ED 179 195 IR 007 856

TITLE Individualized Instruction for Data Access (IIDA). Quarterly Report No. 4.
INSTITUTION Drexel Univ., Philadelphia, Pa. Graduate School of Library Science.; Franklin Inst. Research Labs., Philadelphia, Pa.
SPONS AGENCY National Science Foundation, Washington, D.C.
PUB DATE Mar 79
GRANT DSI-77-26524
NOTE 33p.; For related document, see ED 168 462

EDRS PRICE MF01/PC02 Plus Postage.
DESCRIPTORS *Computer Assisted Instruction; *Computer Programs; Data Bases; Higher Education; *Individualized Instruction; *Information Retrieval; Information Systems; On Line Systems; *Search Strategies; Systems Analysis
IDENTIFIERS *Computer Software; DIALOG; *Individualized Instruction for Data Access

ABSTRACT

A brief summary of the progress and status of the Individualized Instruction for Data Access (IIDA) project is followed by a report focusing on the principles and rules for analyzing a computer search performed using the IIDA software, and adaptation rules for the deferment of error messages in the case of frequent violations of the same rule. The IIDA computer software system consists of four main programs, three in the exercise mode--a preliminary introduction to searching, a practice search exercise, and more advanced training, and one in the assistance mode designed to assist the user in the performance of a search of his own choosing. Twelve principles are enumerated and described for the analysis of such a search: string and cycle analysis, repeated commands, null sets, unused sets, thrashing, dwelling, relevance, print control, use of null sets, time, syntax, and slack and help requests. Diagnostic rules given are concerned with the general configuration of the search and procedural diagnostic rules. Adapting rules deal with the nature of rule adaptation, the assignment of rules to classes, and the evaluation of rule adaptation. Seven references are listed. (CMV)

INDIVIDUALIZED INSTRUCTION FOR DATA ACCESS (IIDA)

Quarterly Report No. 4
March, 1979

Drexel University, School of Library
and Information Science
Franklin Institute Research Laboratories

NSF Grant No. DSI 77-26524

TABLE OF CONTENTS

		Page
1.	Summary of Progress and Status.	1
2.	IIDA Diagnostics.	3
	2.1 Diagnostic Principles	3
	2.1.1 String and Cycle Analysis	4
	2.1.2 Repeated Commands	5
	2.1.3 Null Sets	5
	2.1.4 Unused Sets	5
	2.1.5 Thrashing	6
	2.1,6 Dwelling	7
	2.1.7 Relevance	8
	2.1.8 Print Control	8
	2.1.9 Use of Null Sets	9
	2.1.10 Time	10
	2.1.11 Syntax	10
	2.1.12 Slack and Help Requests	11
	2.2 Diagnostic Rules	12
	2.1.1 General Configuration	12
	2.2.2 Procedural Diagnostic Rules	15
3.	Adapting Rules.	16
	3.1 Nature of Rule Adaptation	16
	3.2 Assignment of Rules to Classes	18
	3.3 Evaluating Rule Adaptation	18
4.	References.	20

1. Summary of Progress and Status

Viewed as a computer software system, IIDA consists of four main programs:

Exercise Mode

1. Exercise 1: A preliminary introduction to searching.
2. Exercise 2: A practice search exercise, using only the basic search commands. The search is assigned by the system.
3. Exercise 3: More advanced training.

Assistance Mode

4. The assistance mode program: The user performs a search of his own choosing, using almost any DIALOG command. The function of IIDA in this mode is to provide assistance in the performance of the search. Those commands excluded are PRINT and those associated with Selective Dissemination of Information (SDI), none of which lend themselves to having results immediately evaluated.

At the present time, all programs are operational. The evaluation plans detailed in IIDA Quarterly Report Number 3 (1) will evaluate program 4 (IIDA as an assistant) and program 1-3 as a whole (IIDA as a teacher).

The program logic description in IIDA Quarterly Report Number 1 (3) is partially obsolete and will be replaced in a subsequent report with an up-to-date account. IIDA Quarterly Report Number 3 (1) deals with evaluation plans. Current plans for subsequent reports are:

QR No. 5, June 1979:	The IIDA teaching programs
QR No. 6, Sept. 1979:	IIDA computer programs, revised documentation
QR No. 7, Dec. 1979:	Preliminary evaluation results
June 1980:	Final Report

Other recent publications resulting from the IIDA project are:

Fenichel, Carol Hansen, Online Information Retrieval: Identification of Measures that Discriminate Among Users with Different Levels and Types of Experience, Ph.D dissertation, School of Library and Information Science, Drexel University, Philadelphia, 1979.

Meadow, Charles T., "The Computer as a Search Intermediary," ONLINE, July 1979, 54-59.

The schedule for completing the project is, in condensed form:

Begin in-classroom evaluation at Drexel	June 19, 1979
Begin Engineering Technical Writing evaluation	June 19
Begin Library Science evaluation	Aug. 6
Begin Industrial Laboratory evaluation	Oct. 1
Complete any needed revision of computer programs	Sept. 15, 1979

2. IIDA Diagnostics

2.1 Diagnostic Principles

In this section we review the nature of the diagnostic analyses of a search performed by IIDA. Individual procedures may vary, as a result of experimental trials, but we expect that the basic areas of analysis will remain constant throughout the remainder of IIDA development. The major areas are listed below and the specific diagnostic rules are given in the next section.

In studies of other intelligent teaching systems (4,5) we find the general pattern that the program "knows" the subject material very well, or perfectly. This means that the program knows exactly what is wrong if the objective of the student is to diagnose a fault, or knows exactly how a problem should be attacked and the kinds of errors students are likely to make if the object is to analyze execution of an algorithm, e.g., addition. Such a model of the search process continues to elude us. We cannot store within the computer a model of exactly how a search should be performed and then compare the student's performance with it. We are designing both the model and the comparator simultaneously. Unfortunately, we need the results made available by the comparator in order to refine the model. The entirety must be viewed as an iterative process, not to be considered complete at the conclusion of the IIDA project.

It is the principal objective of IIDA diagnostic procedures to make the user aware that problems exist and to suggest that he give conscious thought to the structure of the search as a whole, to the fact that a search is a structure to be designed and, if the design proves flawed, to be redesigned. What is important is that we have this level of communication with the user, not that we be able to detect exactly the true nature of his "error." As is often said about contending parties in human affairs, the important thing is to start the dialogue, not necessarily to be certain that the precise flaw or source of conflict has been discovered.

2.1.1. String and Cycle Analysis. A string is a sequence of commands of the same class. For simplicity, COMBINE commands are a class, as are TYPE commands. For a complete breakdown of DIALOG commands, by class as viewed in IIDA, see Table 1. Since IIDA was designed before Lockheed announced SUPER SELECT, that capability is not taught or handled by IIDA. Conventionally (6) searching is done by progression from the select-expand class of commands, to combines, to types, to prints. Also conventionally, searchers go back and repeat this sequence, or a portion of it, several times during a search. A cycle is one progression from the select-expand class of commands to the print commands. A cycle ends whenever a searcher goes back to a previous class in this sequence. It is not necessary that a cycle have an instance of each class of command. The first level of IIDA diagnosis is to examine the length of strings and cycles.

Neither of these measures (string and cycle length) is, in itself, proof of failure or even lack of success in a search. On the other hand, an

c_type_maj	c_type_min	Command	Comments
1	0	BEGIN	
1	1	.FILE	
1	2	END	
1	3	END/SAVE	
1	4	END/SDI	
1	5	LOGOFF	
1	8	LIMITALL	
2	0	EXPAND	yields a segment from alpha index
2	1	EXPAND	yields a segment from thesaurus
2	3	SELECT	single descriptor only
2	4	SELECT	single descriptor for E-table
2	8	PAGE	used in context of EXPAND
3	0	COMBINE	all operators are "AND"
3	1	COMBINE	all operators are "OR"
3	2	COMBINE	mixed "AND" and "OR" operators
3	3	COMBINE	(same as above - distinction to be determined)
3	4	SELECT	multiple descriptors from E-table
3	5	LIMIT	
3	6	SELECT	contains an infix
3	7	SELECT	term is truncated
4	0	TYPE	with set argument
4	1	DISPLAY	with set argument
4	2	TYPE	with accession # argument
4	3	DISPLAY	with accession # argument
4	8	PAGE	used in context of TYPE/DISPLAY
5	0	PRINT	
5	1	PRINT	contains sort fields
5	2	PR-	cancel print command
6	0	EXPLAIN	
6	1	DISPLAY SETS	
7	0	.RECALL	
7	1	.EXECUTE	
7	2	.RELEASE	

Table 1. Command Classification in IIDA

are being created but never used in later commands, the assumption is that they are perhaps ill-considered. This may be an indication of thrashing (See 2.1.5), the rapid change of direction of searching. IIDA counts the number of unused sets created in each cycle, but comments to the user only when, after a cycle, the number of unused sets is greater than it was at the end of the previous cycle, indicating a possible trend toward creating an increasing number of sets never used. Null sets are not charged to the unused set count. This is considered to be a moderately serious problem.

2.1.5. Thrashing. This is the fault of changing "direction" of a search rapidly or often, without pursuing any given direction far enough to see if it can work out. Guessing at possible searcher motivation, it might come from overly easy discouragement with preliminary results, such as getting a null set or an extremely large one from one search formulation, and not trying to modify that formulation, but instead turning to a completely different approach. At its worst, thrashing is indicative of random, uncontrolled behavior. We cannot, of course, measure "direction" precisely. We do it in somewhat arbitrary fashion by taking a measure of the similarity of the boolean expressions in successive COMBINE commands. One consequence of this is that thrashing can occur only within a string of COMBINEs. The measure, called the similarity index, is made up from the percentage of implied terms in common to the two expressions. By "implied" we mean that an expression is expanded, by replacing set numbers with their defining terms, and then the similarity count is done on terms, not set numbers. We compute the number of terms in common, divide that first by the number of terms in one expression, then by the number in the other, and take the mean. This gives the similarity index between any two COMBINE commands.

Thrashing is a mildly serious problem. A little bit of it does no particular harm. A great deal of it indicates a searcher is in trouble, but we would also notice an excessive string length, and possibly other faults as well. Hence, detection of thrashing is primarily of benefit in pin-pointing the nature of the cause of the excessive number of COMBINES

2.1.6. Dwelling. This is behavior opposite to thrashing. It is remaining with a search concept when it may be time to give up and try another approach. Again, we recognize dwelling only within a string of COMBINEs. The similarity index is used to recognize dwelling, but this time it is in cases of similar COMBINE expressions, rather than dissimilar, that an IIDA message is sent to the user. Also used is the concept of convergence or divergence. The searcher will have been asked at the beginning of an IIDA-mediated search to state his search goal, as a numeric step function. If a string of similar COMBINEs shows progress toward that goal, we are less concerned about dwelling; if he is progressing away from the goal but remaining with the same basic search concepts as indicated by the similarity index, he is dwelling. Regardless of convergence or divergence, if a similar string is too long, he is dwelling. Dwelling can be a serious fault in that it might be indicative of a lack of understanding on the part of the searcher as to what a mechanical search, performed by someone of his skill level, is capable of producing. Perhaps the best example is a case in which the data base simply does not have much information on the sought-for subject. Excessive dwelling does not help much; the problem is to realize the meagerness of the lode.

2.1.7. Relevance. Much has been written about the nature of relevance, all of which we neatly side-step in IIDA. Relevance is a judgment made by a searcher about the value of search results to him. We cannot control how he thinks about the concept and we do not go into what might be boring and diversionary teaching on the subject. IIDA does ask the searcher to make a relevance judgment after each printing of a record (using the command TYPE or DISPLAY), unless the display was done using format 1 which shows only accession number. The judgment is based on a scale of 5, from irrelevant to highly relevant. Diagnostics are performed on the average relevance figures for a group of records displayed with a single command, as long as at least three records are typed. Relevance diagnostics are not based upon faults, or search errors, but they are used to direct the attention of a searcher to the fact that a particular set seems not to be fruitful, or that a previously examined set yielded higher relevance scores. Although low average relevance figures give no hint as to the remedy, the problem detected is a serious one -- the searcher has come to the end of a cycle and is dissatisfied with the results according to his own definition of relevance. If this condition recurs repeatedly at the end of cycles, the entire approach to the search comes into question.

2.1.8. Print control. There are two aspects of print control errors. The first is use of a format that is likely to be uninformative. The extreme case here is a DIALOG format which gives accession numbers only and is of no help to a searcher for browsing purposes. The other is excessive printing, typically by typing out too many records in a longer format. While some longer-format typing is necessary, users must be taught to be careful of the use of this resource. In the case of selection of a format, format 6 (title and accession number only) is not too informative, but is frequently used, at least to make an initial judgement about the general relevance of a set. We assume that novice searchers would, at least sometimes, use more information for decision making. Except for the use of format 1, the problem is not critical to the success of a search. It can be serious to the cost. Of course, if a searcher never uses a format other than 6, and also shows inability to find new descriptors when his first choices fail to work out well, then this would be a serious problem. For this reason, IIDA will take the initiative in searching previously viewed records for new terms if the searcher does not do it himself, through a facility called <u>descriptor assistance</u> (<u>da</u>). Invoked by the user, <u>da</u> provides a fast way for a searcher to examine descriptors in retrieved records without use of the TYPE command.

2.1.9. <u>Use of null sets</u>. This class of faults is concerned with cases in which a searcher uses a set which is null in a <u>COMBINE</u>, <u>LIMIT</u> or <u>TYPE</u> command. In the latter two cases, the result will be again a null set or no typing at all. In case of a <u>COMBINE</u>, the result depends on the logic used. Since it has been our observation that novice searchers are often guilty of not reading system messages, including those labeling a set as null, repetition of this fault indicates that the user is not basing his search expressions on results, but on how he hoped the search might work out. (Not now planned, but conceivable, is a diagnostic to see if positive steps are taken to reduced sets of unusually large size, by LIMITING, for example.) Repeated use of null sets is a serious problem. Note: The detection of null set usage is a function of the IIDA parser, rather than a diagnostic program.

2.1.10. Time. The time between a machine's message and the input of the next command by the user is measured. Excessive consumption of time is, of course, expensive and is also an indication of searcher difficulty. In addition to its pedagogical value, the time measurement can be used to terminate a user's session if the delay is extreme, indicating that the person has probably left the terminal but not logged off. Time is not critical to search logic, but long delays inhibit the extent to which a person can remember all he has done previously and, in early experimental work has been found a good discriminator between experienced and inexperienced searchers (7).

2.1.11. Syntax. Syntax errors are caused by a lack of knowledge of the language, typing errors, and sometimes confusion. IIDA can do little about typing errors, but it is essential to catch syntactic errors, either in IIDA or DIALOG. IIDA gives more information to the errant user about recovery; hence we attempt to catch all errors that DIALOG would itself catch. The remedy IIDA offers for errors, once detected, is to provide information about proper form of commands if the user requests it. We frequently see beginners being confused about command formats, particularly confusing one command with another, or one search language with another. While a trained human observer can readily see this pattern, and while we can program a computer to see it, this degree of analysis of the nature of errors we have deemed too expensive.

Hence, we restrict ourselves to detecting the fact of an error, to stating to the student the assumption we have made about a command, part of which is in error, and hope that he will take the initiative in resolving any remaining problems. Suppose the user has created a set 7 with 9 records. Later in the same search he enters the command: TYPE 7/6/10-18. IIDA will identify this error and call it to the user's attention with:

```
****SYNTAX ERROR: command assumed: 'TYPE'; argument assumed: '7/6/10-18'
The argument is acceptable up through '7/6/'. However, the first record
number is greater than the total number of records in the set.
```

Syntax errors are critical in importance from the point of view of catching them, but, since all are caught, either by IIDA or DIALOG, their importance to the outcome of the search is minor.

2.1.12. <u>Slack and help requests</u>. This is a specialized problem for monitoring. IIDA will have self-modifying thresholds for some diagnostic variables (See Sect. 3), with the general rule that they will become more stringent as a particular error with which a variable is associated is repeated. A user may request that this tightening of control by IIDA be eased, asking for a slackening of control. Ideally, he should request this when he has realized what the problem is and has taken remedial action. But, to avoid abuse, we will monitor the use of this slack option. We cannot, at this time, predict the seriousness of the problem.

IIDA will also monitor the use of <u>help</u> by the searcher. In the case of both <u>help</u> and <u>slack</u>, their use is not directly a search logic problem but may indicate a user who is confused or not using this system seriously.

2.2 Diagnostic Rules

The diagnostic rules have changes only slightly from the version described in IIDA Quarterly Report No. 2 (2), but they are reviewed here in their entirety.

2.2.1 General configuration. In its present configuration IIDA uses diagnostic rules to detect procedural errors other than syntactic errors. In addition, its parsing program detects syntactic errors in DIALOG or IIDA commands as well as certain usage errors such as using a null set in a COMBINE or TYPE command.

We present, first, a decision tree which is the logical equivalent to the diagnostic procedures, but does not represent the design of the algorithms. In this tree, shown in Figure 2.1, classes of errors, rather than specific, detailed rules, are illustrated. The warning control program (WCP) (2) is invoked often to make the final decision on what message goes to the user, following one or more detected errors.

Figure 2.1 is an abstract of the full tree. It shows checks for (1) user response time, (2) syntax errors (detection of syntax error in a DIALOG or IIDA command causes rejection of the command, hence there is no further diagnosis), (3) repeated commands, and (4) excessive string length. If a cycle has just ended (5) a check is made for (6) unused sets and (7) low average relevance. If not at the end of a cycle, tests are made for (8) null sets generated, (9) LIMIT/COMBINE command errors (e.g., dwelling or thrashing), (10) TYPE/DISPLAY errors (excessive printing or relevance problems).

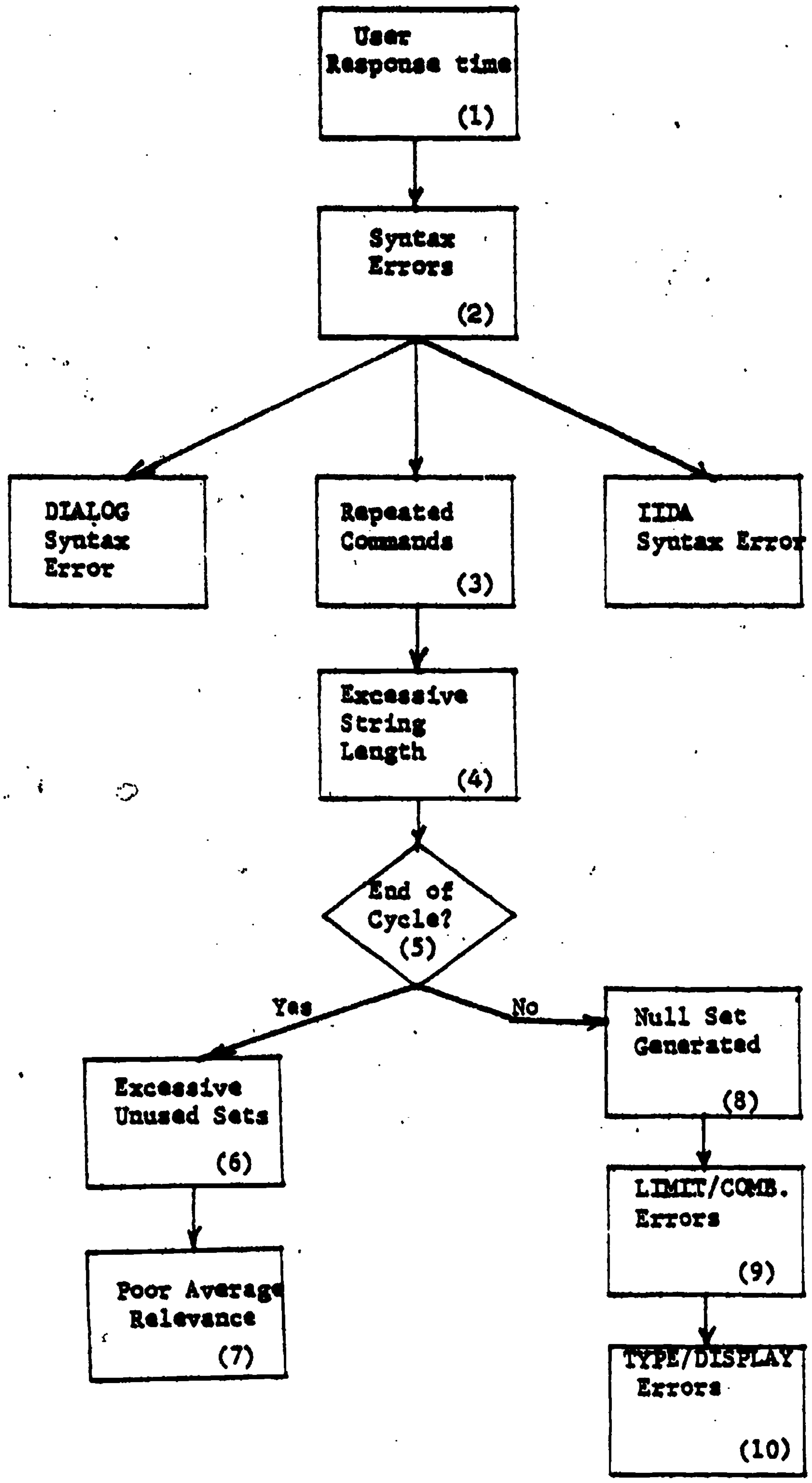

Figure 2.1. Condensed diagnostic decision tree.

Figure 2.2 shows the diagnostics in greater detail. In part a, if user response time (1) is excessive, then a check is made (2) to see if the user has been previously warned. If so (3), his session is terminated. If not (4) he is warned that some response is necessary to avoid termination. If time is not excessive, then where a response is received, it is first checked for syntax validity (5).

An invalid command is rejected (6) with as precise identification of the nature of the error as possible. If the command is a DIALOG command (7) then it is checked for repetition (exact or essential, e.g., C 1 * 2 is essentially a repetition of C2 * 1) (8). If there is a repetition, the WCP is informed (9) and in either case we next check for string length (10). If string length is excessive, the WCP is informed (11). In either case, we move to 2 in part b.

Return, now, to the "DIALOG command" decision (7). At the present stage of development three classes of IIDA command are recognized: HELP, SLACK and DESCRIPTOR ASSISTANCE. If the command is HELP (15) it is executed (16). If not HELP, then DA is tested for (16). If it is DA, that program is executed (17) and a new command accepted. If the command is SLACK, it is tested for repetition (18). If repeated (19), the user is informed. If not repeated (20), it is executed and, in either case, another command is sought. If the command is DA (descriptor assistance) it is executed and, at the end, another command is sought.

In part b of the tree, a test is made for the end of cycle condition (21). If present (i.e. an earlier class of command is used) a test is made for an excessive number of unused sets having been generated during that cycle (22). If the test is positive, WCP is informed (23) and the program proceeds to

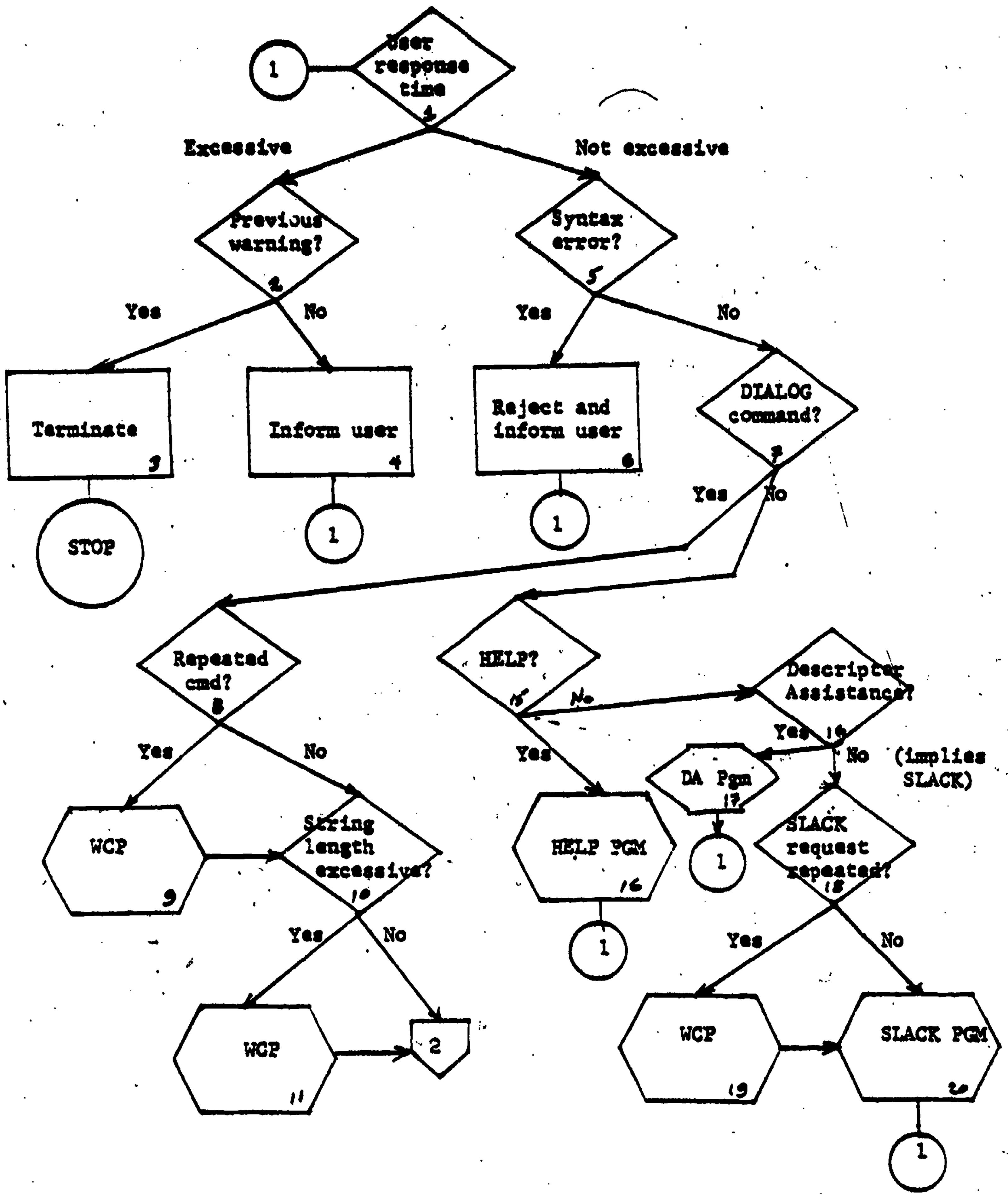

Figure 2.2a. Detailed diagnostic decision tree, part a.

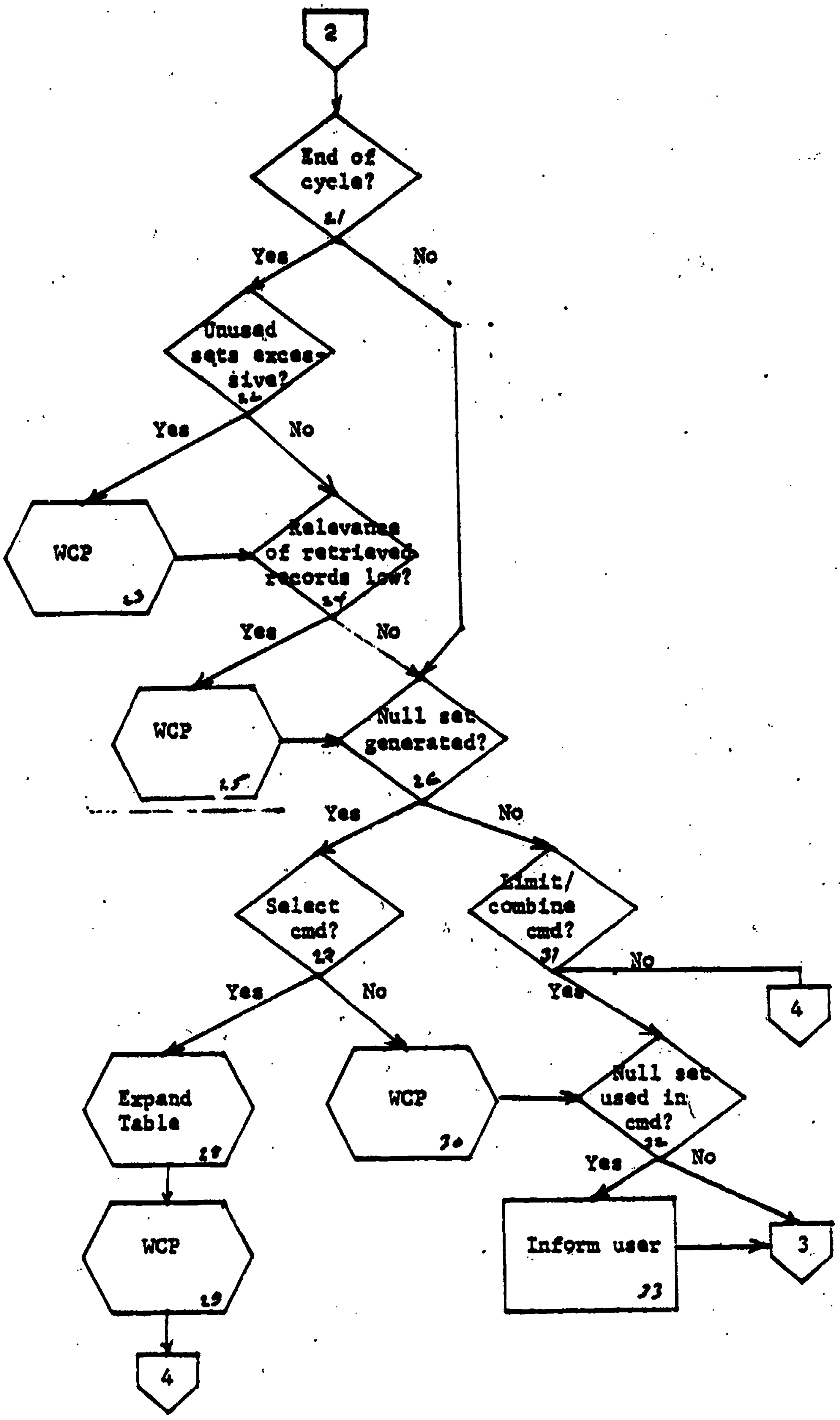

Figure 2.2b. Detailed diagnostic decision tree, part b.

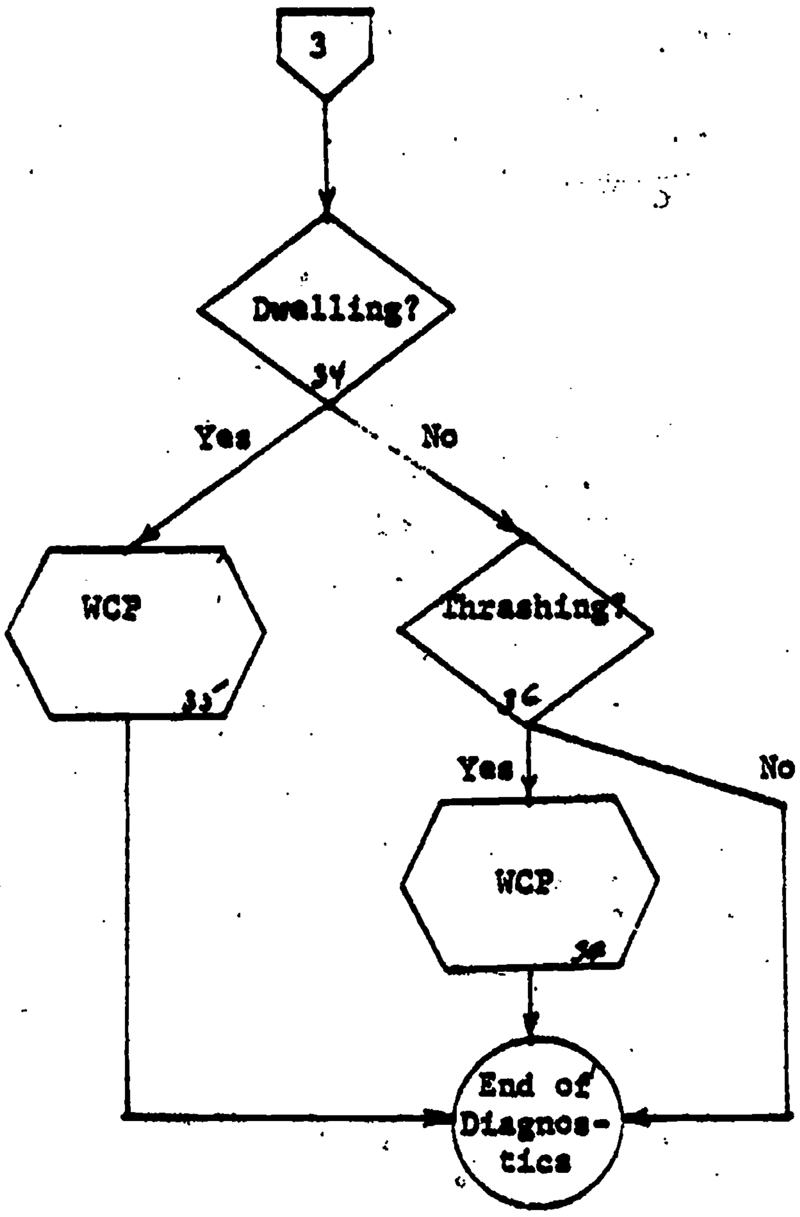

Figure 2.2c. Detailed diagnostic decision tree, part c.

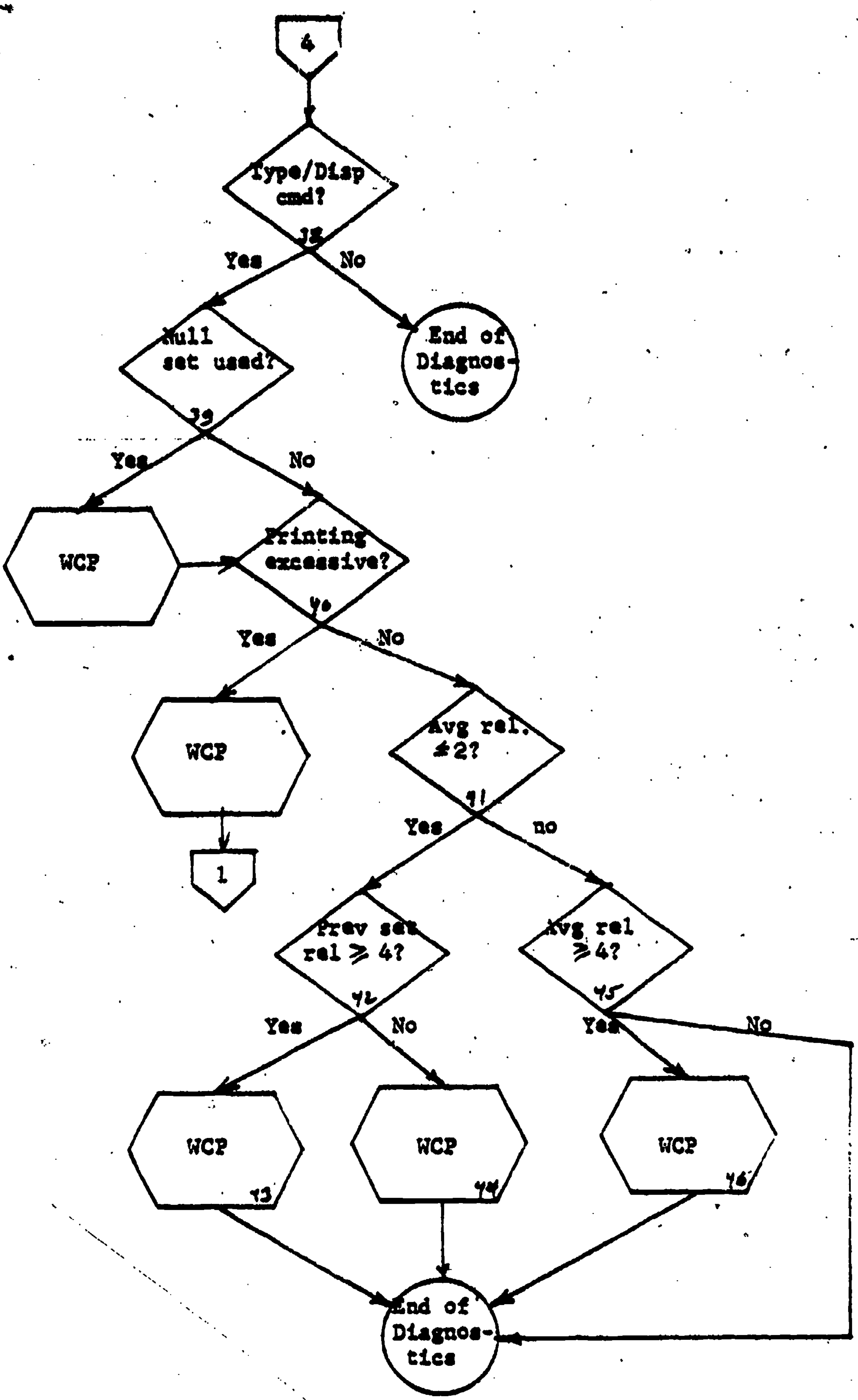

Figure 2.2d. Detailed diagnostic decision tree, part d.

check for low relevance of retrieved records (24). Again, if found, WCP is informed. The next test is for a null set generated (26). If the last command did generate a null set and it was a SELECT command, then the program looks up the term in an IIDA-maintained table which tells whether that term was ever seen in an EXPAND display (28). The WCP is then informed (29). If the command were not a SELECT, then it was a LIMIT/COMBINE (nothing else generates sets) and WCP is so informed (30). If a null set were not generated, then a test is made to see if the command were of the COMBINE/LIMIT class (31) and, if so, whether a null set was used in its argument (32). If so, the user is informed (by the parser) (33), and we continue checking for COMBINE/LIMIT faults in part c, at <u>3</u>.

In part c, dwelling is checked for (34) and thrashing (36). If either is found, the WCP is informed. In any case, the end of the diagnostic tree is reached.

Part d concerns tests on TYPE/DISPLAY commands (38). If the command is not of that class, the tree is terminated. It it is TYPE/DISPLAY, a check is made (39) for null sets used and then (40) for excessive printing. Following this, relevance is checked. If average relevance for the subset of records printed with this command <=2, a diagnostic message will be sent. If there had been a previous set that had relevance >=4, one message is sent (43). If not, a different message is sent (44). If the most recent subset had an average relevance >=4 (45), a message is sent (46) suggesting that the search may be completed. In all cases, the tree is terminated.

2.2.2. Procedural diagnostic rules. The rules shown in Table 2 are for diagnosis of procedure only. Tests for syntactic validity and the use of null sets are made by the IIDA command Parser program (2). Additional details will be forthcoming in a future report.

The rules are grouped into 9 categories:

1. String length
2. Command repetition
3. Null set generation
4. Unused sets
5. Thrashing
6. Dwelling
7. Record relevance
8. Print control
9. User response interval

The rules are listed in Table 2 and are numbered according to these categories, and then sequentially within category. Numbers in parentheses refer to a sequential listing of rules as actually stored within the computer. Each rule consists of a natural language statement of the rule followed by the PL/I programming language statement of the same rule.

1. String length

1.1 (3)
if the EXPAND command occurs 8 times consecutively(cons_now_type) then a message is signalled (te_send(3)).

1.2 (4)
if the SELECT command occurs 8 times consecutively(cons_now_type) then a message is signalled (te_send(4)).

1.3 (5)
if type 2 commands(EXPAND/SELECT) occur 8 times consecutively (cons_now_type) then a message is signalled (te_send(5)).

1.4 (6)
if the COMBINE command occurs 5 times consecutively (string_length(s_last,st_last)) then a message is signalled (te_send(6)).

1.5 (7)
if the TYPE command occurs 5 times consecutively (string_length(s_last,st_last)) then a message is signalled (te_send(7)).

1.6 (8)
if any string of commands is 10 (string_length(s_last,st_last)) and the major command type (c_type_maj(c_last)) is > 4 then a message is signalled (te_send(8)).

1.7 (39)
if the length of any string of commands equals 20 (string_length(s_last,st_last)) then a message is signalled (te_send(39)) and the user is lossed off.

2. Command repetition

2.1 (2)
if the number of essential repetitions thus far in the search (rep_knt_search(2)) is 1 or less and at least one such repetition occurs during the current cycle(rep_knt_cycle(2)) then a message is signalled (te_send(2)) which gives the repeating commands and help is offered.

2.4 (22)
if the number of exact repetitions thus far(rep_knt_search(1)) is 1 or more and at least one such repetition occurs during this cycle then a message is signalled (te_send(22)).

Table 2. Procedural Diagnostic Rules, Part a. Column 1 is the rule number classified by purposes; Column 2 is the number in order of computer implementation, only for programmers purposes.

2.5 (23)
if the total command repetitions thus far(rep_total) is 6 or more then a message is signalled (te_send(23)).

2.7 (38)
if the number of dual repetitions(exact and essential simultaneously) thus far is greater than 1 (rep_knt_search(3)) and at least one such repetition occurs during this cycle (rep_knt_cycle(3)) then a message is signalled (te_send(38)) which gives the repeating commands and help is offered.

2.8 (40)
if the number of consecutive command repetitions(con_rep_knt) is 10 then a message is signalled (te_send(40)) and the user is lossed off.

3. Null set generation

3.1 (9)
if two consecutive nulls occur from the use of the SELECT comman(zero_knt_cons(1)) then a message is signalled (te_send(9)). RULE FUNCTIONS ONLY ONCE IN EACH CYCLE

3.2 (10)
if two consecutive nulls occur from the use of the COMBINE command(zero_knt_cons(2)) then a message is signalled (te_send(10)). RULE FUNCTIONS ONLY ONCE IN EACH CYCLE

3.3 (11)
if two consecutive nulls occur(zero_knt_cons(3)) then a message is signalled (te_send(11)). RULE FUNCTIONS ONLY ONCE IN EACH CYCLE

3.4 (12)
if three nulls occur in a cycle from the use of the SELECT command(zero_knt_cycle(1,s_last)) then a message is signalled (te_send(12)), the arguments given for the nulls, those arguments which have occurred within the scope of a previous EXPAND table are listed separately.

3.5 (13)
if three nulls occur in a cycle from the use of the COMBINE command(zero_knt_cycle(2,s_last)) then a message is signalled (te_send(13)) which gives the arguments of the nulls.

3.6 (14)
if three nulls occur in a cycle(zero_knt_cycle(3,s_last)) then a message is signalled (te_send(14)) which gives the arguments of the nulls.

Table 2. Procedural Diagnostic Rules, Part b. Column 1 is the rule number classified by purposes; Column 2 is the number in order of computer implementation, only for programmers purposes.

3.7 (44)
if the argument of a SELECT command contains more than 3 words (space_knt) and a null set is generated (s_size(s_last)) then a message is signalled (te_send(44)).

3.8 (45)
if the argument of a SELECT command contains more than 3 words (space_knt), there are fewer than 5 SELECT and EXPAND commands in the entire search (d_last), and there are less than 4 records in the set (s_size(s_last)) then a message is signalled (te_send(45)).

3.10 (36)
if the argument of a SELECT command produces a null set and this argument falls within the bounds of a previous EXPAND table (s_flag(c_last) = "1"b) then a message is signalled (te_send(36)) which gives the argument of the previous EXPAND command.

4. Unused sets

4.1 (19)
if the number of non_used sets created during the last cycle is 3 (no_ref_cycle) then a message is signalled (te_send(19)) which gives the arguments of the non-used sets.

5. Thrashing

5.1 (24)
if for a least 4 consecutive COMBINE commands using the AND/OR operators, the similarity index(sim_avg) is less than .25 then a message is signalled (te_send(24)) which gives the arguments of the COMBINE commands.

6. Dwelling

6.1 (25)
if for at least 6 consecutive COMBINE commands using the AND/OR operators, the similarity index(sim_avg) is greater than .75 and the set size is converging toward the goal (set_size_disp) then a message is signalled (te_send(25)).

6.2 (31)
if for at least 4 consecutive COMBINE commands using the AND/OR operators the similarity index is greater than .75(sim_avg) and the set size is diversing from the goal(set_size_disp) then a message is signalled (te_send(31)).

Table 2. Procedural Diagnostic Rules, Part c. Column 1 is the rule number classified by purposes; Column 2 is the number in order of computer implementation, only for programmers purposes.

6.3 (32)
if for at least 5 consecutive COMBINE commands using the AND/OR operators the similarity index is greater than .75(sim_avg) and the set size is static relative to the goal(set_size_disp) then a message is signalled (te_send(32)).

6.4 (33)
if for at least 5 consecutive COMBINE commands using the AND/OR operators the similarity index is greater than .75(sim_avg) and the set size relationship to the goal cannot be determined (set_size_disp) then a message is signalled (te_send(33)).

7. Record relevance

7.1 (26)
if the average relevance of documents viewed at this command (view_avs(r_last)) is less than 3 (t3) and more than 2 records are seen then a message is signalled (te_send(26)).

7.2 (27)
if for at least 4 consecutive TYPE commands the average relevance (fudge_rel - 25 > view_avg(r_last)) is decreasing then a message is signalled (te_send(27)).

7.3 (41)
if for at least 4 successive TYPE commands the average relevance of records viewed is increasing (fudge_rel + .25 < view_avg(r_last)) then a message is signalled (te_send(41)).

7.4 (42)
if for at least 4 successive TYPE commands the average relevance of records viewed is static and no group had a relevance rating greater than or equal to 4 (fudge_rel_hi < 4) then a message is signalled (te_send(42)).

7.5 (43)
if for at least 4 successive TYPE commands the average relevance of records viewed is static and at least one group had a relevance rating greater than or equal to 4 (fudge_rel_hi >= 4) then a message is signalled (te_send(43)) which gives the highly relevant group.

7.6 (28)
if maximum relevance of a group in the previous cycle (max_rel(s_last - 1)) is greater than the maximum relevance of the last cycle(max_rel(s_last)) then a message is signalled (te_send(28)).

Table 2. Procedural Diagnostic Rules, Part C Column 1 is the rule number classified by purposes; Column 2 is the number in order of computer implementation, only for programmers purposes.

7.7 (29)
if the average relevance at this command(view_avg(r_last)) is greater than 4 (t4) and more than 2 records are seen then a message is signalled (te_send(29)).

8. Print control

8.1 (18)
if the record format of the current TYPE command is 1 (record_format(r_last) then a message is signalled (te_send(18)).

8.2 (35)
if the number of records requested in a TYPE command exceeds the threshold for the format requested, according to the table below, then a message is signalled (te_send(35)) and the command is cancelled.

FORMAT (record_format(r_last+1)	THRESHOLD (fmt_recs(record_format(r_last+1)))
1	12
2	4
3	8
4	8
5	4
6	12
7	4
8	4

9. User response interval

9.1 (34)
if the average time taken to enter commands(time_avg) is greater than 30 seconds then a message is signalled (te_send(34)).

Table 2. Procedural Diagnostic Rules, Part e. Column 1 is the rule number classified by purposes; Column 2 is the number in order of computer implementation, only for programmers purposes.

3. Adapting Rules

One of the objectives of the IIDA communication with users is to avoid giving them a feeling of harrasment. This could happen if the same message were repeated too often, even if the same fault is often repeated. The major function originally assigned to the Warning Control Program was to defer messages if a fault were repeated soon after a previous instance of the same fault. A more elaborate version of message deferment has been developed, called rule adaptation. Rule adaptation provides for automatic message deferment (i.e. postponement of sending a message when a rule is re-violated too soon), for making more stringent the thresholds used to decide when to send a message if the rule is violated often, and to provide a means whereby the user can request relief if the program becomes too stringent.

3.1 Nature of Rule Adaptation

For the purposes of adaptation, rules are categorized as follows:

1. Essential command repetition
2. String length
3. Null set generation
4. Non used sets
5. Exact command repetition
6. Dual repetition (both essential and exact)

Only rules in these categories are adaptable. The general idea of adaptation is that some rules, when broken, may not be commented upon by IIDA until they have been violated several times. Then, the threshold is incremented, so that several more violations are necessary before another message will be sent concerning that fault. If the threshold has to be incremented too often, it is reset to a more stringent level. If this more stringent level becomes too restrictive to the user, he can request SLACK which returns the threshold to its original value.

Here is an example of the operation of adaptation logic:

The user enters 8 consecutive SELECT commands. Use of a SELECT is not in any sense a violation, but eight in a row is, hence a threshold is set at 8 for length of a string of this class.

When the threshold is reached, the WCP is informed. This causes a message to be sent to the user, a simple statement of the number of consecutive commands that occurred in the string.

The threshold will now be incremented, by 3, to 11. If the user continues to enter SELECT commands, no new message will be sent until he reaches the 11th command in the string. At that point, a message is sent and it is enhanced to show that the same fault has occurred recently. The threshold is again incremented, by 3, to 14.

If the user continues to enter SELECT commands and reaches command number 14, the WCP again sends him an enhanced message. Now, because the threshold has been incremented twice, the increment is reset, to the more stringent value of 2. Thus, the next threshold value is 16, two more than previously. While 16 consecutive SELECTS may appear far fetched, it does happen.

It might well be that the user knows he is violating an IIDA rule, but feels that there is justification in being so. If he notices that IIDA is getting more stringent and wants this stopped, he enters the IIDA command SLACK. This has the effect of resetting the threshold and its increment to its original values, i.e., the threshold at 8, the increment at 3.

The specific numbers used here, initial threshold of 8, increment of 3, etc., are all arbitrary. They were based upon designers' experiences, but should be ultimately determined by experimentation.

3.2 Assignment of Rules to Classes.

Refer to the listing of rules in Table 2, and the list of rule categories in Section 3.1. Rules are assigned to classes as follows:

Class	Rule nos.
1. (Essential repetition)	1,2
2. (String length)	3-8
3. (Null sets)	9-17
4. (Non-used sets)	19; 20 (18 not adaptable)
5. (Exact command repetition)	21-23
6. (Dual repetition)	37,38 (24-36 not adaptable)
	39-50 not adaptable

3.3 Evaluating Rule Adaptation

The Warning Control Program was created because of the perceived need for flexibility in administering the flow of messages indicating rule violation. A few trials convinced us we needed even more flexibility. For rules whose violations were not critical, we wanted to get more stringent gradually and to allow the user in effect to overrule the machine if he wished. Thus, we designed the variable increment for rule thresholds, by which we would get more stringent as more violations occurred, but also allow for user-initiated slack.

Now, we must face the problem of testing to determine the appropriate level of setting for the various thresholds and increments. Also, it would be desirable to test to determine whether this form of adaptive behavior on the part of IIDA does, indeed, induce the behavioral changes in users that was desired. Our conclusion is that the number of tests required would be so large as to be a near physical impossibility. We would have to run enough tests that all error types occurred often enough to provide a meaningful sample, and then watch how various settings affected the behavior of the errant users. Our estimate is that this would require thousands of searches to be performed through IIDA, with a fairly homogeneous group of users. The cost of this is simply beyond the financial capability of the project.

4. References

1. Individualized Instruction for Data Access, Quarterly Report No. 3, Philadelphia, Drexel University School of Library and Information Science, December, 1978.

2. Individualized Instruction for Data Access, Quarterly Report No. 2, Philadelphia, Drexel University School of Library and Information Science, September, 1978.

3. Individualized Instruction for Data Access, Quarterly report No. 1, Philadelphia, Drexel University School of Library and Information Science, June, 1978.

4. Brown, John Seely and Richard R. Burton, "Diagnostic Models for Procedural Bugs in Basic Mathematical Skills," Cognitive Science 2, 1978, 155-92.

5. Brown, John Seely, R. Burton, and A. Bell, "SOPHIE: A Step Toward Creating a Reactive Learning Environment," International Journal of Man-Machine Studies, 7, 1975, 675-96.

6. Penniman, W. David, "A Stochastic Process Analysis of On-Line User Behavior," P. Annual Meeting of the American Society for Information Science, Vol. 12, Washington, ASIS, 1975.

7. Fenichel, Carol Hansen, Online Information Retrieval: Identification of Measures that Discriminate Among Users with Different Levels and Types of Experience. PhD dissertation, Philadelphia, School of Library and Information Science, Drexel University, 1979, pp. 89-102.