has typed in the name of a school, he should say #(IS IT:SCHOOL). If he wants to show a slide, he should say #(SLIDE:slidename). If he wants to print the results of the complicated instructions in the template for the last job the inquirer mentioned, he should write #(GET:SCHOOLING). In other words, a variety of service functions should be available to users.

Towards the end of the ISVD, GLURP was modified in that direction. Certain procedures were written and stored so that they could be available at all times for all users. These have enabled the script authors to freely reference complex procedures such as SLIDE, COUNT, WD (next word) and TAKEN (has inquirer taken a script?). Other patterns that recur frequently in scripts were noted. These can be standardized and used to accomplish a task whenever it arises. Such patterns should be part of a GLURP users' guide. Examples of these coding patterns are included as Appendix D. Commonly used word lists like QUEST (a list of question words), POS (a list of ways to say YES), NEG (a list of ways to say NO), UNSURE (a list of ways to express uncertainty) are also available to the authors without special efforts to define them.

Creating new functions is a job for a programmer with empathy and patience for those whose expertise lies in areas other than his own. He must coax the logical expression from the author and then implement the procedure to capture the essence of the intention and to generalize the utility beyond the particular case in question.

Just as a subject matter specialist without aid of a computer specialist falls short of his goal, a programmer alone cannot create material of appropriate complexity. The combination of programmer and subject matter specialist who can talk to each other is essential. Each must go out of his way to learn some of the other's jargon.

The problems of disseminating and teaching GLURP to content specialists requires special effort. For most content specialists, it is not enough to explain the syntax of the language, nor is it sufficient to provide a context in which the language might be useful. Although most computer manuals explain each function in detail and provide a coding example, such a manual is not appropriate for a CAI language, since most people do not generalize the function to their own application. Instead, subject matter specialists must see an example from materials very similar to those they themselves will create before they can understand the function. Before most people can use GLURP, they need to solve practice problems and get feedback on their solutions.

To disseminate GLURP as CAI language, the ISVD should produce:

1. ALGOL specifications for implementing GLURP (Mooers, 1966).
2. A functional description of GLURP (Taylor & Roman, 1969).
3. A User's Guide, with many examples explaining the implications and utility of many of the functions (Roman, November 1969b).
4. A set of service routines that handle many common problems simply (Roman, August 1969).
5. A set of standard coding patterns that solve the common problems script authors confront.
6. A set of problems, each with several solutions for practice and study.
7. Some sample scripts and protocols showing Monitoring in action (Roman, October 1969).

Such a package would not only serve to teach about GLURP, but also prove a more viable model for CAI language manuals.

CHAPTER FOUR

PREPARATIONS FOR A FIELD TEST

Section 1: Expectations and Constraints

During the first two years of the ISVD, we believed that a formal computer language like MYNORCA could provide the necessary communication between guidance staff and computer staff. We thought that both groups would interact around the language, describing new language functions and operationalizing the guidance concepts. Accordingly I concentrated on developing functions in the language to improve it as a comminication device. As we moved toward field testing the system, it became obvious that the language was too impersonal and abstract for satisfactory communication.

In July of 1968, at the beginnning of intensive preparations for field testing the system during the school year '68-'69, I was asked to prepare scripts for debugging during August. As time went on my role in the system grew from preparing scripts to planning the development of the field test.

My change in role with the ISVD arose from a change in our understanding of the communication link between the guidance staff and computer staff. Neither group operating separately through GLURP had adequately realized their potential at the ISVD; by assisting the communication between the groups I was able to help each to function nearer their capacities.

In July of 1968, some of the expectations for the field test included:

1. There would be five video terminals with slide capabilities at five field test sites. Sanders Associates would slightly modify their basic video terminals to control a random access slide projector.
2. The five field test sites would range from an elementary school through the Graduate School of Education. This range implies vast differences in reading level, conceptual sophistication, and career development. The field test materials would reflect

these differences among the inquirers. By testing at five levels we would test the theory of decision-making at different discontinuities.

3. The on-line computer system would handle five video terminals simultaneously and the off-line would load data bases and scripts.
4. There would be six data bases: military, college, trade school, occupation, inquirer, and placement office.
5. All hundred and sixty scripts written in the summer of 1967 would be implemented for the field test.
6. The finished ISVD should contain illustrations of as many "functions" as possible. Developing new "functions" would always be preferable to perfecting old materials. By functions, the ISVD meant new kinds of guidance programs. The following list names some of the functions of the ISVD.
   a) The Life Career Game
   b) The EXPLOREONE routine
   c) The ANALYZE routine
   d) The direct access DATA routine
   e) The Preference valuing routines
   f) The TEMPLATES
   g) A network of instructional routines
   h) ORIENTATION routine
   i) SUMMARY routine
   j) Routines integrating slides
   k) GOODORBADITEM routine
   l) Monitoring routines
7. Materials would be revised during the field test. The test would be dynamic, incorporating feedback from early users to improve the system.

Each of the expectations above was impossible to realize within the constraints imposed on time and manpower by the limited ISVD budget. The progress toward each expectation during the first two years of the project provided further constraints on subsequent development.

A first constraint is the computer time. The ISVD purchased computer time four hours each weekday, and only during those hours is the computer accessible. The time sharing system of the ISVD (TISM) usurps the computer entirely. When TISM is operating, no other utilization of the machine is possible. Script debugging or field test time excludes any concurrent system development.

The staff available for implementation and development set other limits. The group includes: six programmers of various levels of competence ranging from one year's experience with batch to a systems designer, three guidance theoreticians, two graduate students and three research assistants, each a college graduate with no formal training in either computers or guidance. One typist and one key puncher were also involved in the implementation effort full time. Additional clerical help was available.

The main weaknesses of the available staff were lack of training and, in some cases, interest in computer languages; and lack of experience in producing instructional materials.

Money was the last and largest constraint. While there were funds to occasionally supplement the staff or the computer time, we were restricted by funds to the approximate levels described above.

Section 2: Status of the ISVD in June 1968

We have reviewed the expectations for the field test and the constraints imposed by staff, computer time, and money. In this section we will examine the status of the system in June 1968. The starting materials determined the final outcome to a large extent.

1. Video Terminals

   In June Sanders Associates had not completed the modification and testing of their prototype model. The ISVD had no video terminal. Problems modifying the ISVD software package, mechanically installing the devices and with the telephone company each could cause delay. Actually, delivery of hardware is always later than promised, so it too should have been anticipated as a potential setback.

2. Five Sites

The original materials were written covering college factors, occupations, and military service. Since then, a trade school package had been developed. Materials for the elementary school and junior high were to be written during the summer of 1968. Most of the material assumes a well developed vocabulary and good reading skills for the level at which it aims.

3. The Computer System

In June the on-line system operated one teletype terminal. Major modifications were required to make it handle multiple terminals. The system could handle an RCA video terminal, but major systems programming was required to make it the SANDERS terminal. In June the system did not fetch or store data and could not execute a script. Only the most fundamental GLURP functions were operating. Most of the coding for the first phase of work was complete, but debugging was not complete. The off-line computer support system to process data bases and scripts was described and being coded during July.

4. Data Bases

The data bases were in various stages of completion. For five of the data bases, the guidance staff had made the decisions about material and collected the relevant data in coded form. About one half of the man hours required to implement a data base had been put in. The rest of the work had never been done in the ISVD and was not recognized in June as a sizable effort. The expectations of the staff was that only trivial processing would be required to transform the coded data into machine usable form.

5. Scripts

One hundred and sixty scripts were available coded in Summer MYNORCA. Of those, thirteen had been revised. The others were in their original form. The thirteen revised scripts were coded and punched, ready to load into the computer. . ˡl were coded by one of the system programmers. In June, she was the only one who could translate scripts from MYNORCA into GLURP. The ISVD has prepared flow charts of the potential script network. These contained boxes for each script. Related scripts were near each other on the chart, but their formal relationships could not be deduced from the chart.

To create a script network one must know the hierarchical arrangements among the scripts; among these are which scripts related logically to others, which depend on knowledge achieved in others, and which depend on data extracted from interaction with other scripts. These relationships were not indicated on the flow charts available in June.

6. <u>New Functions</u>

   In June the ISVD had four illustrations of PREFERENCE scripts, Templates and many instructional routines. None of the other expected functions existed except as claims in the publications of the ISVD.

7. <u>Revisions</u>

   To revise scripts in response to feedback from the field test, some organized method of obtaining and utilizing it was required. Planning in this area was particularly weak. No programs to supply or process data from the field test were planned or specified. No systematic observation in the field was planned.

8. <u>Staff Communication</u>

   The final critical constraint on the system was the lack of communication between the two groups on the staff. The status of materials discussed before makes the communication between the two groups clear. Let us examine some of the implications of the previously mentioned conditions.

Both the data base and the script materials gave the feeling of what the final product would be like but the level of detail was not sufficient to allow implementation directly from the instructions. Interpretation of the data were implied but the way to achieve it was left unspecified.

These observations are important for several reasons. First, there was far more work involved in implementing the materials than the guidance staff realized. What they thought was a month's work really masked several man-years of effort. The computer staff had failed to specify the full set of subtasks involved in data base implementation in sufficient detail so the guidance staff could do it. (Yee, Little, and Roman, 1969)

Secondly, in June of 1968, total separation of the functions of guidance and computer staffs still characterized the ISVD. The attitude of the guidance staff was that implementing materials involved high class clerical functions, not substantive decisions. This attitude prevented the ISVD from benefiting fully from the advantages of close interdisciplinary co-operation.

These observations imply, thirdly, that no Monitoring was in the scripts. As we saw in Chapter One, Monitoring depends on information collected and categorized in script. One must know exactly what is stored and where, knowledge that can only be achieved by a hierarchical structuring of the scripts. One must be certain that scripts depending on the stored information are not called before the information is stored. Without such detailed flow charting Monitoring is impossible.

Section 3: Materials

As the field test evolved, the main focus of my attention changed. At first my focus was implementation of existing scripts and data bases. Later debugging of the software system and its effects on the field test concerned me most. Then Monitoring and other functions became most important. Finally the evaluation and role of feedback in the system occupied most of my time.

Each of these concerns became necessary as the system evolved. In each area my role as communicator between guidance and computer personnel facilitated progress. Each of these areas needed communication. The remainder of this paper discusses the evolution of the ISVD field test. Since my focus shifted with time during the project, so it will shift in the following sections. In each phase my role as intermediary provides the central theme.

One of the first decisions in July was to proceed toward certain fixed target dates. These were as follows:

August 15	a minimal system for in-house experience
October 14	a working system for demonstration at a national conference
December 1	the field test system, stage 1 ready for in-house debugging
January 6	the field test system, stage 1, ready for inquirers
January to June	further stages of the field test system would be added.

The next problem was to get a firm and manageable assignment of tasks to the deadline dates.

When I took on the job of preparing ISVD for the field test, it was obvious that preparation would have to proceed toward the expectations in stages. Trying to meet all the expectations for the system simultaneously could only result in failure to meet any of them. The authority to make priority decisions rested with the guidance staff. When confronted with alternatives during the first years of the project, the invariable response had been: "Can't we do both?" By July of 1968 we could no longer have the freedom to explore alternatives; we needed to make choices.

There were contradictions among the stated expectations governing the field test; implementing all the scripts and developing illustrations of new functions contradict each other since they draw on the same manpower. The statement that we want to illustrate a variety of functions is inconsistent with having six data bases, each of which repeats the functions of the others. Some operational way of deciding which rule took precedence was needed. In order to aid the decision-making process, I prepared a list of thirty items previously suggested as materials.

In a meeting early in July, the guidance staff decided to implement the college data base and the supporting script network for the August system. The college network included seventy scripts, each three to thirty pages long. At that time, decisions about the other twenty-nine items on the list were deferred until more accurate time estimates were available.

Since there was only one month in which to prepare the materials for August, and the guidance staff wanted all seventy scripts, there was no possibility of substantive editing. Similarly the time pressure was so great that training of research assistants so that they could translate from MYNORCA to GLURP proceeded only at superficial levels and involved only one of the potential GLURPers. Only the mechanical aspects of translation were imparted. It took far longer to make someone else understand GLURP than to encode the difficult passages myself.

By August all the major system scripts and the college data base were encoded, punched and proofread. My assigned task was complete; but the computer system could not yet load a data base or set of scripts. Even if they could be loaded, the video terminal was not working and the time sharing system could not execute a script. There was no in-house demonstration in August.

The next deadline was a national convention of computerized guidance projects. Planning began in August with another priority meeting. By that time, I could estimate the amount of time required to implement existing scripts accurately, and had three research assistants trained to do simple translating from MYNORCA to GLURP.

The guidance staff chose to implement the existing scripts rather than pursue new functions. By making this choice, they effectively determined the priorities attached to each expectation. Following this August meeting, the occupational data base and supporting scripts became high priority items. Next in importance was the Life Career Game, a simulation of career choices which was supposed to give the inquirer experience in long-term planning. In theory the career game provided an obvious seat for Monitoring the inquirer's choice behavior in a controlled environment; in practice, the pressure of an October deadline made serious efforts to Monitor the game impossible. (Roman, November 1969a) Because of these priorities, little work was done on Monitoring routines. All efforts were directed toward implementing existing scripts and working the data base materials into form for input to the computer.

In this phase of the field test development, as in the previous phase, the bulk of translation was done by research assistants with minimal training in GLURP. Serious coding problems were simply solved rather than explained when they came to my attention. This procedure was far more economical of my time than providing the intensive training required to make other people into GLURP coders.

Readying the data bases for the machine involved quantities of clerical work, so we hired three assistants for several weeks to prepare the data. A considerable time was needed simply to supervise the work. Clerical help relieved the monotony of coding, but did not free me to create new materials as I had hoped it would. The largest single cause of interruption was the inability of the coders to accumulate questions instead of asking them when they occurred. This in turn was caused by lack of procedures to "hold" their place when the question occurred for later reference. If they waited to ask, they were unable to remember the context. From the experience I learned how difficult it is to effectively use inexperienced clerical help. Extensive planning and detailed written instructions are virtual necessities for work beyond the simplest kind.

By the beginning of October, two weeks before the deadline all script materials and the data were ready to load into the computer. Unfortunately, the peripheral programs to perform the loading were not ready. The scripts would not go out to this disc because some of them were exactly 2048 characters long and the programs did not follow the specifications requiring them to handle 2048 character scripts. I mention this one seemingly trivial detail because it illustrates the type of problem which held up the operating system; something uncomplicated of itself, but sufficient to stop the entire system from working.

In early October we did have an operating computer system. It could execute those scripts we had on the disc. Unfortunately, it did not properly format the screen of a video terminal. With two weeks to go there was plenty of time to debug the major problems of the script network and have a teletype working system for the demonstration. Instead the ISVD decided to try to complete everyting at once; in the remaining two weeks the computer staff would try to debug the programs that loaded data, try to add GLURP functions to decode lists for the preference scripts, and try to make the system operate properly on a video terminal.

By trying to perfect all the relevant computer software, we lost all script debugging time. We did produce a demonstration, but it showed only the most rudimentary scripts, those requiring no binary decoding, no storage and no sophisticated branching based on input by the inquirer. The scripts could not even tell the difference between yes and no. The ISVD had errored in judgment; it would have been better to do something well than everything poorly. We lost track of the orderly evolution of the system, wasting much time and energy.

Section 4: Software

By mid-October then it was clear that implementation of materials was far ahead of the necessary supporting computer programs. A calm look at priorities and orderly evolution of the system was required. In addition, someone had to specify the relationships between computer development and guidance development in order to integrate the guidance priorities with computer priorities.

The next major goal was an in-house system for the field test to be used by the guidance staff in December. The difficult aspects of the computer software were completed, but some of the functions that made the scripts work and allowed Monitoring were not ready. With materials implementation well under control and several people competent to complete the job with only occasional direction, I became freer to work on materials development and the interaction of computer development with the field test. Because my focus of attention did shift to the computer development during November, I will also shift the focus of this discussion to concentrate on development of computer software. To do so, I will first summarize the status of the computer system in October and its implications for the field test.

Both the on-line system and the off-line system of the computer software contributed problems. The off-line system included programs to load data bases on to the discs and programs to load scripts on the disc. The on-line system contains programs that interpret the scripts, find data and communicate with the inquirer.

The data base programs worked only on the military data base which was used for debugging. The military data base was chosen for debugging since it was the shortest and simplest data base. This virtue turned out to be a fault since all the programs were written with input a d work areas too small to handle the largest data base. This occurred because the programmer did not follow the system specifications. With each new input we found new places where larger size caused the program to halt ungraciously. Also, the Military Data Base did not use some features of the program, and these failed when other data bases tried to use them.

A second problem with the data base programs was that they expected perfect input and failed to give any diagnostic messages when they were unable to process the input. When non-computer specialists are preparing input for the machine, the program should be as helpful as possible; these gave no indication of the cause of trouble.

The lesson here is clear. There was a failure to communicate both the nature and variety of the possible input to the data base programs as well as the intended use of the program. The programmer did not take the human use into account. This was a set of programs in which the users should have consulted with the programmer and the analyst who specified the program. Eventually the problems became so burdensome that I practically rewrote the program to load data. Other programmers drastically modified the other routines to make them faster and easier to use.

The scriptloader routine which processes the scripts to tape worked elegantly; it could load both scripts and data, a consequence of scripts and data looking identical to the system. This unity of form greatly simplified all the programs which create and manipulate data and scripts. However, one consequence of treating data and scripts identically is that the scriptloader program cannot tell the difference between them. The programmer's desire for elegant simplicity caused him to throw away information about the scripts. The machine could have checked for illegally formed script steps if it knew it was dealing with a script; doing so would have saved the authors an enormous amount of debugging time. For six months my suggestions to add script debugging to scriptloader were tabled because higher priority programs were not complete. Finally, in April, I wrote the script debug subroutine myself. Even after three months of on-line debugging, the routine found hundreds of previously undetected script errors.

Again the necessity for communication is clear. The programmers must adapt their routines to the needs of the users. However, the needs of the users cannot be accepted as the only guideline; they must be interpreted by someone who understands the problems of working with computers. Making the needs of the guidance staff clear and operational to the computer staff was one important part of my job. The on-line system provides plentiful illustrations of the interaction of guidance and programs. Any new function that a guidance scriptwriter wanted to use in a script required a change to the working system. Let me illustrate with several changes to the language.

Sometimes an author would like data about a particular school interpreted in different ways. For a simple example, a school that requires the College Board exams has a <u>one</u> in a particular place; those that do not have a <u>zero</u>. One scriptwriter might want to say, "Dartmouth College (does/does not) require College Board tests." Another author might want to use the same data to say "Dartmouth College (is/is not) one of the College Board schools." This is a trivial example and could be handled without a special function. However with more extensive lists, containing hundreds of possible codes instead of just two, it was necessary to provide a way to interpret the same code using different words. This feature was unavailable in the early specifications of the system; only one list could decode any given item. The guidance staff simply felt constrained; I was able to suggest a relatively simple change to the system designer that made this possible.

A second example is the use of teletypewriters as alternate choices to the video terminal. On the video terminal, hundreds of formatting characters are transmitted in a matter of seconds. They are required to display data on the screen, but have no function and annoy users on the teletypes. Because I knew how the formatting characters worked, having in fact specified their sequences, I could confidently ask that they be turned off in teletype mode saving hours of frustration for teletype users. System debugging on-line proceeded more easily because I was able to find scripts that tested specific features or to write new ones that did. Knowing how the system was supposed to work helped to localize problems and facilitate debugging.

Section 5: The Logtape

In March, when it became clear that the system would work and we could have a field test, it became imperative in my mind that the system observe itself in operation and provide objective data for evaluation. The ability to make accurate convenient records of interactions gives computer aided instruction an advantage over other teaching methods. Vast quantities of data can be recorded, sorted, and printed in summary form giving researchers and authors an invaluable aid to their work. In the last section of this chapter I will discuss the method ISVD has adopted to record objective data and some of the uses we plan to make of the data.

Systematic observation of computerized instruction can serve many different functions. While the decisions concerning what would be written on the logtape, ISVD's record of interaction, came after considering possible uses, it will be easier here to discuss the contents of the logtape and then the possible uses of the data.

The ISVD logtape contains, among other things

a. every input the inquirer writes
b. every output the system makes
c. every branch the scripts take
d. every reference to data bases
e. every error that occurs in a script.

In addition, each message contains the time it was written, the user identification, and the length of the message. This logtape provides feedback to all involved in the ISVD: computer programmers, data base maintainers, script authors, and guidance theorists. Let us consider the potential uses for each group in turn.

Computer programmers can use the data to prepare accurate statistics concerning the amount of text put through the system per second, the number of disc references per unit time, the number of users and their effect on response time, and the rate at which data are stored. Each of these measurements is vital to the design of the system. The more accurately they are known, the easier it is to plan modifications to the system software to increase efficiency of operation. Given accurate figures for these measurements, a programmer can locate the bottleneck that is slowing the system down. For example, it is already clear from preliminary study of the logtape that keeping two data records instead of one in memory would reduce the number of fetches considerably. These and other statistics will be extremely useful in describing the next prototype of the computer software.

Data base maintainers can take advantage of the data on references to the data bases. Those records and attributes referenced most frequently can be made more accessible by rearranging the input to the data base programs. Tables of the references to data will be prepared by processing the logtape. These will consitute part of the final report of ISVD, and help to specify the next prototype.

The scriptwriters can reap major benefits from the logtape in the form of debugging aid. If a script does not perform as expected, the cause of the trouble can be localized by following the sequence of branches on the logtape. It is often possible to find out why particular users are reporting difficulties in scripts from the logtape. Inquirers make typing errors, spelling errors and forget key characters. They then loudly blame the system; before the logtape was implemented, hours were wasted tracing non-existent problems. Now minutes locate the error.

Besides helping to find coding errors and logical problems, the logtape record provides concrete feedback to authors about their scripts. All uses of a script can be selected from the logtape. The author then has a record of how his script performs. By reading that record, he can locate places where the script is non-responsive and find which inquirer's inputs were misinterpreted or unanticipated. Using that information the author can change his explanations, provide help steps, and improve his set of anticipated responses.

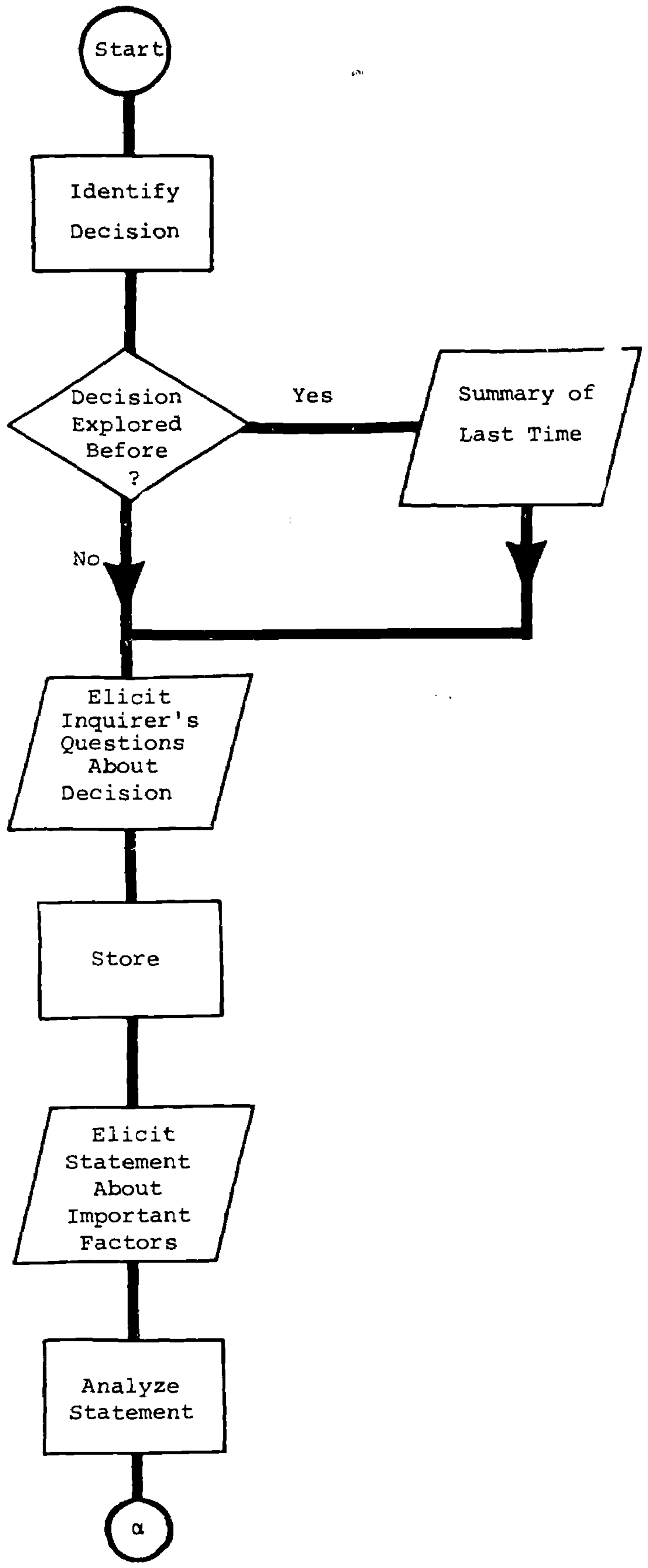
Further computer processing of the logtape can provide a summary of interactions with a script. If certain inputs by the inquirer are key elements in the script, the machine can prepare a list of all inputs at that step and a table of how the script responded to each. If certain branches are important, the computer can summarize the results of those branches. If the author wants to study the change in the inquirer's behavior over several uses of a script, the computer can supply protocols grouped by times used. The possible sorts, selections and summaries are as varied as the author's imagination.

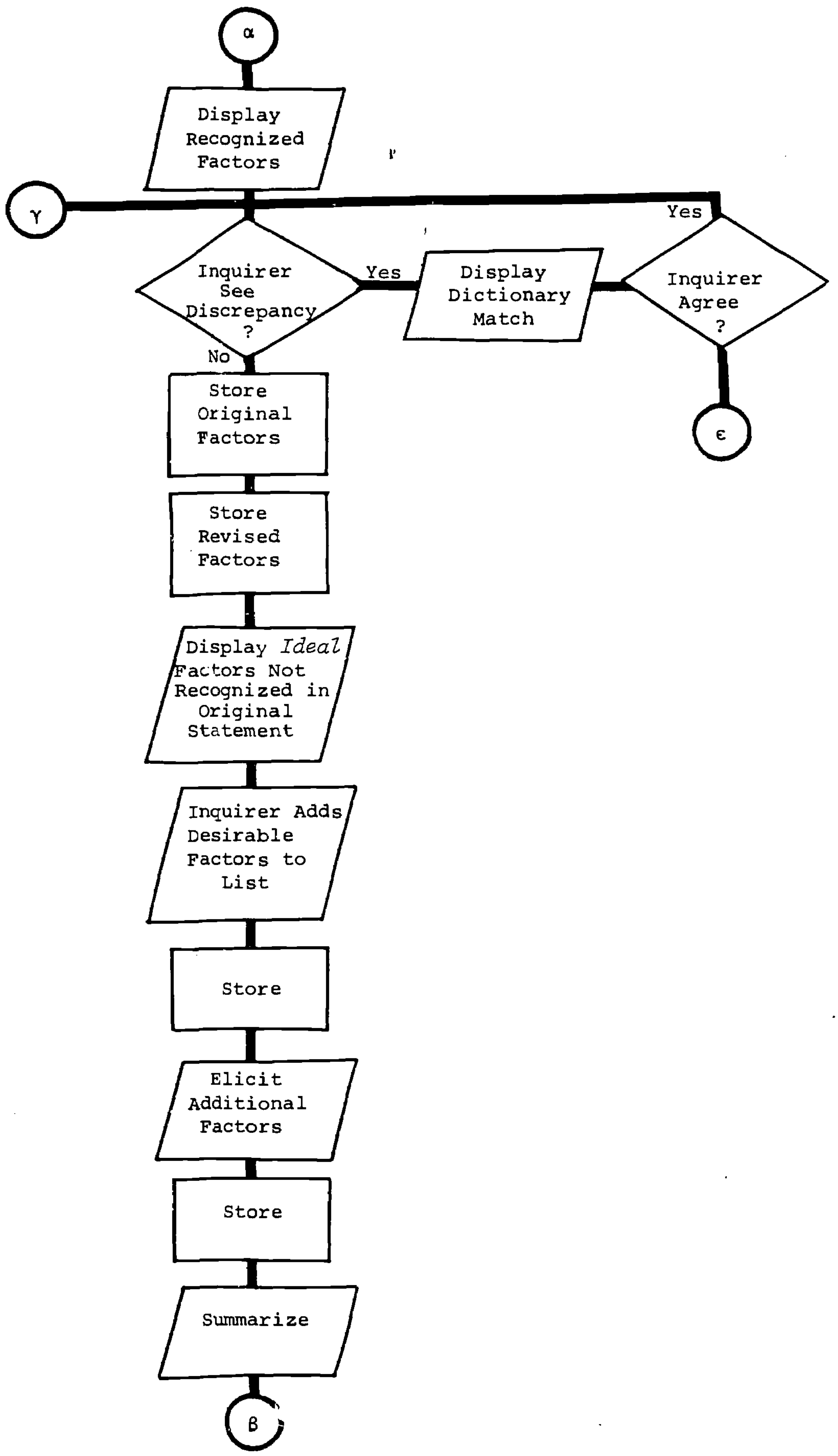
Guidance theorists could develop case studies from the protocols of the system. They could test theory against practice; the testing could lead either to the specification of new scripts or of new theory. The logtape is a rich reservoir of decision-making behavior in concrete form. Careful consideration of its contents will provide guidelines for future versions of the ISVD.

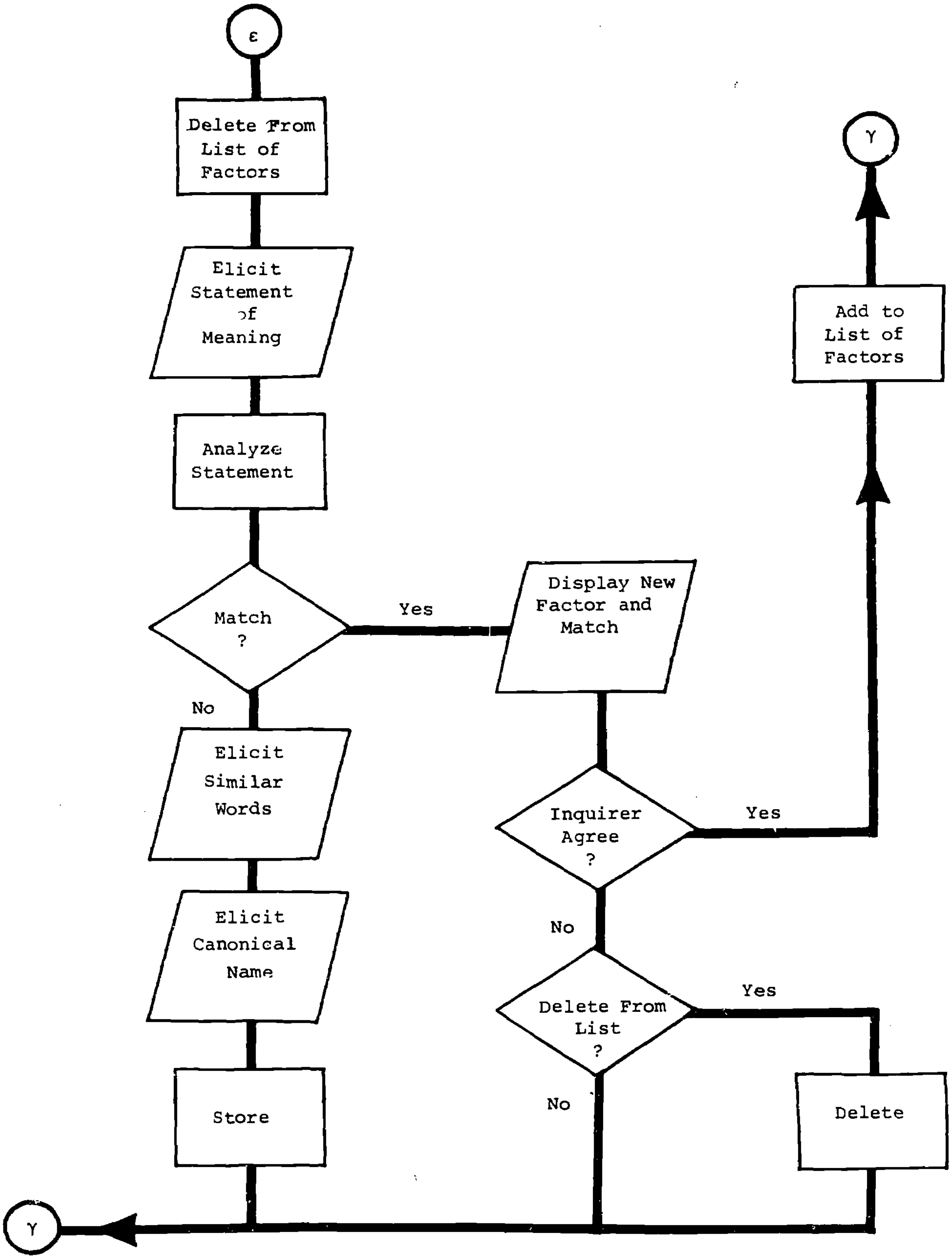
APPENDIX A

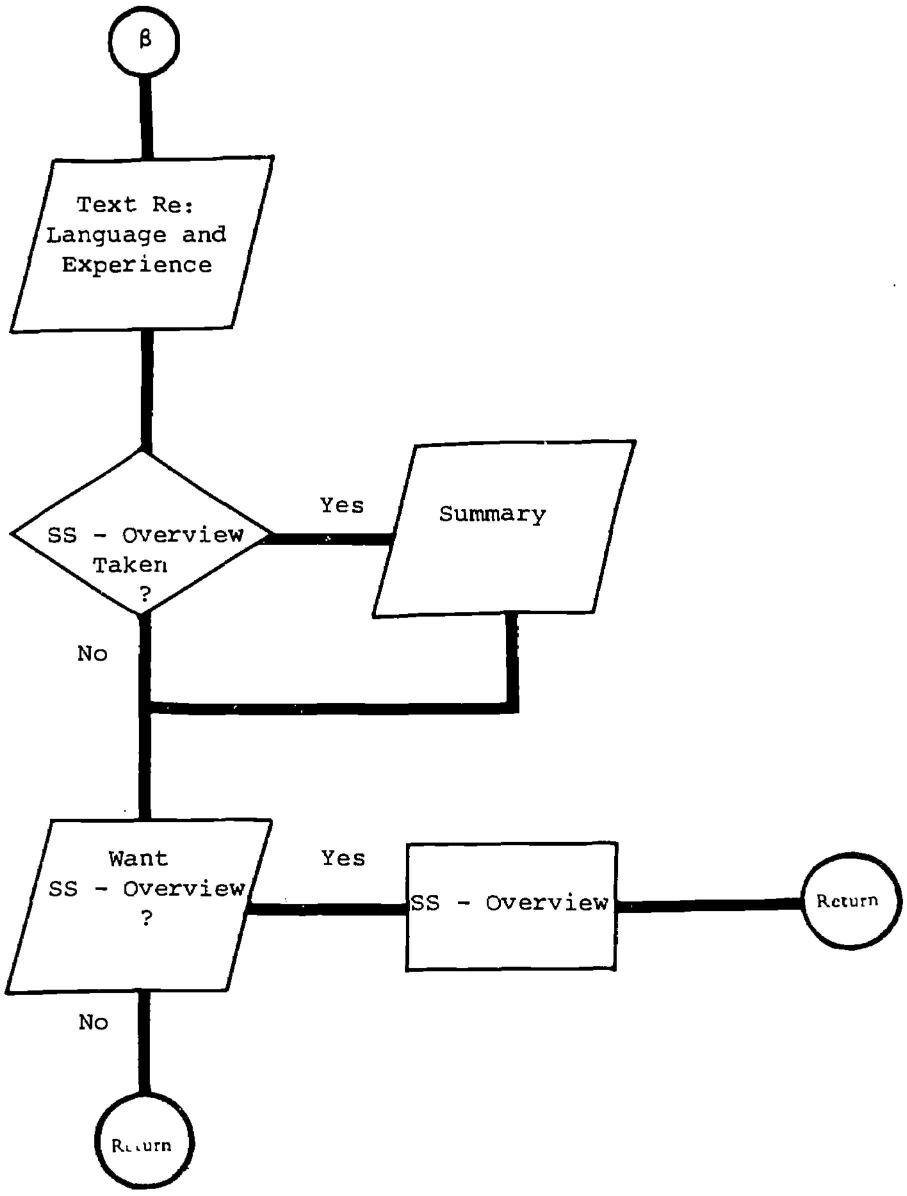
FLOW CHART OF EXPLOREONE

EXPLORONE


Start
Identify Decision
Decision Explored Before ?
Yes
Summary of Last Time
No
Elicit Inquirer's Questions About Decision
Store
Elicit Statement About Important Factors
Analyze Statement
α


α
Display Recognized Factors
γ
Inquirer See Discrepancy ?
Yes
Display Dictionary Match
Inquirer Agree ?
Yes
ε
No
Store Original Factors
Store Revised Factors
Display *Ideal* Factors Not Recognized in Original Statement
Inquirer Adds Desirable Factors to List
Store
Elicit Additional Factors
Store
Summarize
β


ε
Delete From List of Factors
Elicit Statement of Meaning
Analyze Statement
Match ?
Yes
Display New Factor and Match
No
Elicit Similar Words
Inquirer Agree ?
Yes
Add to List of Factors
Y
Elicit Canonical Name
No
Delete From List ?
Yes
Store
No
Delete
Y


β
Text Re: Language and Experience
SS - Overview Taken ?
Yes
Summary
No
Want SS - Overview ?
Yes
SS - Overview
Return
No
Return

APPENDIX B

SUMMER MYNORCA USER'S GUIDE

Roman
7/18/67

MYNORCA FUNCTIONS
A User Guide

This outline of Mynorca functions is designed as a convenient reference for script writers. Some options that were not discussed in the introductory materials are included here for completeness. In addition, several functions that were not discussed at all in the introduction are presented here for the first time.

The secondary function of this material is to allow you to comment on the functions available and to suggest others that you would find useful.

The name of a function is the first thing inside of the parentheses. It is the zeroeth argument of the function. Other arguments are separated from the name by commas. They are numbered first, second, and so on. The dollar sign $ is used to indicate a disappearing function, the sharp sign # is used for replacement functions.

A more technical discussion of these functions is forthcoming for the computer staff.

I. Media Statements

A. Output

1. (AUDIO) = (AU)
This is followed by the literal message to be given. The audio test can contain $(pause) but no other functions.

2. (CRT) = (C)

a. (CRT) This is the normal command. It is followed by text which may contain functions - usually only replacement functions, though disappearing is allowed. The screen is erased and the text is placed on it. The text remains until the next execution of a (CRT) media designation.

b. (+CRT) = (+C) The text following is added to the text already on the screen. The total message remains until the next execution of a (CRT) media designation.

c. (-CRT) = (-C) The screen is erased, and the text is placed on it. The screen is erased again when the first differently numbered step is executed.

d. (+-CRT) or (-+CRT) = (+-C) The text is added to the existing message. All text remains until the next differently numbered step is executed.

TABLE OF CRT WORKINGS

	(CRT)	(+CRT)	(-CRT)	(+-CRT)	(-+CRT)
Is display erased?	yes	no	yes	no	no
When is screen erased again?	at next (CRT) or (-CRT)	at next CRT or -CRT	at next different step	at next different step	at next different step

3. (Printer) = (P)
This will cause the text which follows it to be typed out on paper for the student's reference. The text may contain replacement functions.

4. a. (Slide) = (S) The slide command is followed by $(comment, <u>description</u>) where description is a verbal description of the slide. The description can contain no functions. Alternately, the description may be a reference to a drawing included with the script. The slide will be kept on until the end of the frame in which it appears. <u>Note</u>: This is different than CRT.

b. (-Slide) = (-S) As (S), except the slide is turned off at the beginning of the next differently numbered step.

c. (off Slide) = (off S) This turns the slide projector off wherever it appears. If the projector is already off, nothing happens.

B. Input

1. (Keyboard) = (KB)
The keyboard is a standard typewriter keyboard. It will contain only upper case characters. When this command is issued, the machine will expect the student to respond, and issue some stimulus to tell him so. Presently this is ***. The machine will accept all input up to a control character (now the carriage return) and treat it as the answer to that frame. Capability to erase an incorrect answer is available. Presently this is one " for each character to be removed.

2. (Light pen) = (LP)
The light pen is used to point to the CRT. The input is nearest character to the point. Several answers may be entered simultaneously, incorrect answers can be deleted by student control, and a final control character will indicate the response is done.

3. (tape recorder) no abbreviation
A tape recorder may be supplied with the console. This will be used to take audio messages from the student for subsequent human analyses. The machine will not process such input.

II. Disappearing Functions

A. Control management

1. $(branch, *frame, step) = $(br, *frame, step)
This is used inside a script. The frame is to be found inside the script. The step number is to be found inside the named frame. If the frame is not located in the script, the message

"Script <u>title</u> is missing frame _________."

is created for both student, proctor, and stored in the appropriate script summary where "Branch to it found in frame _____, step _____." is added. A missing step is created for subsequent checking. Finally the next return is taken. If no step number is given, the first step of the frame is taken. If no frame is given, the branch is within the same frame. If neither frame nor step numbers are given, the command is ignored.

2. Link
$(link, new script, *frame of old) This command is used to pass control between scripts. The new script will be executed, starting at its first frame until control hits return. At that point, control will return to the specified frame of the old script and continue from there.

2. If the frame number is left out, return will be to the first frame of the old script. If frame does not exist, it will be an error and handled as branch is.

   If there is no script of that title, appropriate messages are created and the next return is taken.

   (Further explanation is found in the linking function script.)

3. Go to
   $(Go to, script) This function takes a single argument, the script name. This is the script which is to be executed next. No return is specified.

4. Return
   $(return) This function is at the end of any script. Control goes back to the last linked script at the appropriate frame.

5. $(do, *frame) This enables you to repeat a frame later in the script without having to go on to all the branches. This is not clearly defined yet. Don't use it without talking to Dick Roman first.

B. Storage functions

1. $(set, name, contents)
   This allows you to define lists, counters or store special pieces of data. The set function would be used when you need to have certain information available in the script. Thus you might do something like

   ```
   $(set, ok list, fine yes yeah good sure)
   ```

   and then use the list in a nested function like

   ```
   #(member, ok list, #(answer, *15.00)).
   ```

   Another use would be in setting an indicator that the student has taken the script once before.

   $(set, script complete, yes) Later you might test this using .IF. #(script complete) = yes/You have finished/$(return)/

2. $(up, counter, number)
   This will increase the number stored in the place named counter by the amount number. Thus if the contents of a name have been set $(set, times, 4) you could change the number by $(up, times, 3). This would leave the contents of times as 7. This can later be tested by a Boolean action statement.

   If number is omitted, it is assumed to be 1. If it is negative, the algebraic sum is formed.

3. Proctor
   $(proctor, text message) The text message will be typed out for the proctor of the group of consoles. This is a way to get human help while the student is still on line. It

can be used when it is obvious from his responses that he is having trouble. It should not be used for serious guidance problems, but for technical matters like the kid using the equipment incorrectly, or not answering.

4. Summary
   $(summary, text message) This message is placed in a special file that will be printed at the end of the day. Any especially choice bits of information about the child can be put here . . . a job choice, a statement about the system he makes, an educational choice. One thing that should routinely be placed in summary is the direction to see the counselor. At the same time that the child is directed to the counselor, a summary message should be given to the system as a double check. (By the way, this method of getting out of instructional problems is generally frowned upon, as any guidance counselor will tell you.

5. Comment
   $(comment, text) This is used to communicate in your typewritten version with computer people who will eventually implement the script on the console. It is used to make comments, and instruct them about your meaning. Anything you don't know how to do should be explained fully in a comment statement.

6. Pause
   $(pause, seconds) This causes a pause in the presentation of the number of seconds entered in the function. This is most useful in the audio section where pauses often are as important as the words themselves. $(pause) with no time is a normal emphatic pause of about 1/2 second.

III. Replacement Functions

A. Data about the student presently using the script

1. #(age)
   This gives his age in years. It is just a number; no label is supplied.

2. #(IQ)
   This gives his IQ. It is just a number; no label is supplied.

3. #(grade)
   This gives his grade in school. It is the number of grades he has passed. Freshman high school is 9. Senior in high school is 12.

4. #(name)
   This gives his full name. It will sound quite formal in text.

5. #(name, first)
   This gives his first name only.

6. Other data are available. If you would like to use other facts about the student in making your decisions, write a

comment to that effect, and we will make it available to you. Examples are height, weight, eye color, grade point average, number of credits accumulated.

B. Data about present operation

1. #(script)
   This is replaced by the title of the script now operating.

2. #(frame)
   This is replaced by the number of the frame now operating.

3. #(step)
   This is replaced by the number of the step now operating.

C. Data from tables

1. #(copy, *frame 1, step 1, *frame 2, step 2, . . .)
   You may use as many pairs as you like of frames and steps. This will cause the statement in the step named to be copied into the same place and then executed. The frames must be all contained in the same script. If any of the frame numbers are left out, the steps will all be taken from the last frame named. If any of the step symbols are left out, the entire frame will be copied.

2. #(get, table, row, column)
   This looks in the appropriate row and column of the table, and is replaced by the data found there. The table must be specified by you unless we have already done so for you. Use the set function or at least supply a description of the table so we can find the appropriate data.

3. Answer
   #(answer, *frame number, script title) This will be replaced by the student's answer to the frame in the script named. If the script named has not been taken, the place is left blank. If there is no answer, the place is left blank. If the script title is left out of the function, it is assumed to be the present script. If both the script title and frame number are left out, the last answer the student gave is used. (This does not allow you to use previous answers to a frame - only the most recent is available.)

D. Booleans

1. #(member, list, item)
   This Boolean is true if the item is found on the list. The list is the name of a dictionary where the item is looked up. The value is false if item is not on the list. This will usually be used with the answer function nested:

   ```
   #(member, list #(answer, *19.00))
   ```

   This will look for his answer in the list you name. Since the most common use of member will be to look up his last answer, #(member, list) is understood to mean look for his last answer in the list named.

Often you will have to define the list using the set function described under disappearing storage functions.

2. #(first, *frame, script)
This is a Boolean. It is replaced by true if one answer to the frame has been given in the named script. If the student has not responded to the frame at all, or has responded more than once, the function is replaced by false. If the script name is left off, the present script is assumed. If the frame number is left off, the present frame is assumed.

3. #(second, *frame, script) #(third, *frame, script) #(fourth, *frame, script) These all work like first, except that they are true if the student has made two, three, or four responses to the frame respectively.

4. #(keyword, string, *frame, script) = #(kw, string, *frame, script) This Boolean function is replaced by true if the characters in the string are found in the same order in the answer to the indicated frame in the indicated script.

If the script name is left out, the present script is assumed. If the frame number is left out, the present frame is assumed. The value is false if it is not true.

example:

Suppose the answer to the present frame is Washington.
#(keyword,ash) is true
#(keyword, ash) is false (There is no space before ash.)

5. #(phonetic, word)
This is replaced by the phonetic version of the word. This allows most misspelling to be recognized as the same word. Thus

phonetic

fonetic

fonetick

are all recognized as the same word.

By way of warning,

fanatic

bandage are also recognized as matches for phonetic.

The construction

.IF. #(phonetic, anticipated)

is construed as

.IF. #(phonetic, #(answer)) = #(phonetic, anticipated)

APPENDIX C

MYNORCA TEACHING SCRIPT

Single argument replacement function

*40.00 Grade

101.00 (CRT) Another replacement function is [#(grade)]. This is replaced by the subject's grade in school.

Thus you might write a step like [10.00(CRT) So you are in grade #(grade).]

110.00 (CRT) The text printed would be

"So you are in grade 4." if a fourth grader were taking the script and

"So you are in grade ____." if a ninth grader were taking the script.

120.00 (KEYBOARD)

130.00 (ACTION, CRT)

9/That is correct. The output will contain the grade of the present user no matter how many other people have taken the script./*50.00/

4/This was the correct answer for the fourth grader. The value of [#(grade)] changes every time a new student takes the script./*45.00/

/Please try again./120.00/

*45.00

10.00 (CRT) Suppose a seventh grader were using the script. A step that appears is

[9.00(CRT) Being in grade #(grade) is not as hard as high school.]

20.00 (CRT) What would the student see in the blank below?

30.00 (CRT) Being in grade ____ is not as hard as high school.

40.00 (KEYBOARD)

50.00 (ACTION, CRT)

7/Correct, you seem to understand these functions./*50.00/

/Maybe we better review the material presented so far./*40.00/

*50.00

10.00 (CRT) You can have more than one replacement function in a sentence. Both functions are replaced before the message is displayed.

20.00 (CRT) You might write

[10.00(CRT) A #(age)-year-old in #(grade)$^{\underline{th}}$ is really a dolt.]

30.00 (CRT) Both functions would be evaluated and the sentence would read quite normally when it was printed.

*60.00 Get panel

10.00 (CRT) This and many future frames concern the panel provided at the end of the script. Please look at it now.

20.00 (CRT) Did you find it?

30.00 (KEYBOARD)

40.00 (ACTION, CRT)

yes/Great, let's go ahead./*100.00/

no/Take time out to get a copy of it since the rest of the script is meaningless without it./*100.00/

*100.00 table reading

10.00 (CRT) Please look at the panel and type in the height of student B.

20.00 (KEYBOARD)

30.00 (ACTION, CRT)

5'1"/You have located the height of student A. You were asked for student B./*130.00/

5'3"/This is correct. Apparently you know how to use the tables./ *200.00/

7'3"/You have located the height of student C. You were asked for student B./*130.00/

/You made some error in looking at the table. The next frames will give you some instruction on using the tables./*150.00/

*130.00

100.00 You made an error last time, presumably because you were careless. Was it just carelessness?

120.00 (KEYBOARD)

130.00 (ACTION, CRT)

yes/Fine, no sense in bothering you with instruction then./*200.00/

no/OK, let's see if we can clear up the difficulty./*150.00/

```
*150.00  locating data

10.00    (CRT) You were having some difficulty using the table.

20.00    (CRT) The datum of interest is always located at the intersection
         of the row and column named.  Thus if we were looking for student
         C's age, we would choose the column labeled student C and the row
         labeled age.  These intersect at the datum.

30.00    (KEYBOARD)

40.00    (ACTION, CRT)

         14/     /*170.00/

         12/     /*170.00/

         20/This is correct./*200.00/

         11/     /*170.00/

           /     /*170.00/


*170.00

10.00    (CRT) Please come talk with Dick Roman or get someone else to
         explain this to you.

20.00    $(br, *200.00)


*200.00

10.00    (CRT) The obvious nature of this next statement is over apparent.
         We have available the 5 functions mentioned on the panel.  They
         are written

         [#(age)]  [#(grade)]  [#(IQ)]  [#(height)]  [#(name)]

         and have as values the student's

         age, grade, IQ, height, and name, respectively.

20.00    (CRT) We might start a script by saying

         [10.00(CRT) Hi #(name)!]

30.00    If student B were taking the script what would he see?

40.00    (KEYBOARD)

50.00    (ACTION, CRT)

         Hi Tom!/You seem to have been careless, but you have the right
         idea./30.00/

         Hi Dick!/Good, let's continue with some harder stuff./*220.00/

         Hi Susan!/You seem to have been careless, but you have the right
         idea./30.00/

         [Hi #(name)]/No, the replacement function would be replaced by
         the student's name before printing./*210.00/
```

*210.00

10.00 (CRT) Perhaps you were confused in the last example because the square brackets [ ] enclosed the example you were asked to evaluate. They are used to prevent the replacement function from being evaluated in this script, since if they were missing the sentence would have appeared with your name in it rather than as a function.

20.00 $(br,*200.00,20.00)

*220.00

10.00 (AUDIO) As you can no doubt see, the replacement function is a powerful tool for individualizing a script for the student.

Each person can see his own age and name when those are appropriate. There is no need to resort to the hypothetical examples when using a computer.

20.00 (AUDIO) It is this use of other answers to make instruction meaningful that takes computer aided instruction far above programmed instruction in reaching for individualization of material.

*230.00

10.00 (CRT) Suppose we wrote

[5.00(CRT) A #(age)-year-old who is #(height) is tall for his age.]

20.00 (CRT) What would fill the blanks for student B taking the script?

30.00 (CRT) A ____year-old who is ____ is tall for his age.

40.00 (KEYBOARD)

50.00 (ACTION, CRT)

14 5'3"/This is correct. You have made the appropriate substitution for the functions, and probably realize the power of this technique./*300.00/

5'3" 14/You have reversed the order of the two replacements. This would result in saying A 5'3" year-old who is 14 is tall for his age./What if the computer did that to you./*250.00/

/Please try again./40.00/

```
*250.00

100.00    (CRT) You write in one frame

          [6.00 #(frame), you have an IQ of #(IQ) and are #(age) years old.]

120.00    (CRT) Student C is taking your script.

          The things she will see as replacements for the functions ____
          in order are ____, ____, and ____.

130.00    (KEYBOARD)

140.00    (ACTION)

          Susan  80  20/Exactly, now you remember how important order is in
          the business./*300.00/

          /Please try again./120.00/


*300.00

10.00     (CRT) This is the end of the script on single argument replace-
          ment functions.  The next script concerns multiple argument
          replacements.

20.00     $(return)
```

Panel

Datum	Student A	Student B	Student C
name	Tom	Dick	Susan
grade	6	9	12
age	11	14	20
IQ	110	95	80
height	5'1"	5'3"	7'3"
answers to			
* 6.00	physicist	janitor	basketball player
* 9.00	income	income	education
*11.00	A	B-	C+
*12.00	janitor	guidance counselor	teacher

Questions *6.00
What job would you like to hold?

*9.00
Are you interested in income, education, or future prospects of the job?

*11.00
What was your academic average in fourth grade?

*12.00
What job would you most dislike?

Job data table

Job	Income	Education	Future prospects
physicist	$15,000 per year	graduate training	increasing demand
janitor	$1.40 per hour	high school	no change in demand
basketball player	$ 7,000 per season	junior high school	no change in demand
guidance counselor	$13,000 per year	college graduate	decreasing demand
teacher	$10,000 per year	college graduate	no change in demand

REFERENCES

Durstine, Richard M.; and Fitzhugh, Lynne. (1967) "Choosing a Job by Characteristics." In "A Rudimentary Demonstration for the Information System for Vocational Decisions." *Project Report No. 11*. Cambridge, Massachusetts: Information System for Vocational Decisions, Harvard Graduate School of Education.

Mooers, Calvin N. (1966) "TRAC in ALGOL, Level Zero Standard Processor." (Draft) Cambridge, Massachusetts: Rockford Research Institute.

__________. (March 1966) "TRAC, A Procedure-Describing Language for the Reactive Typewriter." In *Communications of the ACM*. 2 & 3.

Rigney, Joseph W. (1962) "Potential Uses of Computers as Teaching Machines." In *Programmed Learning and Computer-Based Instruction*. John E. Coulson (ed.) New York: Wiley and Sons.

Roman, Richard A. (August 1969) "Storage in the ISVD: Theory and Practice." [in-house document]

__________. (October 1969) "The Script Network of ISVD." [in-house document]

__________. (November 1969a) "Implementation of a Career Decision Game On a Time Shared Computer: An Exploration of its Value in a Simulated Guidance Environment." *Project Report No. 25*. Cambridge, Massachusetts: Harvard Graduate School of Education, Information System for Vocational Decisions.

__________. (November 1969b) "A User Guide to GLURP: A Computer-Assisted Instruction Language." [in-house document]

Taylor, Ann; and Roman, Richard A. (1969) "A Functional Description of GLURP." [in-house document]

Taylor, Edwin F. (ed.) (March 1968) *Eliza, A Skimmable Report on the ELIZA Conversational Tutoring System*. Cambridge, Massachusetts: Education Research Center, Massachusetts Institute of Technology.

Wolff, Laurence. (April 3, 1967) "A Procedure for Writing Job Descriptions by Computer." Working Paper. Cambridge, Massachusetts: Harvard Graduate School of Education, Information System for Vocational Decisions.

Yee, Patricia; Little, Priscilla; and Roman, Richard A. (forthcoming) "Preparing and Implementing a Data Base." Cambridge, Massachusetts: Harvard Graduate School of Education, Information System for Vocational Decisions.