COMPUTER-ASSISTED ITEM AND TEST PRE-ANALYSIS: A NEW DIRECTION IN QUALITATIVE METHODS

by

Clay Alan Sales

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF ARTS IN EDUCATION

in

Curriculum and Instruction

APPROVED:

John Burton, Chairman

Larry Weber

Patricia Bartos Foutz

August, 1987

Blacksburg, Virginia

COMPUTER-ASSISTED ITEM AND TEST PRE-ANALYSIS: A NEW DIRECTION IN QUALITATIVE METHODS

by

Clay Alan Sales
Committee Chairman: Dr. John Burton
Division of Curriculum and Instruction

(ABSTRACT)

To date, the major emphasis in test and item evaluation has been directed toward statistical post-measures which rely heavily on data gathered from the administration of the instrument. These primarily summative techniques are limited, however, in that they are incapable of providing information about and item/test before it has been sent for field trials. This research presents a new direction in test and item analysis which, using test/item writing heuristics, provides a previously unavailable technology for instrument pre-analysis. The new field of "qualitative item and test pre-analysis" is proposed and described. The implications to the field are discussed in addition to specific suggestions for the use of this new technology.

The design and creation of a base-case item and test pre-analysis expert system (ITAX) is also detailed, including the heuristics incorporated, implementation methodologies and limitations. The heuristics incorporated into the system include the detection of: two varieties of grammatical cues, negation/multiple negation, repetition of phrases within an options list, presence of too few options, inconsistent length of distractors, use of all- and none-of-the-above, repetition of significant words from the stem to the options, randomness of multiple choice answer placement, balance of true/false items and length of true/false items. A comprehensive reference to the system is also provided.

Acknowledgements

...At the risk of making this sound like an "Academy Awards" acceptance speech, I find that there are many people to whom I owe a deep debt of gratitude for their help, both personal and professional, during the course of my work here at Virginia Tech. Therefore, like some tired old long-winded actor, I want to thank the following people:

- My long-suffering parents, sister, and brother – who have stood behind me from day-one, (and who will no doubt breath a sigh of relief when I finally leave Blacksburg),
- The gang from the 1987 TAUT cast of "HAIR", who brought me back to my humanity and taught me joy,
- Sandy, Gay, Ron, Adam and the rest of the 102 crowd for their help (and forbearance) with the CS Department computer systems,
- Dr. Foutz of the Statistical Consulting Center for his user-friendly, natural-language explanations of the statistics I used in ITAX,
- Dr. Roach (my boss) for both is tolerance of my variable working hours and his direction in the expert system background discussion,
- Dr. Larry Weber for providing me with a much needed crash-course in test construction theory and his enthusiasm for my work,
- for making me do it the "right and elegant" way, and for seeing me through seven years of Tech,
- and, finally, Dr. John Burton (major advisor extraordinare), who has always been willing to put up with me and my sometimes-on, sometimes-off graduate academic career, direct me though the twisted bureaucratic labyrinth that is the Tech administrative process, and generally stick with me to the bitter end.

Enough, already. I'm going home.

Say goodnight, Gracie...

Contents

List of Figures

List of Tables

Chapter 1

Introduction & Background Literature

1.1 Introduction to the Study

The classroom test has long held an important, yet controversial, place in education. From the moment when a child first enters the classroom, he or she is subjected to a torrent of educational measurement intended to evaluate not only student progress, aptitudes and achievement levels, but also teaching effectiveness. Paradoxically, the majority of classroom achievement tests are constructed by teachers who, in many cases, are either ill-trained in test writing, simply do not want to devote the time necessary to the task, or have little test writing facility. This inevitably leads to misguided perceptions of either student or teacher performance based on invalid test results.

Up to now, much of the energy devoted to researching methods for evaluating the effectiveness of tests and items has been directed primarily toward statistical analyses following the (often repeated) administration of a particular instrument. While benefiting from the robustness which such numerical analyses enjoy, these methods all suffer a common drawback: they cannot offer any measurement of effectiveness until after the fact; that is, they are all summative, rather than formative in nature. The test constructor is left, therefore, with no way of evaluating the instrument until after it has been put into use – at least on a trial basis. In addition, such analyses are beyond the training or time restraints of most classroom teachers. What is needed is a methodology, usable by classroom teachers, by which a test can be examined before its administration.

In research described here, a new direction in test and item analysis is proposed. This technique, called "qualitative pre-analysis", uses the great amount of knowledge about test and item construction which already exists to form the basis of a test pre-analysis expert. An expert system of this type can analyze items and tests before they are administered, thus providing the formative test construction tool which has,

until now, not been available.

The purpose of this document is to describe the specification and development of a new approach to the problem of test analysis whereby the items composing an instrument are examined directly for flaws in construction. The automation (computerization) of such a process provides not only for computer-based analysis, but also demonstrates the ultimate feasibility of the construction of expert systems with test construction theory as their knowledge base. By bringing together aspects of such varied fields as education, artificial intelligence, and linguistics, we indicate a new direction for the study of tests and item construction.

1.2 Overview of Testing

1.2.1 What is a test?

The concept of a "test" can mean different things to different people, depending upon their goals and objectives. To the counselor or psychologist, a test is a means of measuring aptitudes and interests in order to help the student make sound decisions. Alternatively, a test may provide information on program effectiveness to the school administrator. Finally, there is the test – administered by a classroom teacher – as a means of assessing student achievement (Ahmann, 1981). The latter, most common conception of tests will be the primary focus of this discussion.

Good (1973), in his book entitled "Dictionary of Education", provides the following definition for the term "test":

> (1) a group of questions or tasks to which a student is to respond, the purpose being to produce a quantitative representation of the pupil trait that it is designed to measure...

In plainer terms, a *test* can be described as a group of questions posed to a student in order to provide a *measurement* of that student's level of knowledge on a subject. This process results in the assigning of some numerical value (quantification) to that level, which can then be used to *evaluate* the student's achievement. This evaluation typically takes the form of a comparison against the measurements of other students (as in the case of norm referenced testing) or against some achievement standard (for criterion referenced testing).

To the classroom teacher, tests serve two purposes. During instruction, *formative* testing gives the teacher an indication of how well objectives are being attained. The results from such tests affect the future course of instruction: reviewing, accelerating or slowing the teaching process in order to best serve the needs of the student. Upon completion of the teaching process, tests serve the *summative* function of evaluating students' assimilation of the material. The formative test might be likened to the "weekly quiz" whereas the summative test is more closely associated with a "final examination".

That tests are an important factor in education to both the student and teacher (although for different reasons) is well documented. One might therefore infer that tests should exhibit some quality of "goodness". This quality may take many forms, as will next be explored.

1.2.2 What describes a "good" test?

Two "goodness" attributes of tests are *reliability* and *validity*. These two qualities, although often incorrectly used in similar contexts (Ahmann, 1981), measure related but separate aspects of a test.

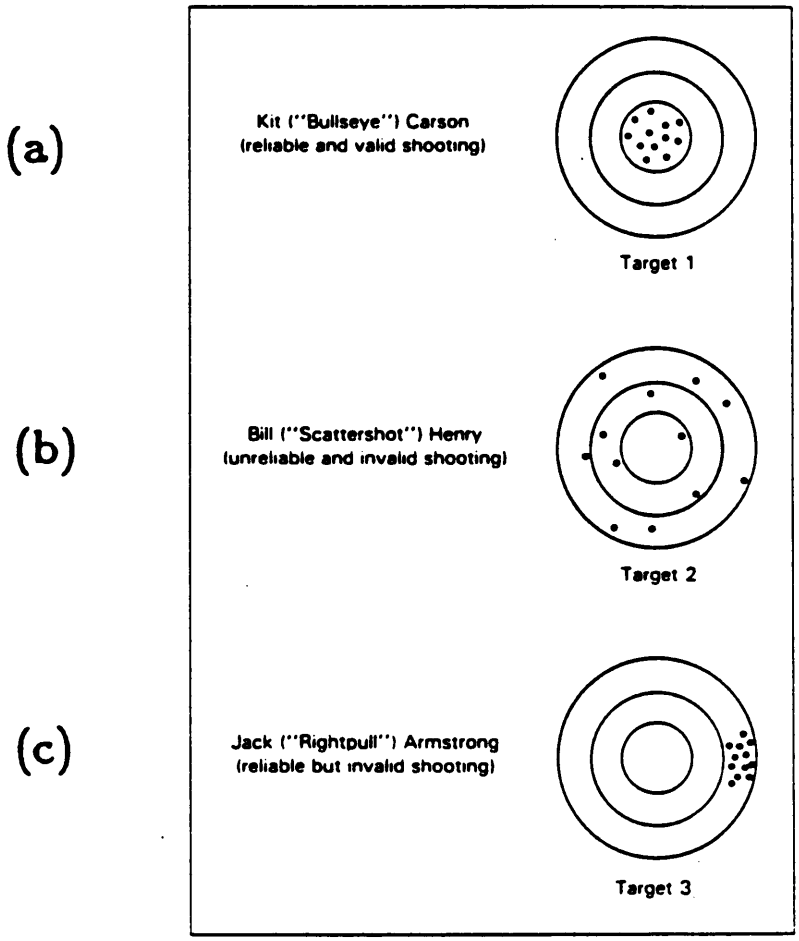
Reliability

Reliability is a measure of a test's consistency; that is, if a test is administered to populations which are stable in the attribute being measured (or to the same person twice), the results should be essentially the same. In Figure 1.1, (a) and (c) exhibit reliability because the shots were clustered within a small area, regardless of their location (value). Reliability, then, is a reference not to the actual quantitative values supplied by the test, but to the predictability with which those values occur for a given population. This predictability says nothing about the result itself as it relates to the quality being tested (*validity*, to be discussed later). On the contrary, it merely allows the tester to infer, with some degree of confidence, that the test produces a consistent result.

Several factors can influence the consistency (or reliability) of a test. Wiersma (1985), Ebel and Frisbie (1986), and Gronlund (1985) all point out several primary sources of unreliability. These seem to fall into three broad categories: student-related, test-related, and scoring-related. Table 1.1 provides a summary by category. Recognizing that no test can be completely reliable, (due mainly to the very nature of those items appearing in Table 1.1), the task of the test constructor becomes the maximization of reliability. For the classroom teacher, this can be accomplished through something as simple as suggesting to the student that he "get a good night's rest" to something more involved like the multiple testing (test/retest) method and other more statistically oriented means. [See Gronlund (1985) for a discussion of reliability coefficients] In whatever form it might take, the maximization of reliability should be a goal for the test constructor for, as Gronlund (1985) observes "...reliability ...provides the consistency that makes validity possible."

Validity

Once reliability has been established, it becomes logical to talk about validity. The *validity* of a test may be described as "...the extent to which it measures what it is intended to measure" (Wiersma & Jurs, 1985). In other words, validity tells the test user whether the test is doing the job for which it was devised. A test cannot possibly



[From Gronlund(1985)]

Figure 1.1: Reliability vs. Validity

Table 1.1: Summary of Sources of Unreliability

- Student Related
  - -fatigue
  - -understanding of test parameters
  - -motivation
  - -external conditions

- Test Related
  - -length
  - -confusion about response method (answer sheets)
  - -test difficulty

- Scoring Related
  - -objectivity of scoring
  - -correctness of scoring

be "doing its job" if it yields unpredictable scores, and therefore the connection with reliability can readily be seen. Again referring to Figure 1.1, (a) depicts validity (and, of course, reliability) since all of the trials produced approximately the same result, and that result was not only consistent (reliable) but also "correct" – in this case a bulls eye.

Evidence for the validity of a test is commonly gathered in three areas, one which is the result of a logical inspection of the test itself (*content* validity), and two which are statistically based (*criterion related* and *construct* validity). Since validity itself is not an absolute quality, but rather a degree measure, the process of validating a test includes the gathering of evidence for all three components. The strongest case is made in the situation where strong evidence exists in all three (Gronlund, 1985).

To gather evidence for content validity, the test is examined in terms of its relationship to the teaching objectives for which it was written. Since, in most situations, it is impractical to test every piece of knowledge which was to have been learned, the test must be composed such that a *representative sample* of the totality is tested (Wiersma & Jurs, 1985). Since this sampling is a logical (subjective), rather than statistical process, making any concrete declarations about representation is difficult. For this reason, content validity is often assured during the test construction process, rather than being assessed after-the-fact. In the teacher-constructed test, this often occurs automatically as the teacher reviews lecture and/or objectives notes when planning a test (Wiersma & Jurs, 1985). There is, however, a simple planning method which, when used, can increase the content validity of a test.

The development of a *table of specifications* entails reviewing each learning objective and assigning the relative weight it should have on the test. First, the total number of questions to be written in decided upon, and then each objective is examined in turn and assigned some portion of that total. In Figure 1.2, the objectives are listed on the left side of the table, followed by the number of each type of question which will be included on the test. In this example, the type of questions follow Bloom's taxonomy (although this is by no means the only method which can be used). By using this technique, every objective is given consideration and the relative weighing of topical areas can be defined based upon their importance. The development of a table of specifications not only helps the test constructor write content valid tests, but also provides an effective aid in test planning (Wiersma & Jurs, 1985).

Criterion-related validity refers to the relationship between the test's score and some other factor, or criterion. This relationship takes two forms: *predictive* and *concurrent*. Predictive validity concerns the use of a test to predict the subject's future performance in some related area. The primary examples of this type of usage are the standardized tests used for college admissions. By examining these scores, a college admissions officer is, in effect, saying that the test score is a predictor of the student's future achievement in college. There are several statistical methods for generating the correlation between a test and the fact it is to predict [see Gronlund (1985) for discussion and examples].

Where predictive validity seeks to project future performance, *concurrent* validity has as its goal the association of the test score with some other current ability level. In many such cases, a test is given to predict the subject's possible performance on a different instrument. A common instance here is the replacement of a long test with a relatively shorter one. If the shorter test can be shown to be functionally equivalent to the more lengthy version, then the tester could infer that the shortened test could be used in the place of the longer form, thus saving testing time while maintaining validity (Wiersma & Jurs, 1985). To determine the concurrence, both tests are administered to a group with little or no intervening time period. A correlation is then drawn through statistical methods [again, see Gronlund, (1985) for a detailed discussion].

The final area of criterion-related validity is *construct* validity. A construct is a psychological term which refers to a "...conceptualization about an aspect of human behavior that cannot be measured or observed directly." (Ebel & Frisbie, 1986) Examples of constructs are *intelligence*, *honesty*, *mathematical aptitude* and *reading comprehension*. Construct validity, then, is the degree to which the test can be said to measure one or more of these constructs. The correlation drawn here is usually between the test in question and other tests which claim to measure the same construct(s). Unfortunately, this basis is often shaky at best, owing to the inherent problem that it can be very difficult to discriminate between closely allied constructs, especially if they manifest themselves in similar ways – a case in point: understanding of scientific principles and general intelligence (Ebel & Frisbie, 1986). This is compounded with the difficulty of actually making the inference that a body of tests does describe a construct's manifestation. In practice, psychologists must rely on suppositions which are made based on theory (Ebel & Frisbie, 1986). [Once again, see Gronlund (1985) for a presentation of several methods for determining construct validity.]

There are numerous factors, both internal and external to the test, which can have an effect on a test's degree of validity in general. Gronlund (1986) points out several factors within the test itself which can have an adverse effect on validity. Among these are:

- unclear directions
- inappropriate reading level
- inappropriate difficulty level
- poorly constructed items
- ambiguity
- inappropriate items
- test brevity

- improper arrangement of items
- identifiable patterns of answers

Among those factors external to the test are the following (Gronlund, 1985)(Wiersma & Jurs, 1985):

- inappropriate use of a test
- inconsistent scoring
- homogeneous groups of subjects
- cheating
- time limitations
- environmental considerations

All of these introduce "noise" into the measure, and will be explored further in succeeding sections of this chapter.

So, going back to the question posed at the beginning of this section, "what is a good test", the answer must be that a "good" test is one which is both reliable and valid. Tied up in these concepts are such related topics as test item writing, administration, and test construction procedures.

1.3 Test Construction

Constructing a reliable and valid test presents several problems. First, the test constructor must make the decisions about what should be tested. Through careful examination of the learning objectives under consideration, the teacher must assign relative weights and importances to each. Because of the inherent subjectiveness of these decisions (Wiersma & Jurs, 1985), there is no clear cut method of making choices which will yield completely valid results. Although several methods (i.e. test specifications) can be employed to make the process more methodical and less prone to error, the fact remains that the deciding factor in each test's validity lies ultimately with the expertise of that test's writer. The validity of a test can, however, be enhanced through the careful construction of the *items* (individual questions) from which it is composed (Wiersma & Jurs, 1985). In this section, the various types of commonly used items will be discussed, pointing out their inherent strengths and weaknesses and the factors and procedures which will maximize their validity. Since the great bulk of all tests are constructed by classroom teachers (Ahmann & Glock, 1981), these so called "teacher-made tests" will serve as the focus for this discussion. The five common item formats which will be examined are *multiple choice*, *matching*, *true/false*, *supply*, and *essay*.

1.3.1 The *Multiple Choice* Item

Summary:

The multiple choice item is made from two components: the *stem*, which is the statement of the problem, and the *options*. The options consist of the correct answer to the stem and some number of incorrect answers or *distractors* (also called *decoys* or *foils*). The task of the examinee is to select, from among the offered options, the correct (or most correct) answer to the stem.

This type of item is perhaps the most commonly used test question format (Ebel & Frisbie, 1986), and due in part to this popularity, quite a bit has been said both in favor and against the form. While proponents cite its flexibility, objectivity and directness, critics claim that multiple choice items are often superficial and ambiguous, testing surface facts rather than understanding. In a properly constructed question, however, these difficulties can be overcome. Contrary to popular belief, multiple choice questions can be very difficult to construct (Ahmann & Glock, 1981), especially in those cases where a deeper understanding of a concept is to be tested. The effort, though, results in a question which not only tells the tester whether the student knows the material, but can also diagnose problems by the choice of incorrect response.

The phrasing of the question typically takes one of two forms: the *direct question* and *incomplete statement*. Figure 1.3 presents the same question phrased in both ways. The direct question has the virtue of being able to present a more completely stated problem, where the incomplete statement is often more concise. Generally, the former is used, only going to the latter when clarity demands (Gronlund, 1985). The other primary distinction in items of this type is the use of either the *best*-answer or *correct*-answer format. For correct-answer questions, one and only one of the options is correct, whereas in the best answer style the options are closely related. Often it is true that more than one could be considered correct. In the second case, the task of the testee is to choose that option which best answers the question. This style tends to be more difficult both to construct and answer, although it is more successful in discerning finer points of understanding and application (Gronlund, 1985).

Numerous authors offer suggestions for the construction of multiple choice items and avoidance of pitfalls. A selection of these appears next.

Preparing Stems:

Be careful of grammatical clues (Wiersma & Jurs, 1985). In Figure 1.4, the first question inadvertently indicates that the correct answer begins with a vowel (due to the "an" at the end of the sentence) and the second question points to

	Learning Outcome				
Content	Knowledge	Comprehension	Application	Analysis	Total
City development	2	3	2	2	9
City characteristics	2	1	4	0	7
City governance	1	1	2	0	4
City purpose	1	3	4	2	10
Total	6	8	12	4	30

[From Weirsma & Jurs (1985)]

Figure 1.2: Table of Specifications

(a)

Direct-question form:
In which one of the following cities is the capital of California?
A Los Angeles
Ⓑ Sacramento
C San Diego
D San Francisco

(b)

Incomplete-statement form:
The capital of California is in
A Los Angeles.
Ⓑ Sacramento.
C San Diego.
D San Francisco.

[From Gronlund(1985)]

Figure 1.3: Direct-question vs. Incomplete-statement

1. A prehistoric birdlike animal is called an _____.
   a. triceratops.
   b. archaeopteryx.
   c. stegasaurus.
   d. diplodocus.

2. Fruits that were once considered poisonous to humans are _____.
   a. tomatoes.
   b. pear.

[From Weirsma & Jurs (1985)]

Figure 1.4: Grammatical Cues

a plural answer. These types of errors do indeed test something, but grammar was perhaps not the intended purpose.

In general, state the stem in the positive (Gronlund, 1985). By using negative modifiers, the question could end up confusing the student as to the terms of the problem, especially if multiple negatives are used. This creates a "logical tangle" which the student must unravel before addressing the true purpose of the problem. If a negative must be used, emphasize it by capitalization, bold facing, etc.

The stem should pose a complete problem (Gronlund, 1985). In figure 1.5 a, the stem presents neither a direct question nor an incomplete statement. Rephrased in b, the same material is covered, this time giving the student more direction as to the focus of the problem.

Don't repeat key words in both the stem and correct answer (Ebel & Frisbie, 1986). In Figure 1.6, the word "vertical" in both the stem and option (a) clues the examine.

Preparing Options:

Use an appropriate number of distractors (Gronlund, 1985). Since the possibility of a random correct guess is reduced with a larger number of possible answers, it is recommended that at least 4 options be used (offering a 25% chance).

Make options of a similar length (Wiersma & Jurs, 1985). The longer option is typically the correct one because correct answers often require modifiers.

Make all of the distractors plausible (Gronlund, 1985). To include an impossible distractor is effectively the same as not including it at all, since the student will most likely eliminate it from consideration from the start.

Use "all of the above" with restraint (Gronlund, 1985). Because any savvy test-taker knows that this option can be eliminated if just one of the options can be eliminated, its effectiveness is limited.

The correct answer should be positioned randomly in the list (Wiersma & Jurs, 1985). By establishing some pattern, the test cues the examine as to possible guessing patterns. This is exemplified by the fact that most inexperienced test constructors try to "hide" the correct option in the middle of a list, rather than at the first or last position. Again, the savvy test taker will know this

and always guess to the center of a list. By randomly placing the correct answer, the percentage chance of guessing a question is preserved at the level specified by the number of options.

When giving directions for answering multiple choice questions, indicate clearly the method of response and whether a correct answer or best answer is desired (Wiersma & Jurs, 1985).

1.3.2 The *Matching* Item

Summary:

The matching item is made from a list of *premises*, a list of *responses*, and *matching instructions*. The task of the examine is to match (associate) each of the premises with an element from the list of responses.

Traditionally, the two lists are presented as columns, with the premises on the left and the responses on the right (Gronlund, 1985). Matching questions are most appropriately used for the testing of long vocabulary lists and other large, closely related sets of information. The knowledge tested is generally factual in nature, such as associating authors with books, events with dates, etc. In many ways, the matching item is very much like an extended multiple choice question. By combining closely related questions, the answer for each question serves as a distractor for the others. In addition to conventional association tasks, the matching item can be used to test higher levels of learning. As presented in Figure 1.7, the matching item can be used to make repeated tests of the same concepts, using a very small number of responses and lengthening the number of premises. Another interesting usage of the matching format is the compound matching item (Figure 1.8). In this case, what might have been presented in two separate matching problems has been condensed into a single question. Admittedly, this technique may have limited applications, but it does illustrate nicely that, in test construction, a little original thought can go a long way.

Although less attention has been paid to the matching item, there remain several relevant suggestions for their construction:

Make the list of responses longer than the list of premises (Ahmann, 1981). By providing more "answers" than there are questions, the test writer can avoid the situation where the testee uses "elimination" to arrive at some of the answers. As was the case in the multiple choice item, the larger the number of possible distractors, the less the chance of guessing providing a correct response.

Poor: South America

A is a flat, arid country.
B imports coffee from the United States.
C has a larger population than the United States.
(D) was settled mainly by colonists from Spain.

Better: Most of South America was settled by colonists from

A England.
B France.
C Holland.
(D) Spain.

[From Gronlund(1985)]

Figure 1.5: Statement of Complete Problem in Stem

When used in conjunction with the T-square, the left vertical edge of a triangle is used to draw

***a. vertical lines.**
b. slant lines.
c. horizontal lines.
d. inclined lines.

[From Ebel & Frisbie (1986)]

Figure 1.6: Repetition of Significant Words

Classification Variety

Directions: Each of the following statements is a complete sentence. Determine whether the sentence is a simple, complex, compound, or compound-complex sentence. Using the list below, find the letter corresponding to your choice and write it in the blank to the left of the sentence.

A. simple sentence
B. complex sentence
C. compound sentence
D. compound-complex sentence

(C) 1. During the winter the days are short and the nights are long.
(A) 2. Jane rode to school on her bicycle.
(B) 3. If Mary Lou had been home she could have visited with her grandparents and their friends.

[From Ahmann & Glock (1981)]

Figure 1.7: Classification Type of Matching Item

Choose homogeneous premises and responses (Ebel & Frisbie, 1986). Since each of the responses is, in turn, serving as a distractor for the other premises, it is necessary that they be plausible. Again, as in the multiple choice item, implausible responses are immediately set aside from consideration and thus raise the guessing percentages.

Arrange the responses logically (Gronlund, 1985). By logically arranging the response list (whether by alphabetizing or what have you), the test constructor can shorten the time required to make the associations, since the testee will be able to locate the desired response more readily.

It is imperative that the directions for a matching item clearly state the relationship to be considered when associating the premises and responses. Explicit instructions as to how answers will be registered should also be included.

1.3.3 The *True/False* Item

Summary:

The true/false item is simply a *declarative statement* to which the examine is to assign *truth* or *falsity*.

The most common usage of the true/false item is in determining if a student can recognize the accuracy of a statement made about a fact, definition, or statement of principles (Gronlund, 1985). In a more general setting, this item type functions as a degenerate version of the multiple choice format. In this case, only two options are available, and therefore only one distractor. This means that, in terms of the minimization of the guessing factor, the true/false item is lacking in reliability. For any given item, the student will always have at least a 50% chance of guessing the answer. However, if this shortcoming is set aside and a little creativity is applied, the true/false format can be bent for use in some surprising ways. Figure 1.9a presents a configuration which tests knowledge of cause and effect relationships.

The following is a sampling of suggestions for true/false item construction:

Avoid using negatives, especially double negatives (Gronlund, 1985). As was the case for multiple choice question stems, negation often needlessly confuses the issue, shifting emphasis from the content of the question to trying to discern its request. If negatives are unavoidable, emphasize them well.

Write declarations which are clearly true or false (Wiersma & Jurs, 1985). Probably the biggest problem with most true/false questions is ambiguity. Because the only allowable responses are true and false, the statement must

		Authors		Novels
(H)	(3)	1. Alexandre Dumas	A.	*The Adventures of Tom Sawyer*
(G)	(2)	2. George Eliot	B.	*Barchester Towers*
(F)	(3)	3. Victor Hugo	C.	*Call of the Wild*
(C)	(1)	4. Jack London	D.	*David Copperfield*
(E)	(1)	5. Herman Melville	E.	*Moby Dick*
(I)	(2)	6. William Thackeray	F.	*Notre Dame of Paris*
(B)	(2)	7. Anthony Trollope	G.	*Romola*
(A)	(1)	8. Mark Twain	H.	*The Three Musketeers*
			I.	*Uncle Tom's Cabin*
		Nationalities	J.	*Vanity Fair*
		1. American	K.	*War and Peace*
		2. English		
		3. French		

[From Ahmann & Glock (1981)]

Figure 1.8: Compound Matching Item

Directions: In each of the following statements, both parts of the statement are true. You are to decide whether the second part explains *why* the first part is true. If it does, circle Yes. If it does not, circle No.

Yes	(No)	1. Leaves are essential	*because*	they shade the tree trunk.
Yes	(No)	2. Whales are mammals	*because*	they are large.
(Yes)	No	3. Some plants do *not* need sunlight	*because*	they get their food from other plants.

[From Gronlund(1985)]

Figure 1.9: True/False Item Testing Cause and Effect Knowledge

exhibit one quality or the other. If there is some question as to the interpretation of the declaration, the student has no way to demonstrate his knowledge/understanding of the subject.

Confine the declaration to a single concept (Ebel & Frisbie, 1986). Unless a cause and effect relationship is to be tested, placing compound ideas into a single declaration can be confusing to the examinee, since some portion of the statement may be true and the other false. Simple, direct and concise declarations are best.

Have approximately the same number of true and false answers (Gronlund, 1985). As was the situation with multiple choice items, any discernible pattern of answers increases the percentage chance of guessing correctly. This is especially important in the true/false format where the percentage is already 50% at the start.

Although many consider the true/false format to be ineffective, Ebel and Frisbie contend that there is much to be gained from a well-written true/false item. One of their primary reasons for favoring this format is that "...students can attempt three true-false items in the time required attempt a pair of multiple choice items" (Ebel & Frisbie, 1986). An increase in the number of questions posed implies greater reliability and validity. In the final analysis, they say that "...a good one-hour true-false test is likely to be as effective as a good one-hour multiple-choice test" (Ebel & Frisbie, 1986).

1.3.4 The *Supply* Item

Summary:

Completion, short answer, and fill-in-the-blank are all examples of supply items. In each, the question consists of some amount of dialogue culminating in either a question or a blank in the sentence which must be filled (Ahmann & Glock, 1981). The answers for supply questions are extremely brief in nature, ranging from a single word to no more than a sentence or so. It is the testee's task to supply the answer outright, hence the categorization of this item format.

In many ways, a supply item is just a multiple choice item without the associated options list. In this respect, the supply item (especially in the short-answer form) bridges the gap between the multiple choice question and the essay item. The supply item receives the added bonus of having little or no guessing factor with which to contend. The primary objectives areas in which the supply question excels are recall of factual information and limited mathematical calculations. Scoring can be some-

what slower than the multiple choice format, since answers are provided rather than selected. However, a well planned test form with answer blanks positioned along one side of the page can overcome the scoring difficulties.

The suggestions for writing supply items center mainly on focusing the question well enough that a single answer presents itself.

Word the question such that is points to a single possible answer (Gronlund, 1985). In order to speed grading, the question should have only one possible answer, rather than a set of possible responses. This requires the test writer to clearly state and focus the problem. In many cases, it is helpful to construct the answer first, then write a question which leads to it (Ebel & Frisbie, 1986).

In numerical problems, clearly indicate the units in which the answer is to be expressed and the degree of precision expected (Ebel & Frisbie, 1986). Having all of the answers for a single problem expressed in the same units with the same precision will speed the grading process since this will serve to standardize the responses.

In completion items, make all blanks the same length (Ahmann & Glock, 1981) Blanks of different lengths may cue the testee as to the length of the correct answer.

Directions for supply items should clearly define the type and form the answer should take, including length and positioning of the response on the page.

1.3.5 The *Essay* Item

Summary:

The essay item is one which, after some amount of dialogue, requires the testee to supply some form of structured response. Such a response is typically a narrative (essay) at least one paragraph in length including some level of detail.

The essay item enjoys a long-lived popularity among educators. Historically, this form of testing has been in use since before 2300 B.C. in China, and up to the turn of the century was the only examination process in wide use (Ebel & Frisbie, 1986). Several factors have contributed to the endurance of this item format. A widespread belief that the essay format is inherently more effective than any other method coupled with the argument that essay questions measure higher-level thought processes have cemented this format firmly in testing practice (Wiersma & Jurs, 1985). Wiersma and Jurs go on to list several learning outcomes which can be measured by the essay item. These include such relational discussions such as comparison and contrast, developing solution plans for new problems, and organizing a defense of a position

(Wiersma & Jurs, 1985).

Certainly, the essay item can provide a considerable amount of insight into a student's thought processes. They are also generally easier to prepare – not because the items themselves are any easier to construct, but because fewer of them are required. There are a couple of things an essay item writer can do to make the question more effective.

Define the scope of the response (Wiersma & Jurs, 1985). By defining the scope and length of the response, the tester can indicate to the student what amount of discussion is required to answer the question. This can be made either in the form of a word count limit or topical bounds list.

Ask questions that will require a demonstration of mastery of essential concepts (Ebel & Frisbie, 1986). Questions of this type will reach beyond just restating textbook discussion by requiring several different factors to be integrated into a single response.

The scoring process for the essay item is less obvious than for the other four item types. Since the student is providing a free response, procedures and standards must be defined in order that all of the students' responses will be scored equitably. Identify crucial elements of the correct answer and then look for these in the responses. Also, by grading all of the students' essays on a single question before going on to the next essay item, since this keeps the overall standards of the responses freshly in mind. The final technique for grading essay items is to mask the identity of the writer as much as possible, and seek outside evaluation of some of the answers to confirm fairness. Because the essay question is, by nature, non objective, it is important to make scoring decisions from a well-informed perspective, keeping in mind the requirements of the question, the time allotted, and the overall quality of the responses.

1.3.6 Summary of Test Construction

The construction of a good test items is not an overnight process. Careful thought must go in to the construction of each to insure that it will perform the task it is designed to do: evaluate the student's learning of an educational objective.

1.4 Computers in Testing

During the course of the last ten to fifteen years, researchers have sought applications for the computer in the testing process. These usages have ranged from simple statistical calculations to advanced item generation programs. This section will provide a general review of the ways computers have been integrated into testing.

Research and development into Computer-Aided Testing (CAT) has generally taken one of three directions: Computer-Aided Test Adminstration (CATA), Compu-

ter-Aided Test Construction (CATC), and Computer-Assisted Statistical Analysis of Testing. Of the three, discussion here will revolve mainly around the first two, since the computerization of statistical calculations is a relatively simple task involving the coding of formulas and procedures which are well-known in the field of statistical analysis. Furthermore, many of the systems developed for CATA and especially CATC include provisions for statistical data. CATA and CATC offer a fertile field for research and development mainly because of the comparative lack of supporting work in test and test item writing (Roid & Haladyna, 1982).

1.4.1 Computer-Aided Test Administration

To a certain degree, computer-aided test administration must include some amount of computerized construction, since the test must be coded in a machine recognizable format. For purposes of this discussion, however, construction issues will be deferred until later.

The application of computers to the administration of tests, in its simplest form, is nothing more than a replacement for the traditional test form and answer sheet. Test questions are presented just as they would be in written form, with the student registering his answer on the keyboard rather than by written means (McKinley & Reckase, 1980). In this format, the primary benefits to the teacher come not at test-time but grading-time. Since the students' answers are already in machine format, the computer can be made to not only score the examinations, but also provide expansive statistical evaluations of overall performance, item performance, and the weaknesses of test and testee. Although this in itself can represent a considerable amount of information (to which the teacher might not have had access before), some researchers are making use of the interactive nature of online testing to explore new areas. Most of these center around *individually tailored testing.*

In tailored testing, the computer takes a more active role than simply presenting items and grading them post- facto. Instead of the traditional method of choosing questions, then administering the test, tailoring selects items during the testing process itself (McKinley & Reckase, 1980). The goal is to provide each student an examination which is geared to his specific abilities. This ability level is determined on-the-fly as the test proceeds. In this way, reliability is maximized by providing each student with a test which is geared toward his specific ability level. McKinley and Reckase (1980) reported that higher reliability was obtained for tailored tests which were less than half as long as conventional tests.

Another related usage of computer-aided test administration is in mastery and formative testing. In this method, described by both Brightman, et al (1984) and Bitzer & Braunfeld (1961), the computer administers one item at a time, providing feedback to the student about their choice of answer. Incorrect responses are met with dialogue detailing why that answer was incorrect, then providing a "second chance" for the student to answer the question. The PLATO system (Bitzer & Braunfeld,

1961) is a prime example of this type of system in action. PLATO provided short tutorials followed by diagnostic quizzes which directed the student to continued review, if needed. By using the computer test as a diagnostic tool, many of the advantages of individualized instruction were gained without any increase in required teacher time.

1.4.2 Computer-Aided Test Construction

Computer-aided test construction (as its name implies) concerns the use of computers – in some degree – in the contruction of tests. In its simplest form, this entails nothing more than a sophisticated word processing system which is specifically designed for the formatting of test items. CATGEN, created by L. W. McCallum (1985) is a representative system of this type. In this system, the test constructor (user) enters the test items in a special format language, then directs the system to produce the test and accompanying scoring key. Data relevant to each item (statistical measures of effectiveness, etc.) may also be entered with each question. The primary benefit of this variety of CATC is in the consistent and pleasing formatting of test pages. This does not, however, represent all that CATC has to offer the educator.

In a project undertaken by Jason Millman and Scott Outlaw (1978), the user enters not the items themselves, but *item programs*, or *shells*. The shells detail the procedures for generating an entire class of related and equivalent items, allowing the system to create many and varied tests, all different but of the same relative difficulty and content. This is an extension on the concept of *item banking*, a familiar term in CATC delevopment, in which databases of items are collected and classified so that test constructors or random test generators may choose from them. The idea of item shells holds great promise because the item shell writer does not code a question to test a specific objective, but rather the description of an item which would test that objective. This could ultimatly have connections to the field of artificial intelligence, since expert systems are themselves encodings of knowledge. Batteries of item shells could be considered as a primitive expert system representing a teacher. By developing these shells, educators would not only be gaining a valuable tool, but would be learning much about the testing process as they examine their own practices and procedures.

1.4.3 Summary of Computers in Testing

Although computers, in readily accessible form, are a relatively new occurrence, educators are already making use of them to further educational goals. In testing, these machines are providing advanced statistical diagnostics at the touch of a finger, as well as making possible new forms of testing and test construction. Some think that tests may one day be routinely administered by computer (David, 1985). Before this can happen, however, teachers must have time and motivation for becoming more computer-literate so that they may begin to realize the contribution that computers

can make to education.

1.5 Conclusion

In this chapter, various topics surrounding the field of educational testing have been considered. The "goodness" of a test was defined as that test's level of reliability and validity. The methods of constructing tests and test items were reviewed, placing emphasis on those techniques which will serve to enhance the reliability and validity of the instrument. Now that a working vocabulary for testing terminology has been established, discussion will now proceed to a presentation of "test pre-analysis" as a new means of judging the test and item performance.

Chapter 2

Test and Item Pre-Analysis

2.1 Current Directions in Analysis

To date, the great majority of research in test analysis has centered around the development of statistical measures which can, given the data from multiple administrations, produce statements about the measure's effectiveness. Through such methods, the researcher can not only determine the usefulness of an entire test, but also the particular performance of each individual item. [Note: For excellent discussions of the statistical methods involved, refer to (Gronlund, 1985) and (Wiersma & Jurs, 1985)]. Unfortunately, although these methods embody the robustness of mathematics, they can only be utilized on instruments which have already been administered, in many cases on a very wide scale. The researcher is left, then, with two equally distasteful alternatives: either a large group of test subjects must be located to whom the test can be administered, or the test must be given to the target population in the hopes that it is at least to some degree effective (with the intention of improving it before the next administration).

Computers have taken an important place in the evaluation of test data through statistical means. Many systems exist which can be used (by anyone with a personal computer) to perform very complicated and sensitive statistical tests. A quick examination of software house catalogs reveals this to be true. With the increasing prevalence of small, inexpensive computers in the schools, these systems are becoming more and more accessible.

Concurrent with this great effort in the development of statistical measures has been an equally strong effort in the definition and description of test and item writing "heuristics". A "heuristic" may be defined as "...a 'rule of thumb'" (Charniak & McDermott, 1985). That is to say, an *ad-hoc* rule used to describe a situation or define a rule which is generally known to be true, but is not definable in quantitative terms. In this way, any heuristic, and those involved in test construction in particular, are very qualitative in nature. Most elementary texts in test construction [including Ahmann & Glock, (1981), Ebel & Frisbie, (1986), Gronlund, (1985), Peddie & White,

(1978), and Wiersma & Jurs, (1985)] include some level of discussion about test and item writing, and then proceed to offer suggestions (heuristics) as general guidelines for construction. As a group, these collections of heuristics constitute an extensive body of qualitative, "rule-of-thumb" knowledge about test and item writing.

Although all of the aforementioned volumes mention the use of these heuristics as analysis guidelines, none considers the possibility of automation. Indeed, to date, no attempt has been made to consider this knowledge as a base for the construction of an automated system of analysis. It is this gap in both thought and implementation which this thesis seeks to address.

2.2 Pre-Analysis: A New Direction

In this research, we proposed a new direction in test and item analysis: qualitative pre-analysis. By combining knowledge and techniques from computer science and the qualitative test/item construction knowledge previously discussed, we proposed the creation of "expert systems" which, based on the heuristic knowledge in testing, can examine an item or test and make determinations as its "goodness", or prospective reliability/validity. By considering the instrument before it is put in to use, it may be possible to intercept many of the factors which would contribute to its poor performance, thus streamlining the validation process. It is very appropriate to consider the construction of such an expert system, since the knowledge to be used is of the common type; that is, heuristic rules which must be encoded in some fashion to be computer-understandable and usable.

2.2.1 Expert Systems

Expert systems are so named because "...they address problems normally thought to require human specialists for their solution" (Duda & Shortliffe, 1983). In this case, the specialist to be emulated is a test construction expert; someone who has a great deal of knowledge and practical facility in the processes of test and item writing. The goal of such development is to create the capability of placing the expertise of a testing specialist at the hands of anyone who might need its services (i.e., the classroom teacher concerned about his/her tests).

Many expert systems have been constructed and are now in use which make use of diverse areas of knowledge ("knowledge bases"). These include several in the area of medical diagnosis (Mycin, Puff, Internist, KMS and Casnet), and others in mathematics (Macsyma), chemistry (Dendral) and mineral exploration (Dipmeter Advisor and Prospector) (Nau, 1983). In most of these cases, the knowledge base consisted of heuristics, upon which an expert system was built. In this respect, the use of test construction heuristics as a knowledge base is very appropriate. The task, then, becomes the isolation of these heuristics in as complete a form as possible and

their subsequent translation to a machine-usable format. It is these tasks which the field of expert systems devotes its energies.

2.2.2 Developing an Expert System

The development of an expert system is no small task. Indeed, at the present time, many factors stand in the way of the creation of any new system. These include the effort involved in defining the heuristics and the development of a machine-representation for those heuristics. Although tools specifically designed with the expert system developer in mind are currently under development (Nau, 1983), they are far from being complete to the point of being user-ready. For now, the test and item construction heuristics included in the referenced volumes will serve as the source of expert knowledge (although this is done will the full knowledge that such knowledge is most definitely incomplete in its scope). The task then becomes the development of computer-usable representations of these heuristics. This, in itself, presents no mean task, given that many of those rules require advanced capabilities in linguistics and natural-language understanding. Additionally, the inclusion of some analysis factors (heuristics) requires domain specific knowledge of the field in which the test is written (content area knowledge). The concerted efforts of appropriate professionals in these fields could, however, produce just such a comprehensive system. Much of the required theories and procedures already exist, and other required knowledge (natural language understanding, content area knowledge) is the subject of a great deal of research in the artificial intelligence community.

2.3 Summary

Qualitative pre-analysis shows great promise as a new direction in test and item analysis. It affords the ability to use the great body of heuristic knowledge about test and item development which has heretofore gone unconsidered in a format which will allow test developers to receive preliminary evaluations of instruments before they are sent for field trials. Such pre-analysis and correction has the possibility to not only save time, but also money, as field trials represent a non-trivial financial outlay.

Much of the technology exists to develop such a system, although a complete implementation will have to wait upon the completion of ongoing research in natural language processing and content-area knowledge representation. The capability does, however, exist, to create a base-case implementation consisting of the less demanding analysis factors (heuristics). The construction of such a system will be the focus of the next chapter.

Chapter 3

The Base Case Expert System

3.1 System Overview

The Item and Test Analysis eXpert (ITAX) is a small scale PC-based expert system which performs pre-analysis of items and tests based upon accepted test construction heuristics. Designed primarily as a demonstration of the use of educational theory as the knowledge base for a computer expert, ITAX's design and implementation was focused mainly on the theoretical issues inherent in the development of any expert system. Accordingly, its user interface is not "market-ready", and was never intended to be. It is, however, a serviceable and dependable shell to which the more important work in expert systems design could be attached. A final interface could conceivably include anything from graphics displays to example items to interactive video remedial lessons. The focus of this research, however, was to show the viability of computer-based pre-analysis as a new direction in testing.

The emphasis in the following research report will, as a result, be primarily focused on the theoretical and technical issues concerned with the construction of an intelligent expert. A more complete reference work for the system may be found in the accompanying Appendices. These include not only a reasonably comprehensive reference manual for ITAX, but also documentation and design extracts.

ITAX consists of four stages, as shown in Figure 3.1. First, the user selects whether item analysis only is to be performed, or item and then test analysis. In some cases, the user may wish only to analyze a few unrelated items, in which case a test analysis is not necessary. Once this operating mode choice has been made, the user is placed in the item editor sub-system. This is a specially designed word processing system specifically geared toward the entry and editing of multiple choice and true/false items (for a discussion of the editor functions, see Appendix A, part 1.2, "Item Editor Functions"). Once the item is complete and ready f or analysis, the user selects the analyzer and is subsequently placed within the item analysis sub-system. At this point, the item is examined by the nine item analysis factors and appropriate diagnostic messages are displayed. Upon completion of analysis, the user may choose

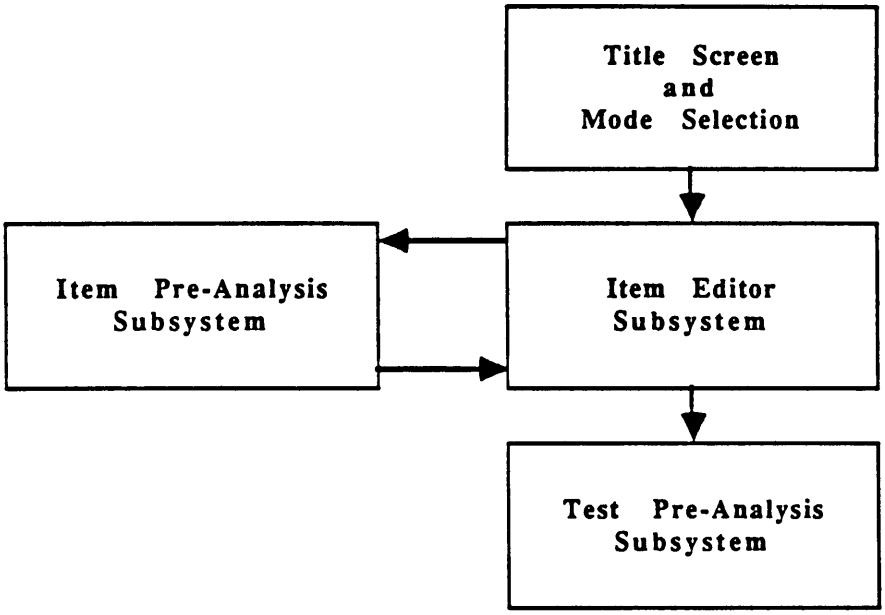


Figure 3.1: ITAX System Layout

to return to the editor in order to either edit the current item (for corrections) or enter a new one. This process continues until all items are processed.

At this point, if the user has selected test analysis, he/she may choose to continue to the test analyzer sub-system, where the entire body of items are analyzed for compliance with three test analysis factors. Completion of test analysis terminates the system and returns the user to the computer's command level. For a more detailed and in-depth presentation of user interface, see Appendix A, part 1, "User Dialogues".

3.2 Expert Systems in Testing

As mentioned earlier, the development of an expert system requires that the desired knowledge base be expressed in terms of heuristics. These heuristics are then used as the basis for the construction of an automated system which can function (to some degree) as an expert. In this case, the target knowledge base was the vast amount of heuristic knowledge in test and item construction. The goal of our research was the successful specification and development of an automated system capable of pre-analyzing items and tests for robustness of construction.

Although, in the general case, the expert system developer must work directly with a human expert in order to codify the required heuristics, this step was not needed in this case. Through the considerable work which has already been accomplished in test construction, much of the heuristic knowledge in this field has already been defined. In this way, one of the most difficult tasks in the expert system construction process was made unnecessary. Work in specification began, therefore, directly with the problems of choosing appropriate heuristics for consideration, and the subsequent translation of those heuristics into a machine-understandable format.

From the previous discussion of test construction theory, it may easily be noted that item and test writing heuristics cover a wide range of problem areas, ranging from simple parsing and detection of repetition, to detection of language and grammatical errors, to content-specific determinations of plausibility and consistency. The first task in the development of a base case system was, therefore, to glean through the great bulk of these rules to select a small number which are not only representative of the totality, but are also within the realm of probability of implementation.

Upon examination, the heuristic rules fell into two categories: item-related and test-related. Item related heuristics were those which concerned themselves with the analysis of a single, independent item, where test related rules required the analysis of a body of items. From the previous test construction theory presentation, nine item related and three test related heuristics were chosen for consideration in this system. For simplicity's sake, these heuristics deal with either multiple choice or true/false item types, this making the problem of item entry and editing much more manageable. The heuristics which were selected form the basis for the testing expert, and are hereafter referred to a "analysis factors". They will comprise the implementation domain for ITAX.

3.3 Item Pre-Analysis Specifications

Eight item pre-analysis factors were originally selected from the body of testing heuristics. Because one was later found to actually comprise two different but highly related problems, it was split into an 'a' and 'b' portion, thus giving, in effect, a total of nine item factors for consideration. Each item pre-analysis factor will be presented in conjunction with a discussion concerning its heuristic basis, final implementation and limitations. It should be noted that these analyses are invisible to each other. That is, the operation of one will not affect the performance of any other. In this way, the installation of additional pre-analysis factors as they are identified and designed is made simple. A more technical discussion of the item pre-analysis factors may be found in Appendix A, part 2.1, "Item Factors".

3.3.1 Item Pre-Analysis Factor #1: A/An Cues

The first item pre-analysis factor is concerned with grammatical consistency from the stem to the options in a multiple choice item. One of the item writing heuristics presented in Chapter 1 describes the errors which are introduced when there are options which do not grammatically follow the stem. The experienced testee knows, for instance, that a stem which ends in the word 'an' requires an answer which begins with a vowel, or 'h' in order to be grammatically correct. Conversely, a stem ending in 'a' must be followed by an option beginning with a consonant. If any of the options do not complete the sentence in a grammatical fashion, they can be discarded. This effectively nullifies the effectiveness of invalid options as distractors and increases the chance of a correct guess. It is therefore necessary to ensure that, should a stem end with either article, the options should be not only grammatical but homogeneous.

Item factor #1, therefore, takes as its data a multiple choice item and analyzes it for this situation. If the item is found to be in compliance with the rule, it is passed on without comment. If an invalid grammatical situation is detected, the user is alerted to the situation through the means of a short diagnostic message explaining the error and suggesting possible corrective measures. The offending options (those which do not follow grammatically from the stem) are highlighted on the screen display for reference.

3.3.2 Item Pre-Analysis Factor # 2: Plural Cues

The detection of plural cues also constitutes a grammar consistency detection. In this case, however, the task is somewhat more involved. Specifically, this heuristic states that the options must agree with the implied number in the stem. That is, if the stem implies a plural answer (normally through the tense of the verb), then all of the options should be stated plurally. Again, as was the case with a/an cues, any grammatical inconsistency would place incorrect options out of consideration as

possible correct answers, thus negating their value as distractors. If this situation is found to exist in the item, the options detected as singular are highlighted for reference.

This heuristic presents an interesting and challenging problem in implementation. On the surface, it would seem to require the system to have an advanced knowledge of natural language, and indeed this would need to be the case in any complete implementation. For the purposes of this demonstration system, however, the situation was simplified and narrowed in scope. Instead of determining the tense of the verb (a considerable task), the solution used was to examine the options for consistency of number. If there is some inconsistency in number, then this would signal that, no matter the tense of the verb in the stem, there would be a problem. This method does not, however, detect a homogeneous set of options which are cumulatively in disagreement with the tense of the verb.

Plurality of options was also defined in a very narrow scope to bring it into a more manageable range for implementation. For these purposes, the options must consist of single words (discounting leading incidental articles) or conjunctions of single words (with 'and') which, when singular do not end in 's' and are pluralized by the addition of 's' or 'es'. Any conjunction in an option makes that option plural no matter the endings of the associated words.

Under these constraints, the system correctly identifies items have inconsistencies in number. Granted, this represents a significant simplification of the plural detection problem, but it is easy to see that, under more complete implementation circumstances, the redefinition of this item pre-analysis factor to perform true natural language parsing and detection is not outside the realm of possibility. In fact, most of the cases of simplification which were necessary for this base case system do not represent significant departures from the possible reality of complete implementation. Such completeness would, however, require advanced abilities in natural language processing and would moreover require a significant amount of computing facilities.

3.3.3 Item Pre-Analysis Factor #3: Negatives

Item Pre-Analysis Factor #3 is concerned with detecting the positive/negative sense of the stem (multiple choice) or statement (true/false) of an item. Specifically, the heuristic states that is is generally better to state a stem/statement in the positive in order to minimize testee confusion. Additionally, multiple negatives are to be avoided as they only exacerbate the problem. Negativity, however, is another of the item factors whose specification is tied up with a grammatical issue; that is, the detection of negativity. Were it solely the case that negativity is represented by negative *modifiers* in the sentence, there would be little difficulty (simply parse and detect). Unfortunately, this is not the case. It is also possible that other methods (most notably negative-implying prefixes and suffixes) can denote negativity. As

was the case in Item Pre-Analysis Factor #2, simplification and strict definition was necessary in order to bring this item into the realm of implementability within the parameters of this system.

For the purposes of this demonstration system, negativity will be restricted to the sensing of the presence of a specific set of negative modifiers in the stem/statement. Multiple negativity, then, will be defined as the presence of more than one of these negative modifiers. For our purposes, the set of negatives will be defined as the following: *no, never, not,* and *none.* The presence of one of these words will be considered as making the stem/statement negative, and the presence of more than one will make it multiply negative.

When negativity is detected, the sensed modifiers will be highlighted for reference, and a message explaining either single or multiple negativity will be displayed.

3.3.4 Item Pre-Analysis Factor #4: Repeated Phrases

This pre-analysis factor is also concerned with the presentation of a non-confusing item. In this case, however, the concern is not grammatical negativity, but the multiple repetition of a phrase in each of the options of a multiple choice item. Again, as a matter of clarity, the item construction heuristic says that each item option should begin uniquely, in order to maximize the perceived difference among the options. This also creates a more parsimonious item which reads fluidly.

With Item Pre-Analysis Factor #4, we finally come to a relatively simple heuristic to implement. Essentially parse-and-detect, all this involves is the determination of repetition, and if it is found, its extent. If repetition is found, the repeated phrase in each of the options in highlighted in order to demonstrate the situation and an appropriate explanation is supplied.

3.3.5 Item Pre-Analysis Factor #5: Too Few Options

Another test construction heuristic details the performance characteristics of multiple choice items with given numbers of options. It is always possible to register a "guess" on any multiple choice item by randomly selecting any of the items in the list. Accordingly, it is necessary that the effects of such random guessing be minimized. Statistically, items with only two options represent a 50% possibility that the testee will select the correct answer. This, however, places it on essentially the same ground as the true/false item, and therefore does not supply any additional sensitivity to the test. It is generally agreed, therefore, among test construction theorists, that any multiple choice item should have at <u>least</u> three options in the options list, with four or five being most desirable.

The implementation of this factor is a trivial matter of counting the number of options in the list and providing an appropriate diagnostic message. In this case, the options need not be sequential in the list; that is, internal, unused option positions

will not present any difficulty to the measure.

3.3.6 Item Pre-Analysis Factor #6: Inconsistent Length of Distractors

In this pre-analysis factor, the heuristic deals with an issue commonly referred to as the "savvy test-taker". A "savvy test-taker" is someone who, after having taken a great number of tests (especially multiple choice format tests) has come up with his/her own set of test-taking "heuristics", designed to maximize the possibility that guesses will turn out to be correct. These heuristics commonly deal with tendencies that inexperienced or incapable test constructors generally demonstrate. One of these tendencies is the focus of this factor. The heuristic involved states that, among inexperienced test constructors, there is a tendency to make the correct option in a multiple choice item the textually longest of the list. This often occurs because it may take extra modifiers to specify the correct response. In a guessing situation, therefore, the savvy test-taker will choose this as the correct option. Accordingly, it is advisable that all of the options in a list be of approximately the same length, in order to foil this testee heuristic.

The method of implementation for this factor involves a simple statistical manipulation. The length of any option is defined as the number of words in that option. From these lengths, a mean and standard deviation of the length of options are calculated. Chebyshev's Inequality (Walpole & Myers, 1978) states that the probability of any length falling outside a range defined by k standard deviations on either side of the mean is described as:

$$P(|X - \mu| > k\sigma) \leq \frac{1}{k^2}$$

That is to say, the chance percentage that any individual length is outside the k standard deviation range of the mean is less than or equal to $\frac{1}{k^2}$. This, however, assumes a worst-case scenario and an unknown distribution. In the case of a normal distribution, this percentage becomes markedly less. For example, the probability that any X falls outside a 2 standard deviation range on either side of the mean in a normal distribution is on the order of 5%. Chebyshev's Inequality merely defines the worst-case that is possible.

Using the example in Appendix A, part 2.1, Item Factor #6 as a sample of an expert-defined detectable situation and working backwards to the k required to make it detect, we calculate $k = 1.47$. For current purposes, this k value will be used as the detection range for this factor. (For any future system, it would be necessary to work with experts in the final calculation of this constant.) Given this k value, the <u>maximum</u> percentage of options which would be expected outside the acceptable range would then be $\frac{1}{k^2}$ or approximately 46%. Although this may seem, at first, a little high, it should be noted that this represents an <u>exceedingly</u> pessimistic statement, and the <u>actual</u> percentage is most probably much lower.

Due to the statistics involved in this factor, one limitation applies. There should be more than one option in the list, since the statistics involved mean very little in this trivial case ($P(|0-0| > 1.47(0)) \leq 1$ is a useless statement...).

3.3.7 Item Pre-Analysis Factor #7a: "All of the Above"

The heuristic for this factor refers to the use of (the ever-popular) "all of the above" in a options list. Again, we have a "savvy test-taker" situation. In this case, if any two options in a list can be determined as correctly following the stem, then "all of the above" can be chosen immediately as the correct solution. Conversely, if any option can be eliminated from consideration, the "all of the above" can be immediately disregarded. This action cripples the effectiveness of remaining options, since they never come into consideration. "All of the Above" is therefore discouraged in use.

The implementation of this factor is another "parse-and-detect" situation. Specifically, the "all of the above" must appear in exactly one manner: all words included, each in lower case letters. Extra spaces between words will be ignored. A detected occurrence will be highlighted and supplemented with an appropriate diagnostic message,

3.3.8 Item Pre-Analysis Factor #7b: "None of the Above"

This factor is very much like the previous one. In point of fact, they were originally destined to be considered within the same analysis factor. Implementation difficulties, however, deemed their separation useful. "None of the above" is discouraged because of clarity problems, although testing experts do agree that there are applications where its use can be indicated.

As was the case with factor #7a, the string must appear with no missing words and all in lower case letters for the pre-analyzer to detect its presence. Highlighting and a diagnostic message, this time not stating error but just warning, provide the dialogue.

3.3.9 Item Pre-Analysis Factor #8: Repetition of Significant Words

The final pre-analysis factor returns to a more difficult detection problem. In this case, the heuristic requires that significant words in the stem of a multiple choice item not be repeated in any of the options. This repetition may provide a clue to the correct option, thus reducing the item's effectiveness. The main difficulty with this factor is the arrival at a working definition of "significant". Were the system to have content-area knowledge, significant words might be defined in terms of content-area specific terms. In our system, however, we need a more direct and simple definition. For our purposes, therefore, significance will be defined as any word which is a) over

two letters in length and b) is not any of the incidentals *the, not, nor, but,* or *and.* Note that additions to this list are easily made and are anticipated.

The factor creates a list of the stem words which qualify as 'significant'. These are then compared against the words in the 'significant' words in the options, searching for duplications. All of the existing occurrences are detected, in sequence from the beginning of the stem 'significant's to the end. It should be noted that two 'significant' words will match only if they are expressed identically (identical cases of letters).

3.4 Test Pre-Analysis Subsystem

Test Pre-Analysis is the process of applying some test heuristic which requires a subset of items upon which to work. Of the three which were selected for incorporation into ITAX, two have the totality of true/false items which were entered as their subset, and the remaining one uses the group of multiple choice items. It should be noted that, since all three factors use statistical measures to some degree, a sufficient number of items must be present in the subset for these calculations to have meaning. Generally speaking, a minimum of ten would be appropriate.

As was the case with the item pre-analysis factors, each of the following test pre-analysis factors will be presented in conjunction with a discussion concerning its heuristic basis, final implementation and limitations. A more in-depth and technically oriented review of these factors can be found in Appendix A, part 2.2, "Test Factors".

3.4.1 Test Pre-Analysis Factor #1: Randomness of M/C Answer Placement

The first test pre-analysis factor is concerned with a test-wide "savvy test-taker" heuristic. If patterns in correct answer placement can be detected in the test, the savvy test-taker may become aware of these and use them as cues when guessing. By being consistent and predictable in correct answer placement, the test constructor is therefore reducing the effectiveness of each item, since the distractors in each list are being totally ignored in favor of a completely non-content sensitive answering strategy. It is therefore imperative that the positioning of the correct option in the distractor list be made as statistically random and free from pattern as possible.

The determination of randomness and the detection of patterning actually constitute two separate (but related) problems for the pre-analysis factor. In this context, randomness of placement is defined as the relative balance between the number of times each of the five possible option positions is used for the correct option. This means that items which have two options must be considered separately from those with three and so on; this because items with three options did not consider position five, etc. Patterning of placement, on the other hand, is a much more difficult concept with which to contend. In simplest terms, patterning of placement means

the detection of <u>any</u> recognizable pattern in the placement of the correct option. The important (and difficult) word here is 'recognizable'. Recognition of such patterning is an ongoing area of study in the field of artificial intelligence and involves many complex ideas. Statistically speaking, it may be possible (with a large enough sample) to demonstrate *tendencies* in placement, whether to the left, right or center of a list, for example, but it is very difficult to say anything more. Because this notion of pattern detection demonstrates a high degree of complexity and is, moreover, virtually on the leading edge of artificial intelligence research, it will not be addressed in this study. This could, however, prove a fertile area for further combined efforts of computer scientists and testing experts. Restricting ourselves to the determination of randomness, then, brings us to a more narrowly defined and realistically implementable factor.

Because this factor is examining a set of multiple choice items with regards to the placement of the correct option in the option lists, it is inappropriate to consider items with 3 options in the same class as items with 4, and so forth. The lack of equal probability of placement makes them non-homogeneous situations, and therefore in separate classes. This being noted, then, the problem of randomness divides itself into 3 separate categories, or classes: items with 3, 4, and 5 possible options. Each of these will be considered independently. If any of the three show non-randomness, they will be reported as such, stating the deficiency and class of items which are at fault.

The statistic used to make the randomness determination is the Chi-Square Goodness-of-Fit test. In this case, the distribution to be fitted is linear, where all cells (option positions) have equal weight (expressed as the total number of items in the class divided by the number of cells in that class). With this scenario, we construct the following chi-square test for each class. Given the total number items in the class (N), and the number of cells in the class (C, the number of possible option positions), define the expected frequency for each cell in the class (e) as $\frac{N}{C}$. For each class, then, the observed frequency (o_i) for each cell is represented as the number of items in the class which have their correct option in the position represented by the cell. Since there are C possible option positions in each class (3, 4, or 5), there will be C cells, denoted o_1 to o_C. The null hypothesis then becomes:

$$H_0 : e = o_i \quad \text{for } 1 \leq i \leq C$$

and calculate the chi-square statistic for each class with the formula:

$$\chi^2 = \sum_{i=1}^{C} \frac{(o_i - e)^2}{e}$$

With a 0.05 level of significance and $C - 1$ degrees of freedom, the critical values of the chi-square test are as follows (from Walpole & Myers, 1978):

C	df	critical value
3	2	5.991
4	3	7.815
5	4	9.488

If the calculated chi-square value for the class is less than the associated critical value, then we fail to reject H_0 and say that the placement of the correct option is random. Otherwise, we reject H_0 and say that the distribution is *not* balanced.

Because the chi-square test is used, the sample size for each class must exceed $5C$, in other words, each cell of the class must have *at least* 5 members for the statistic to be accurate. This restriction is acceptable for current purposes, although it should be noted that samples with less than $5C$ items are a valid situations and should be considered in the creation of any future versions of the system.

3.4.2 Test Pre-Analysis Factor #2: Balance of True/False Items

The heuristic for this pre-analysis factor is concerned with the assurance that there are approximately the same number of 'true' and 'false' true/false items in a test. In order that the "guess percentage" of 50% be maintained, the true/false items must not be shown to demonstrate any bias toward one answer or the other. If the test-taker detects an imbalance, this may allow him/her to make guesses based on the notion that "most of the true/false items have been true so far, so this one probably is, too". This biases the student's correct guesses away from the 50% maximum and works against the test's effectiveness.

The implementation of this factor uses the Chi-Square Goodness of Fit test with two cells, which implies a needed minimum of 10 items to make it valid. This constitutes an acceptable limitation. For the test, we define a null hypothesis that the observed number of 'true' and 'false' items (represented by o_t and o_f, respectively), will each be equal to one half the total number of items ($\frac{N}{2}$), or:

$$\mathrm{H}_0 : o_t = o_f = \frac{N}{2}$$

One of the two cells contains the number of 'true' true/false items (N_t) and the other contains the number of 'false' (N_f). From this, we calculate $o_t = \frac{o_t}{N}$ and $o_t = \frac{o_f}{N}$. Given a 0.05 level of significance and 1 degree of freedom (# of cells - 1), the critical value for the Chi-Square test will be 3.841 (Walpole & Myers, 1978). Using the following formula, we also calculate the chi-square value for the data:

$$\chi^2 = \frac{(N_t - \frac{N}{2})^2}{\frac{N}{2}} + \frac{(N_f - \frac{N}{2})^2}{\frac{N}{2}}$$

If the calculated chi-square value for the data is less than the critical value, then we fail to reject H_0 and say that the number of true/false items is balanced. Otherwise, we will reject the null hypothesis and say that it is *not* balanced. A rejection of H_0 brings a message to the user about the imbalance, including the number of 'true' and 'false' items and the calculated chi-square value for reference purposes.

3.4.3 Test Pre-Analysis Factor #3: Length of True/False Items

The final test pre-analysis factor deals with an issue which has been addressed previously for Item Pre-Analysis Factor #6: Inconsistent Length of Distractors. In this case, however, the heuristic is concerned not with the length of a set of multiple choice options, but the lengths of all of the true/false items in a test. The heuristic states that, much as was the case before, inexperienced test constructors tend to write proportionally longer 'true' true/false items, since these seem to require extra modifiers and descriptors. The savvy test-taker sees this and judges his/her guesses accordingly, thus reducing the effectiveness of the item.

This factor is addressed in much the same fashion as was item factor #6, in that a mean and standard deviation for statement lengths is calculated and any items which fall outside an "acceptable" range of the mean are flagged as being "too long". In this case, the value of k was chosen as 1.41, which (by Chebyshev's Inequality) places the sensitivity level of the measure at almost exactly 50%. This was chosen because item factor #6 itself required a large tolerance (about 46%) and this situation is very much the same. Through trial-and-error testing, this percentage has been shown to perform reasonably well. The development of more complex versions of this system will no doubt require that some sampling of expert-generated items be done in order that a value with a more firm statistical base may be produced.

3.5 Summary

In the implementation of these pre-analysis factors, all of the measures were designed to be exceedingly conservative and pessimistic in their judgements. That is, error was biased on the side of detecting and flagging only those items and situations which truly meet the heuristic requirements. This was done, in many cases, with the full knowledge that instances of the heuristic which should be detected may slip by the system. In more statistical terms, if the null hypothesis is that the item or test is free from error, then the system has been biased to make TYPE II errors (accepting it, even though false). It was deemed important that the system be as correct as possible when rejecting an item or test. This is reflected in the user dialogues for each factor which, when they find no error, report that the item/test has "cleared the pre-analysis factor" rather than being free from error (a much more sweeping

statement).

During the discussion of the factors, many of the more difficult issues in detection were placed outside the boundaries of the system's capabilities. These should not be considered as impossible. Rather, they represent situations which were deemed too complex for this system, but may, in many cases, be possible in a more complete, more comprehensive system. Indeed, the researchers have noted that it may be desirable for more than one implementation of the heuristics to be incorporated so that the system may be made as sensitive as possible. It is suggested, however, that any future implementations continue the bias toward TYPE II errors. It is much more important that the expert system, functioning as a diagnostic tool for test constructors with little ability or knowledge, be encouraged to reject *only* those items/test which are demonstrably incorrect for some reason. In the flagging of a correct item as incorrect, user confusion is bound to result.

The implementation of this expert system represents a new direction for both test analysis and expert systems development. For educators, it provides the possibilities that the proposed field of pre-analysis has to offer. For the computer scientist, it offers a fertile proving ground for artificial intelligence and expert systems theories, since it incorporates such a diverse range of different problems within the field.

Chapter 4

Conclusions

As a new direction in test analysis, computer-assisted pre-analysis has much to offer educators. As a formative measure, it can offer suggestions as to the robustness of item and test construction; suggestions which the more conventional statistical methodologies cannot make due to their summative functioning. As an addition to the analysis methods already in existence, it brings the great body of test construction heuristics to bear in a diagnostic format. As an analysis method in itself, it operates as a direct method for isolating the specific problems in items and tests, rather than just identifying statistically ineffectual components.

The creation of expert systems which use test construction heuristics as their knowledge base represents a significant new use of the pre-codified heuristic knowledge which has existed in the field for some time. With these new systems, the educator is able to not only directly analyze instruments before field use, but is also able to bring this knowledge into the classroom on a more effective, more robust level. Systems of this kind can be incorporated into computer-aided instruction packages in test and item writing, as well as being used as a diagnostic tool for evaluating test and item writing facility. The addition of new interface techniques such as interactive video and graphics may also add a new dimension to testing expert systems, providing not only for analysis, but also remedial, on-the-spot tutoring.

Created as an example of what a computer-based pre-analysis system might be capable, ITAX, the Item and Test pre-Analysis eXpert, uses well-known test writing heuristics to analyze items and tests for errors in construction. Its implementation represents a significant new area of research in both education and computer science, and sets the stage for more complete, more comprehensive systems to be created.

As a new field in test analysis, there are many directions which future research may take. The complete codification of test writing heuristics with an eye toward implementation, the development of more complex and comprehensive experts of this type, and the use of such systems to evaluate test writing facility among educators are only a few of the new and exciting problems which this research brings to life.

Pre-Analysis represents a significant new methodology for test analysis, and should

be vigorously pursued in continuing research in order that the fullest possible use may be made of this advance in the field of test and item analysis.

References

Ahmann, J.S., & Glock, M.D. (1981). Evaluating student progress: Principles of tests and measurements (6th ed.). Boston: Allyn and Bacon.

Bitzer, D.L., & Braumfeld, P. (1961). PLATO: An automatic teaching device, (CSL Report I-103), Urbana: University of Illinois, Coordinated Science Laboratory.

Brightman, H.J., Freeman, J.L. & Lewis, D. (1984). Constructing and using computer-based formative tests. Educational Technology, 24(6), 36-38.

Byrkit, Donald (1972). Elements of statistics. New York: Van Nostrand/Reinhold.

Charniak, E. & McDermott, D. V. (1985). Introduction to artificial intelligence. Reading, MA: Addison-Wesley.

David, C.W. (1985). A student can learn even as he is being tested. College Board Review, 136, 27-30.

Duda, R. O. & Shortliffe, E. H. (1983). Expert systems research. Science, 220(4594), 261-268.

Ebel, R.L. & Frisbie, D.A. (1986). Essentials of educational measurement, (4th ed.). Englewood, NJ: Prentice-Hall.

Edwards, J.S. (1980). Computer-assisted testing. Journal of Research and Devel -opment in Education, 14(1), 9-15.

Good, C.V. (Ed.). (1973). Dictionary of education (3rd ed.). New York: McGraw-Hill.

Gronlund, N.E. (1985). Measurement and evaluation in teaching (5th ed.). New York: Collier Macmillan.

McCallum, L.W. (1985). CATGEN: A computer-assisted test generator. Teaching of Psychology, 12(1), 45-46.

McKinley, R.L. & Rechase, M.D. (1980). Computer applications to ability testing. AEDS Journal, 13(3), 193-203.

Millman, J. & Outlaw, S. (1978). Testing by computer. AEDS Journal, 11(3), 57-72.

Nau, D. S. (1983). Special feature: Expert computer systems. IEEE Computer, 16(2), 63-85.

Peddie, B. & White, G. (1978). Testing in practice. Aukland, New Zealand: Heinemann.

Roid, G.H. & Haladyna, T.M. (1982). A technology for test-item writing. New York: Academic Press.

Walpole, Ronald E. & Myers, Raymond H. (1978). Probability and statistics for engineers and scientists. New York: MacMillan.

Wiersma, W. & Jurs, S.G. (1985). Educational measurement and testing. Newton, MA: Allyn and Bacon.

Appendix A

ITAX Reference Guide

A.1 User Dialogues

A.1.1 Item Editor Dialogues

At start-up, the user will be presented with a title screen identifying the system. At this time, he will also be asked to choose one of the two modes of execution: item analysis only or item/test analysis. Should item analysis only be chosen, no cumulative record of entered items will be kept. This mode would be appropriate in circumstances where there are individual items to be examined which do not constitute an examination. Only the item analysis factors would then be under consideration. On the other hand, in item/test analysis mode, an account would be kept of all entered items in order that test factors might also be considered. For this mode, both sets of target factors (item and test) would be used. This allowance for two different modes of execution adds desirable flexibility to the system. Once the user has made his choice, the title screen will be replaced by the item entry screen, where he may enter his items for analysis. This screen is composed of four areas. The screen label area identifies this as the item entry function of the system. Following that is the current item type area, which denotes the currently active item type for which the entry area is configured (multiple choice or true/false). The entry area, then, constitutes the central portion of the display where the item itself is entered. For multiple choice items, space is provided for entry of the stem and up to 5 options. The correct option is indicated by moving the "CORRECT" marker to the appropriate option. For true/false items, the statement space is followed by marker spots for TRUE and FALSE. The editor for both versions will function as a full-featured screen editing system. In the fourth area, the currently available functions menu, all of the currently active functions are displayed. All function selection will be accomplished through the use of the function keys available on the keyboard. A description of the action of each function now follows.

Function Key #1: Item Type Toggle

Selecting this function causes the display to be reformatted for entry of the other item type (essentially a toggle).

Function Key #2: Analyze This Item

This function will cause the system to proceed with an analysis of the currently active item.

Function Key #3: New Item

This option should be selected when the user has finished manipulating the current item and wishes to continue to the next.

Function Key #4: Test Analysis

This function will only appear on the screen if the user selected the item/test analysis mode of execution from the title screen. It signals to the system that the user has completed entry of items and wishes to go on with the test factor analysis.

Function Key #5: Exit The System

This terminates the session and returns the computer to the operating system command level.

A.1.2 Item Editor Functions

The ITAX Item Editor is designed to function as a specialized full-screen editing environment for the entry and alteration of Multiple Choice and True/False items. Although the editor screen is configured differently for each type of item, the action of the editor takes only two forms:

Textual Editing

In the stem and options fields of the Multiple Choice item and the statement field of the True/False item, the ITAX Item Editor functions as a typical full-screen editing system.. That is, the user may access any portion of the edit text by moving the cursor from line to line and column to column through the use of the arrow keys on the IBM PC keyboard. When the cursor passes from one field to another (for example, the stem field to the options field of the Multiple Choice item), it is repositioned to the beginning of the first line of the new field.

Addition of text to that being edited is accomplished in two modes: overtype and insertion. In overtype mode (the default), any character typed by the user will replace the character upon which the cursor resides. In insertion mode (toggled through the use of the INS key on the keyboard and indicated by a small 'Ins' in the lower right corner of the screen), a newly typed character takes the position of the character

upon which the cursor resides, and all characters to the right are shifted one position to the right.

Deletion of text from the field is also accomplished through two modes: rubout and deletion. In rubout (through the use of the backspace key), the character to the immediate left of the cursor is deleted and all characters to the right of the cursor are shifted one position left to fill the resulting hole. In deletion (through the use of the DEL key), the character under the cursor is deleted and all characters to the right are again shifted one position to the left to fill the resulting vacancy.

Textual editing, then, takes the common full-screen form which any reasonably experienced user will recognize.

Non-Textual Editing

The editing of non-textual fields takes a different form. These fields (the correct answer fields for both item types) are handled in a more direct method. For the Multiple Choice item, when the user moves the cursor past the last line of the options field, the 'CORRECT' marker is illuminated and may be moved through the selection of the correct option. This is accomplished by striking the appropriate numeric key on the keyboard. A further striking of the Downarrow key will move the cursor from this field and back to the stem area. In the True/False item, passing beyond the statement field illuminates the TRUE or FALSE marker, which may be selected through the striking of the 'T' or 'F' key on the keyboard. Again, a further striking of the downarrow key will move the user back to the statement area for standard textual editing.

A.1.3 Item Analysis Dialogues

Upon selection of Function #2 on the item entry screen, the system will proceed to an examination of the item with regards to the nine item analysis factors. At this time, the item entry screen will be replaced by the item analysis screen. The item analysis screen is composed of five areas. The screen label area now indicates that item analysis is the currently active function and the item type area denotes the type of item under analysis. In the item display area, the text of the item being considered is displayed for reference. As was the case with the item entry screen, this area takes two forms, one for multiple choice items and one for true/false items. Below this is the analysis dialogue area, where notations will be presented as the item is examined by each factor. Finally, the function menu displays the currently available option. At this point, there are two possible functions:

Function Key #1: Display Next Analysis

Upon selection, this causes the next applicable analysis dialogue to be presented. Upon presentation of the last of these, this will read "Analysis Complete". Subsequent selection will cause the cycle to begin again.

Function Key #2: End Analysis

This will cause the system to return to the item entry screen, retaining intact the currently active item.

A.1.4 Test Analysis Dialogues

Upon selection of Function #4 on the item entry screen, the system will proceed to an examination of the test as a whole with regard to the three test analysis factors. At this time, the item entry screen will be replaced by the test analysis screen. The test analysis screen is composed of four areas. The screen label area now indicates that test analysis is the currently active function. In the analysis specification area, additional information/data pertaining to the test analysis factor currently being considered is displayed for reference. Below this is the analysis dialogue area, where notations will be presented as the test is examined by each factor. Finally, the function menu displays the currently available options. At this point, there are two possible functions:

Function Key #1: Display Next Analysis

Selection of the function key advances the test analysis system to the next applicable test analysis factor.

Function Key #5: End Analysis

Selection of this function key terminates test analysis, and also the entire system. The user is returned to the operating system command level.

A.2 Specification of Analysis Factors

There are many factors and suggestions in good test design which can be either partially or completely incorporated into an automated system. By including as many of these as possible into a computer-based test analyzer, the test writer can be afforded the most complete evaluation of that test which is possible without actual performance data. For this base case implementation, however, the goal is to construct a first-generation system to show the practicality, feasibility and usefulness of automated pre-analysis. Accordingly, many factors whose implementations would rely heavily on advanced artificial intelligence/expert systems techniques or content area knowledge must be excluded from consideration at this time. For demonstration purposes, the multiple choice and true/false item types will be considered, since they represent the great bulk of item writing guidelines and lend themselves well to both entry and analysis.

The identified factors for analysis fall into two categories: item-related and test-related. Item-related factors are those factors which may be applied to individual

items, regardless of their association to the rest of the test. On the other hand, test-related factors are those which require the examination of some subset of items on the test. The specific factors chosen for implementation in this base-case system will now be presented.

A.2.1 Item Factors

Each of the following item analysis factors, taken from the previous test construction theory discussion, can be applied to items independently of the remainder of the test. Included with each factor will be its usage (multiple choice or true/false items), specification (the conditions which will trigger the factor), a triggering example, an example of the dialogue which would be presented to the user, and any special notations about the performance of the factor.

Item Factor #1: A/An Cues

Usage: Multiple Choice Items

Specification:

This factor will be triggered if the stem ends with the word 'a' or 'an' and any of the options does not agree.

Trigger Example:

```
The word 'the' is called an
    a) article
    b) noun
    c) verb
    d) adjective
```

Dialogue:

Your usage of 'an' ('a') cues the testee that the highlighted option(s) will not be correct. Suggestion: Replace 'an' ('a') with 'a' ('an') or rephrase the stem.

Notes:

In this case, options b) and c) would be highlighted.

Item Factor #2: Plural Cues

Usage: Multiple Choice Items

Specification:

This factor will be triggered if the options are not consistent as to number (singular or plural). This detection is restricted in that the options must be single words (disregarding articles) which are made plural by the addition of 's' or 'es'. Conjunctions (using 'and') of such single words are also allowed and will be considered plural.

Trigger Example:

```
Animals which bear and suckle live young are called:
    a) invertebrate
    b) mammals
    c) amphibian
    d) reptiles
```

Dialogue:

The options do not agree as to number. This may make some of them grammatically incorrect as answers to the stem. Suggestions: re-examine the stem as to number and make sure the options all agree.

Notes:

The restrictions were necessary, since the more obvious method of testing for number (parsing the stem to determine the plurality of the verb) presents a level of complication which is beyond the scope of this work.

Item Factor #3: Negatives

Usage: Multiple Choice and True/False Items

Specification:

This factor will be triggered if any of the following negatives are located in the stem/statement: *no, never, none, not.* Detection of one negative will produce Dialogue A, while detection of multiple negatives will produce Dialogue B.

Trigger Example:

```
Dialogue A
Multiple Choice-
     Which of the following is not a fish?
          a) whale
          b) trout
          c) perch
          d) salmon

True/False
     A whale is not a fish.

Dialogue B:
Multiple Choice:
     Which of the following is never considered not
     to be a fish?
          a) whale
          b) trout
          c) perch
          d) seal

True/False
     No shark is ever found not swimming.
```

Dialogues:

A: You have phrased your stem/statement in the negative. It is generally better to use the positive when possible. Suggestion: re-examine your stem/statement to see if it can be phrased positively.

B: You have used a multiple negative in your stem/statement, which may lead to testee confusion. Suggestion: rephrase to simplify the negation, preferably phrasing it in the positive.

Notes:

The detected negative(s) will be highlighted on the screen for reference. Negative prefixes are <u>not</u> included in this factor.

Item Factor #4: Repeated Phrases

Usage: Multiple Choice

Specification:

This factor will be triggered when the options are all found to begin with the same word or phrase.

Trigger Example:

```
Helium is
    a) an example of a noble gas
    b) an example of a compound
    c) an example of an elastomer
    d) an example of a suspension
```

Dialogue:

The highlighted word/phrase is repeated in every options, thus decreasing clarity. Suggestion: move this word/phrase to the stem.

Notes:

The phrase 'an example of a' in each options will be highlighted.

Item Factor #5: Too Few Options

Usage: Multiple Choice Items

Specification:

This factor will be triggered when the number of options given is less than 3.

Trigger Example:

```
Helium is an example of a
    a) noble gas
    b) compound
```

Dialogue:

You have only supplied (number) options. In general, it is best to have at least 3. Suggestion: review the stem and develop additional options.

Notes:

None.

Item Factor #6: Inconsistent Length of Distractors

Usage: Multiple Choice Items

Specification:

This factor will be triggered when the length of any option is significantly longer than the mean length of the set.

Trigger Example:

```
What is the purpose of the United Nations?
    a) to maintain peace among the nations of the world.
    b) to establish international law.
    c) to provide military control.
    d) to form a new government.
```

Dialogue:

The highlighted option(s) is significantly longer than the others, possibly cueing it as the correct choice. Suggestion: re-examine the options and make their lengths more uniform.

Notes:

In the example, option a) would be highlighted.

Item Factor #7a: All of the Above

Usage: Multiple Choice Items

Specification:

This factor will be triggered when the option 'all of the above' is used.

Trigger Example:

```
Which of the following are nouns?
    a) car
    b) pencil
    c) cow
    d) all of the above
```

Dialogue:

You have made use of the option 'all of the above'. Be advised that its use is seldom indicated. Suggestion: try to reword the item so that it is not needed.

Notes:

The 'all of the above' in option d) will be highlighted.

Item Factor #7b: None of the Above

Usage: Multiple Choice

Specification:

This factor will be triggered when the option 'none of the above' is used.

Trigger Example:

```
Which of the following are verbs?
    a) car
    b) pencil
    c) cow
    d) none of the above
```

Dialogue:

You have made use of the option 'none of the above'. Although seldom useful, it does have applications. Suggestion: re-examine the item to assure that this is necessary.

Notes:

The 'none of the above' in option d) will be highlighted.

Item Factor #8: Repetition of Significant Words

Usage: Multiple Choice

Specification:

This factor will be triggered when any significant word from the stem is repeated in one or more of the options. Significance will be determined as any word, three or more letters long, which is not an article or incidental word.

Trigger Example:

```
The gas Helium is an example of a
     a) compound
     b) noble gas
     c) molecule
```

Dialogue:

The highlighted word appears in both the stem and the indicated options, thus providing a possible cue. Suggestion: reword either the stem or the affected options.

Notes:

The word 'gas' in the stem and option b) will be highlighted.

A.2.2 Test Factors

For each of the following test-wide analysis factors, it is necessary to identify some applicable subset of items upon which the factor operates. In most cases, this subset will be the totality of items of a given type (multiple choice or true/false). As will be discussed later, some items within a given subset may, for some reason, be discarded from consideration.

Test Factor #1: Randomness of Multiple Choice Answer Placement

Usage: All Multiple Choice Items

Specification:

This factor will be triggered when the placement of the answer in all of the multiple choice items does not display sufficient randomness.

Dialogue:

The correct options in the multiple choice items are note placed randomly. Suggestion: re-examine the multiple choice items to identify and correct biases in placement.

Test Factor #2: Balance of True/False Items

Usage: All True/False Items

Specification:

This factor will be triggered when there is a significant imbalance in the number of true and false true/false items.

Dialogue:

There is a significant imbalance between the number of true and false answers in true/false items. The current ration is (true-number):(false-number). Suggestion: review these items to bring the ratio into balance.

Test Factor #3: Length of True/False Items

Usage: All True/False Items

Specification:

This factor will be triggered when there are items which are significantly longer than the mean length.

Dialogue:

The indicated true/false item numbers were significantly longer/shorter than the rest, thus possibly cueing the testee that they are 'true'. Suggestion: reword these items to bring them closer to the mean length of (mean-length).

Appendix B

Subroutine Listing

The following is an alphabetized listing of all procedure and function names in ITAX followed by the page number in Appendix C where its associated documentation extract may be found.

Appendix C

Documentation Extract

Included in this appendix is the extracted header documentation for each module in ITAX. Appendix B provides an index and alphabetized listing of all modules.

```
************
Program Name: analysis
************

File Location: analysis.pas

Called By: None (this is the driver)

Calling: title
         item_analysis
         test_analysis

Defining: read_function_key
          title
          clear_dialogue
          item_analysis
          test_analysis

Including: readfkey.pas
           title.pas
           cleardia.pas
           cleari.pas
           swap.pas
           display.pas
           analyzei.pas
           editi.pas
           analyzet.pas

Parameters: None

Purpose: This is the driver for the system.

Constants: F1 thru F10- the ASCII codes for the 10 function keys.
           MC - the test constant for multiple choice.
           TF - the test constant for true/false.
           displaycolor - the color for the display portions of
                          the screen.
           errorcolor - the color for error message display.
           datacolor - the color for data display.

Types: function_set - type for a set of function key constants.
       stemline - type for one line of a MC stem.
```

```
        optionsline - type for one MC option.
        statementline - type for one line of a TF stem.
        stemtype - type for the stem of a MC item.
        optionstype - type for the options list of a MC item.
        statementtype - type for the statement of a TF item.
        item_type - the type for items as input by the editor.
            .itype - the type of item (MC or TF)
            .stem - the stem for a MC item
            .num_options - the number of options in a MC item
            .correct_option - the number of the correct option
            .statement - the statement for a TF item
            .answer - the answer for a TF item.
        TFdatatype - the type for the TF cumulative data variables.
        chicellstype - the type for the MC cumulative data variables.

Variables: test_mode - true if test analysis has been selected.
           force_exit - true if an exit is requested from
                        item analysis.
           TFdata - the cumulative variable for TF item data.
                    Positions 1-100 are the lengths of TF items,
                    Position 101 is the number of true items,
                    Position 102 is the number of false items,
                    Position 103 is the number of items included.
           chicells - the cumulative variable for MC item data.
```

```
************
Program Name: analysis_screen
************


File Location: analyzei.pas

Called By:  analyze_item

Calling: None

Defining: None

Parameters: which_type - the type of the item being analyzed

Purpose: This procedure changes the edit screen to the analysis
         screen

Constants: None

Types: None

Variables: None
```

```
************
Program Name: analyze1
************


File Location: analyzei.pas

Called By: analyze_item

Calling: clear_dialogue
         vowelstart
         reverse_list

Defining: None

Parameters: current_record - the current item

Purpose: This procedure performs the A/AN cue test

Constants: None

Types: None

Variables: i - a general loop counter
           a_message, an_message - flags to indicate that a detection
                                   message is needed
           lastword - the last word of a stem
```

```
************
Program Name: analyze2
************


File Location: analyzei.pas

Called By: analyze_item

Calling: clear_dialogue
         and_in
         end_s

Defining: and_in
          end_s

Parameters: current_record - the current item

Purpose: This procedure performs the Plural cue test

Constants: None

Types: None

Variables: i - a general loop counter
           alltrue, allfalse - flags for the plurality of all options
           plural - flags for the plurality of each option
```

```
************
Program Name: analyze3
************


File Location: analyzei.pas

Called By: analyze_item

Calling: clear_dialogue
         reverse_word

Defining: None

Parameters: current_record - the current item

Purpose: This procedure performs the Negativity test

Constants: None

Types: None

Variables: i - a general loop counter
           negative - the number of negatives located
           search_list - the list of negative words in the item
```

```
************
Program Name: analyze4
************


File Location: analyzei.pas

Called By: analyze_item

Calling: clear_dialogue
         reverse_word

Defining: None

Parameters: current_record - the current item

Purpose: This procedure performs the Repeated Phrases test

Constants: None

Types: None

Variables: i - a general loop counter
           word_position - the position within the item being scanned
           comp - the lines of the option
           no_match - flag for non detection of a repetition
           nilcount - the count of the number of nil options
```

```
************
Program Name: analyze5
************


File Location: analyzei.pas

Called By: analyze_item

Calling: clear_dialogue

Defining: None

Parameters: current_record - the current item

Purpose: This procedure performs the Too Few Options test

Constants: minimum_options - the minimum number of allowable options

Types: None

Variables: i - a general loop counter
           number_of_options - the number of options in the item
```

```
************
Program Name: analyze6
************


File Location: analyzei.pas

Called By: analyze_item

Calling: clear_dialogue
         wordlength

Defining: None

Parameters: current_record - the current item

Purpose: This procedure performs the Inconsistent Length of
         Distractors test

Constants: trigger_level - the triggering level for the Chebyshev
                           mean test

Types: None

Variables: number_of_options - the number of active options
           i - a general loop counter
           length_of_options - the lengths of each option
           sum_of_options - the sum of the lengths
           mean_length - the mean length of the options
           flag - the flag for detection of inconsistency
           std_dev - the standard deviation of the options
           square_sum - a step toward to calculation of the standard
                        deviation
```

```
************
Program Name: analyze7a
************


File Location: analyzei.pas

Called By: analyze_item

Calling: clear_dialogue
         reverse_list

Defining:None

Parameters: current_record - the current item

Purpose: This procedure performs the check for ALL-OF-THE-ABOVE

Constants: None

Types: None

Variables: i - a general loop counter
           wordcount - the number of words in an option
           allflag - not used
           flag - flag for the detection of the all-of-the-above
           recon_string - the reconstituted option line
           recon_list - the list of words to be reconstituted into
                        a line
```

```
************
Program Name: analyze7b
************


File Location: analyzei.pas

Called By: analyze_item

Calling: clear_dialogue
         reverse_list

Defining: None

Parameters: current_record - the current item

Purpose: This procedure performs the check for NONE-OF-THE-ABOVE

Constants: None

Types: None

Variables: i - a general loop counter
           wordcount - the number of words in an option
           allflag - not used
           flag - flag for the detection of the none-of-the-above
           recon_string - the reconstituted option line
           recon_list - the option list to be reconstituted into
                        a line
```

```
************
Program Name: analyze8
************


File Location: analyzei.pas

Called By: analyze_item

Calling: clear_dialogue
         make_sig_list
         search_options
         disposeit

Defining: make_sig_list
          search_options

Parameters: current_record - the current item

Purpose: This procedure performs the Repetition of Significant Words
         test

Constants: None

Types: None

Variables: i - a general loop counter
           flag - flag for the detection of repetition
           keynum - the key pressed by the user to go on in the
                    analysis
           stem_list - the head of the list of words in the stem
           option_list - the head of the list of words in the
                         options
```

```
************
Program Name: analyze_item
************


File Location: analyzei.pas

Called By: item_analysis

Calling: analysis_screen
         make_list
         read_function_key
         display_item
         analyze1
         analyze2
         analyze3
         analyze4
         analyze5
         analyze6
         analyze7a
         analyze7b
         analyze8
         save_data
         repair_screen
         dispose_all


Defining:  analysis_screen
           repair_screen
           make_list
           disposeit
           disposeall
           reverse_word
           reverse_list
           display_record
           vowelstart
           wordlength
           savedata
           analyze1
           analyze2
           analyze3
           analyze4
```

```
            analyze5
            analyze6
            analyze7a
            analyze7b
            analyze8

Parameters: current_item - the currently active item
             test_mode - set if test analysis is enabled
             TFdata - the accumulated TF data for test analysis
             chicells - the accumulated MC data for test analysis

Purpose: This procedure takes a completed item of either type and
         performs the analyses indicated

Constants: factor1-10 - the analysis factor constants
           factor9 - a factor constant needed for ranging
           max_factor - the greatest active factor

Types: factorset - a set of analysis factor constants
       wordpointer - a pointer to a word unit
       wordrecord - a word unit consisting of
           .word - the word itself
           .printline - its line location on the screen
           .printcolumn - its column beginning location on the screen
           .next - the next word in the line
       wordpointerarray - a set of 5 lines of words
       itemrecord - the record for an item consisting of
           .itype - the type of the record
           MC
             .stem - the stem of the MC item
             .options - the set of up to 5 options
             .correct - the correct option number
           TF
             .statement - the TF statement
             .truefalse - the answer to the TF item

Variables: valid_factors - the set of currently useful analysis
                           factors
           active - the currently active function key numbers
           action - the currently selected function key function
           current_factor - the currently executing factor
           done - the flag for analysis completion
```

```
error - the flag for an input error
current_record - the current item in record form
```

```
************
Program Name: and_in
************


File Location: analyzei.pas

Called By: analyze2

Calling: None

Defining: None

Parameters: the_option - the option to be scanned for the 'and'
            RETURNS - true if the word 'and' is in the option

Purpose: The function checks for the presence of 'and' in an option.

Constants: None

Types: None

Variables: return_value - the flag value to be returned
           list_head - the moving head of the list
```

```
************
Program Name: clear_dialogue
************


File Location: cleardia.pas

Called By: repair_screen
           analyze1
           analyze2
           analyze3
           analyze4
           analyze5
           analyze5
           analyze6
           analyze7a
           analyze7b
           analyze8
           test_analyze1
           test_analyze2
           test_analyze3

Calling: None

Defining: None

Parameters: None

Purpose: This procedure clears the dialogue area of any screen

Constants: None

Types: None

Variables: i - a general loop counter
```

```
************
Program Name: clear_item
************


File Location: cleari.pas

Called By: item_analysis
           swap_item_type

Calling: None

Defining: None

Parameters: current_item - the item variable to be cleared

Purpose: This procedure clears the item variable for new use

Constants: None

Types: None

Variables: i - a general loop counter
```

```
************
Program Name: clear_line
************


File Location: display.pas

Called By: display_item

Calling: None

Defining: None

Parameters: line - the line number to be cleared
            rowstart, rowend - the starting and ending rows to be
                               cleared

Purpose: This procedure clears the designated sequence of screen
         locations to the current background color.

Constants: None

Types: None

Variables: i - a general loop counter
```

```
************
Program Name: clipblanks
************


File Location: analyzei.pas

Called By: make_list

Calling: None

Defining: None

Parameters: clipstring - the string to be edited

Purpose: This procedure clips superfluous trailing blanks from
         a string

Constants: None

Types: None

Variables: cliplength - the length of the real portion of the
                        string
           clipstop - the index of the last real position in
                      the string
```

```
************
Program Name: command_processor
************


File Location: editi.pas

Called By: edit_stem
           edit_options
           edit_statement

Calling: wrap_check
         invalid_char

Defining: wrap_check


Parameters: next_char- the character entered at the keyboard
            insert_mode- the flag denoting the current state of
                         inserting (insert or overtype)
            curline, curcol- the current editing location within the
                             field
            number_of_lines- the number of lines in the editing field
            line_length- the length of the lines in the editing field
            edit_line- the current line being edited
            action- the command code to be returned to edit_item
            edit_field- the index of the field in the item being
                        edited
            exit_field- the flag denoting that the field has been left

Purpose: This procedure takes the command character from the keyboard
         interface routine (get_char) and decodes it meaning.  If it
         is a command character, it set it up for return to edit_item.
         If it is an editing control code, the appropriate action is
         taken.

Constants: None

Types: None

Variables: coloffset, lineoffset- the true location of the cursor on
                                  the screen
```

****
Program Name: display_MC
****

File Location: swapi.pas

Called By: swap_item_type

Calling: None

Defining: None

Parameters: None

Purpose: This procedure displays the MC edit/analysis format

Constants: None

Types: None

Variables: i - a general loop counter

```
************
Program Name: display_TF
************


File Location: swapi.pas

Called By: swap_item_type

Calling: None

Defining: None

Parameters: None

Purpose: This procedure displays the TF edit/analysis format

Constants: None

Types: None

Variables: i - a general loop counter
```

```
************
Program Name: display_item
************


File Location: display.pas

Called By: analyze_item
           edit_item

Calling: clear_line

Defining: clear_line

Parameters: current_item - the item to be displayed

Purpose: This procedure displays the currently active item according
         to its type

Constants: None

Types: None

Variables: i - a general loop counter
```

```
************
Program Name: display_record
************


File Location: analyzei.pas

Called By: Utility Routine

Calling: None

Defining: None

Parameters: current_record - the record to be displayed

Purpose: This is a utility routine for displaying the contents of a
         record

Constants: None

Types: None

Variables: i - a general loop counter
```

****
Program Name: dispose_all
****

File Location: analyzei.pas

Called By: analyze_item

Calling: disposeit

Defining: None

Parameters: current_record - the record to be disposed

Purpose: This procedure disposes of all elements of a record

Constants: None

Types: None

Variables: i - a general loop counter

```
************
Program Name: disposeit
************


File Location: analyzei.pas

Called By: analyze_item
           analyze8

Calling: None

Defining: None

Parameters: list_head - the list to be disposed

Purpose: This procedure disposes the atoms of a list

Constants: None

Types: None

Variables: next_dispose - the next atom to be disposed
```

```
************
Program Name: edit_MC
************


File Location: editi.pas

Called By: edit_item

Calling: edit_stem
         edit_options
         edit_MC_answer

Defining: edit_stem
          edit_options
          edit_MC_answer

Parameters: current_item- the multiple choice item to be edited
            action- a returned command code for edit_item
            active- a set of currently active function keys on the
                    menu

Purpose: This procedure is responsible for the input/editing of
         multiple choice items.  Separate routines handle each of
         the three fields of the item: stem, options and answer.

Constants: stemfield- the index for the first item field
           optionsfield- the index for the second item field
           answerfield- the index for the third item field

Types: None

Variables: edit_field- the currently active field
           valid_action- flag denoting that a valid menu choice has
                         been made
           insert_mode- flag denoting the currently active insertion
                        mode
           esc_char- the escape code for the next character typed
```

```
************
Program Name: edit_MC_answer
************


File Location: editi.pas

Called By: edit_MC

Calling: get_char
         invalid_char

Defining: None

Parameters: correct_answer - the number of the correct option
            action - the next menu command to be passed to
                     item_analysis
            active- the currently active menu function keys
            edit_field- the next MC field to be edited

Purpose: This procedure gets the correct option number by allowing
         the user to move the "CORRECT >" marker from option to
         option in the list.

Constants: None

Types: None

Variables: exit_field- flag denoting that editing is complete
           next_char- the next character entered from the keyboard
           esc_char- the escape code entered at the keyboard
```

```
************
Program Name: edit_TF
************


File Location: editi.pas

Called By: edit_item

Calling: edit_statement
         edit_TF_answer

Defining: edit_statement
          edit_TF_answer

Parameters: current_item- the true/false item to be edited
            action- a returned command code for edit_item
            active- a set of currently active function keys on the
                    menu

Purpose: This procedure is responsible for the input/editing of true/
         false items.  Separate routines handle each of the two
         fields of the item: statement and answer.

Constants: statementfield- the index for the first item field
           answerfield- the index for the third item field

Types: None

Variables: edit_field- the currently active field
           valid_action- flag denoting that a valid menu choice has
                         been made
           insert_mode- flag denoting the currently active insertion
                        mode
```

```
************
Program Name: edit_TF_answer
************


File Location: editi.pas

Called By: edit_TF

Calling: get_char
         invalid_char

Defining: None

Parameters: answer- the answer portion of the true/false item
            action- the next menu function to be passed to item_anal
            active- the set of active function keys
            edit_field- the next TF field to be considered

Purpose: This procedure handles the input and editing of the answer

Constants: None

Types: None

Variables: exit_field - a flag denoting that editing is complete
           next_char, esc_char - the next character typed at the
                                 keyboard
```

```
************
Program Name: edit_item
************


File Location: editi.pas

Called By: item_analysis

Calling: display_item

Defining: None

Parameters: current_item - the currently active item
            active - the set of active function key numbers
            action - the function key action to be taken

Purpose: This procedure performs the entry and editing of an item

Constants: ESC - the ASCII code for the escape key
           BS - the ASCII code for the backspace key
           RTN - the ASCII code for the return key
           RARROW - the ASCII code for the right arrow key
           LARROW - the ASCII code for the left arrow key
           UARROW - the ASCII code for the up arrow key
           DARROW - the ASCII code for the down arrow key
           INS - the ASCII code for the insert key
           DEL - the ASCII code for the delete key

Types: None

Variables: None
```

```
************
Program Name: edit_options
************


File Location: editi.pas

Called By: edit_MC

Calling: get_char
         command_processor
         process_character

Defining: None

Parameters: options- the options list for a multiple choice item
            action- a returned command code
            active- the set of currently active function keys
            edit_field- the index of the next field to be edited
            insert_mode- the currently active insertion mode

Purpose: This procedure handles the editing of multiple choice item
         options lists.

Constants: line_length- the length of an options line
           number_of_lines- the number of multiple choice options in
                            the list

Types: None

Variables: exit_field- a flag denoting that the field is to be exited
           curline, curcol- the current location within the field
           next_char- the next character entered at the keyboard
           esc_char- the escape code for the next character
```

```
************
Program Name: edit_statement
************


File Location: editi.pas

Called By: edit_MC

Calling: get_char
         command_processor
         process_character

Defining: None

Parameters: statement- the statement portion of the true/false item
            action- the next menu function to be passed to item_anal
            active- the set of active function keys
            edit_field- the next TF field to be considered
            insert_mode- the currently active insertion mode

Purpose: This procedure handles the input and editing of the statement
         portion of a true/false item.

Constants: line_length- the length of a line in a statement
           number_of_lines- the number of lines in a statement

Types: None

Variables: exit_field- a flag denoting that editing is complete
           curline, curcol- the current position in the statement
                            field
           next_char- the next character typed at the keyboard
           esc_char- the escape code for the next character typed
```

```
************
Program Name: edit_stem
************


File Location: editi.pas

Called By: edit_TF

Calling: get_stem
         command_processor
         process_character

Defining: None

Parameters: stem- the MC stem to be edited
            action- a returned command code
            active- the set of active function keys on the menu
            edit_field- the index of the next field to be edited
            insert_mode- the currently active insertion mode

Purpose: This procedure handles entry and editing of multiple choice
         stems

Constants: line_length- the length of a stem line
           number_of_lines- the number of stem lines

Types: None

Variables: exit_field- a flag denoting that the field is to be exited
           curline, curcol- the current position within the field
           next_char- the next character entered at the keyboard
           esc_char- the escape code for the next character
           next_char- the next character typed at the keyboard
```

```
************
Program Name: end_s
************


File Location: analyzei.pas

Called By: analyze2

Calling: None

Defining: None

Parameters: the_option - the option to be checked for words which end
                         in 's'
            RETURNS - true if the option has terms which end in 's'

Purpose: This function checks for the presence of words ending in 's'
         in an option

Constants: None

Types: None

Variables: list_head - the moving head of the list
```

```
************
Program Name: get_char
************


File Location: editi.pas

Called By: edit_stem
           edit_options
           edit_MC_answer
           edit_statement
           edit_TF_answer

Calling: None

Defining: None

Parameters: next_char - the next character from the keyboard
            esc_char - the escape code for the kety pressed

Purpose: This procedure gets the key code from the next character
         pressed at the keyboard

Constants: None

Types: None

Variables: char_entered - flag denoting the striking of a key
           truex, truey - the current location of the cursor on the
                          screen
```

```
************
Program Name: get_next_word
************


File Location: analyzei.pas

Called By: line_to_list

Calling: None

Defining: None

Parameters: the_line - the line source for the word
            the_word - the word accessed
            position - the index position of the word in the line
            next_index - the index position for the next word to be
                         accessed

Purpose: This procedure gets the next word from a line and rectifies
         the index position of that word in the line

Constants: None

Types: None

Variables: end_word - the ending index of the word
           alphanum - the set of acceptable word characters
           enders - the set of word delimiters
           stringlength - the length of the current line
           start_word - the start index for the word in the line
           done - the completion flag
```

```
************
Program Name: invalid_char
************


File Location: editi.pas

Called By: command_processor
           edit_MC_answer
           edit_TF_answer

Calling: None

Defining: None

Parameters: next_char - the character pressed in error
            esc_char - the escape code of the invalid character

Purpose: This is the handler for the entry of an inappropriate
         character

Constants: None

Types: None

Variables: oldline, oldcol - the position of the cursor before
                             displaying the error message
```

```
************
Program Name: item_analysis
************


File Location: analysis.pas

Called By: analysis

Calling: swap_item_type
         edit_item
         analyze_item
         clear_item

Defining: clear_item
          swap_item_type
          display_item
          analyze_item
          edit_item
          get_char
          invalid_char
          command_processor
          process_character
          edit_MC
          edit_TF

Parameters: test_mode - set if the system is enables for test
                        analysis
            force_exit - set if the user chose to exit from the
                         item analysis subsystem
            TFdata - the cumulative TF item data
            chicells - the cumulative MC item data

Purpose: This procedure directs the operation of the item edit and
         analysis subsystem

Constants: blank - the space character

Types: full_line - the maximum length of a screen input line

Variables: current_item - the currently active item
           action - the currently running function key function
```

```
error - the flag for keystroke error
active - the set of currently useful function keys
```

```
************
Program Name: join_lists
************


File Location: analyzei.pas

Called By: make_list

Calling: None

Defining: None

Parameters: the_line - the second line to be made a list
            line_number - the number position of the line
            add_list - the list of words to be added to this new list
            lineoffset - the line position offset of the line
            coloffset - the column position offset of the line

Purpose: This procedure is used in creating a single list out of the
         5 lines in a stem or statement

Constants: None

Types: None

Variables: add_position - the adding position in the list
           new_list - the newly generated word list
           adder - the list to be added
```

```
************
Program Name: line_to_list
************


File Location: analyzei.pas

Called By: make_list

Calling: None

Defining: None

Parameters: the_line - the line to be transformed into a list
            line_number - the number of the current line
            lineoffset - the line position offset for the line
            coloffset - the column position offset for the line
            RETURNS: a pointer to the head of the list

Purpose: This function takes a textual line and creates a linked list
         of words

Constants: None

Types: None

Variables: new_atom - a new word atom for the list
           line_to_list_holder - a temp holder for the head of the
                                 list
           current_tail - pointer to the current tail of the list
           actual_line - the actual screen line position
           actual_column - the actual screen column position
           position - the position index within the line
           new_index - the generated new  index into the line
           current_offset - the current printing offset within the
                            line
           the_word - a returned word from the line
```

```
************
Program Name: make_list
************


File Location: analyzei.pas

Called By: analyze_item

Calling: clipblanks
         line_to_list
         join_lists

Defining: None

Parameters: current_item - the current item in editing format

Purpose: This procedure changes the edit format item to the analysis
         formatted item

Constants: mcstemlineoffset - the printing offset for the mc stem
           mcstemcoloffset - the printing column offset for the mc
                             stem
           mcoptionslineoffset - the line offset for the mc options
           mcoptionscoloffset - the column offset for the mc options
           tflineoffset - the line offset for the TF statement
           tfcoloffset - the column offset for the TF statement

Types: None

Variables: i - a general loop counter
```

```
************
Program Name: make_sig_list
************


File Location: analyzei.pas

Called By: analyze8

Calling: None

Defining: None

Parameters: this_list - the list to be processed
            old_list - the existing list of significant words to be
                       added
            RETURNS - the head of a list of significant words

Purpose: This function funds the significant words in a list and adds
         them to the cumulative list

Constants: max_dictionary - the number of insignificant words in the
                            dictionary

Types: None

Variables: dictionary - the insignificant words
           source_list - the list to be scanned
           significant - set of the words qualifies as significant
           i - a general loop counter
```

```
************
Program Name: process_character
************


File Location: editi.pas

Called By: edit_stem
           edit_options
           edit_statement

Calling: None

Defining: None

Parameters: next_char- the character to be inserted
            edit_line- the line the character is to be inserted into
            exit_field- not currently used.
            insert_mode- the current insertion mode (overtype or
                         insert)
            curcol- the active position within the edit line
            line_length- the length of the currently active edit line

Purpose: This procedure adds the entered data character to the
         currently active edit field according to the insert_mode

Constants: None

Types: None

Variables: trueline, truecol- the actual location of the cursor on
                              the screen
```

```
************
Program Name: read_function_key
************


File Location: readfkey.pas

Called By: test_analysis
           analyze_item

Calling: None

Defining: None

Parameters: active - the set of active function keys
            keynum - the function key number that was pressed
            error - set if an inactive function key was selected

Purpose: This procedure gets the user's selection of a function key
         and checks it against the set of currently active
         function keys.

Constants: escape - the ASCII code for the escape character

Types: None

Variables: None
```

****
Program Name: repair_screen
****

File Location: analyzei.pas

Called By: analyze_item

Calling: clear_dialogue

Defining: None

Parameters: which_type - the type of item on the current screen

Purpose: This procedure resets the analysis screen to the edit screen

Constants: None

Types: None

Variables: None

```
************
Program Name: reverse_list
************


File Location: analyzei.pas

Called By: analyze1
           analyze7a
           analyze7b

Calling: clear_dialogue

Defining: None

Parameters: list_head - the head of a list of words to be reversed

Purpose: This procedure displays all of the words in the list in the
         error color

Constants: None

Types: None

Variables: reverse_head - the current position in the list
```

```
************
Program Name: reverse_word
************


File Location: analyzei.pas

Called By:  reverse_list
            analyze3
            analyze4

Calling: None

Defining: None

Parameters: reverse_me - the word to be highlighted on the screen

Purpose: This procedure displays the indicated word in the error
         color

Constants: None

Types: None

Variables: None
```

```
************
Program Name: save_data
************


File Location: analyzei.pas

Called By: analyze_item

Calling: None

Defining: None

Parameters: current_record - the currently active item record
            TFdata - the cumulative TF data
            chicells - the cumulative MC data

Purpose: This procedure updates the saved data for test analysis to
reflect the contribution of the current item

Constants: None

Types: None

Variables: i - a general loop counter
           count - a counter for the number of options
```

```
************
Program Name: search_options
************


File Location: analyzei.pas

Called By: analyze8

Calling:  None

Defining: None

Parameters: stem_list - a list of words which are significant in the
                        stem
            option_list - a list of words which are significant in
                          the options

Purpose: This function scans the options list of significant words
         to check for matches in the stem list.  Duplications are
         reversed.

Constants: None

Types: None

Variables: search_list - the list to be searched
           flag - for the detection of a duplication
```

```
************
Program Name: swap_item_type
************


File Location: swap.pas

Called By: item_analysis

Calling: clear_item
         display_MC
         display_TF

Defining: display_MC
          display_TF

Parameters: current_item - the currently active item and type
            test_mode - is set if test analysis is enabled
            active - the set of currently active function key numbers

Purpose: This procedure changes the displayed item type from MC to
         TF or back again during editing.

Constants: None

Types: None

Variables: None
```

```
************
Program Name: test_analysis
************


File Location: analysis.pas

Called By: analysis

Calling: test_analysis_screen
         read_function_key
         test_analyze1
         test_analyze2
         test_analyze3

Defining: test_analysis_screen
          test_analyze1
          test_analyze2
          test-analyze3

Parameters: TFdata - the cumulative data from the TF items
            chicells - the cumulative data from the MC items

Purpose: This procedure directs the test analysis portion of
         the system

Constants: factor1-4 - the constants for the analysis factors
           max_factor - the maximum factor number

Types: factorset - a set of factor constants

Variables: valid_factors - the set of currently useful analysis
                           factors
           active - the currently active function keys
           done - flag for completion
           error - flag for detection of keying error
           action - the currently active function key action
           current_factor - the currently active analysis factor
```

```
************
Program Name: test_analysis_screen
************


File Location: analyzet.pas

Called By: test_analysis

Calling: None

Defining: None

Parameters: TFdata - the cumulative TF item data
            chicells - the cumulative MC data
            valid_factors - the valid factor numbers to be considered

Purpose: This procedure sets up the screen for test analysis and
         sets the active analysis factors

Constants: None

Types: None

Variables: i, j - general loop counters
           mcgo - flag for the analysis of MC items
```

```
************
Program Name: test_analyze1
************


File Location: analyzet.pas

Called By: test_analysis

Calling: None

Defining: None

Parameters: chicells - the cumulative MC data

Purpose: This procedure performs the MC randomness test

Constants: None

Types: None

Variables: partial - the partial chi-squared value for a cell
           cell - the cell number being considered
           expected - the expected frequency for each cell
           chipartials - the calculated position of the chi-square
                         value
           chisqu - the calculated chi-squared value
           degrees_of_freedom - the degrees of freedom for the
                                chi-square test
           flag - flag for a factor detection
           chisqu_critical - the critical values of the chi-square
                             statistic
           column - the column of the current print location
```

```
************
Program Name: test_analyze2
************


File Location: analyzet.pas

Called By: test_analysis

Calling: None

Defining: None

Parameters: TFdata - the cumulative TF item data

Purpose: This procedure performs the TF balance of true and false
         answers test

Constants: chisquare_critical - the critical value for the chi-square
                                statistic

Types: None

Variables: true_questions - the number of true items
           false_questions - the number of false items
           total_questions - the total number of TF items
           chisqu - the calculated chisquare value
           expected - the expected frequency
           flag - flag for detection of the factor
```

```
************
Program Name: test_analyze3
************


File Location: analyzet.pas

Called By: test_analysis

Calling: None

Defining: None

Parameters: TFdata - the TF item data

Purpose: This procedure performs the TF inconsistent length test

Constants: tolerance - the 50% tolerance level for the Chebyshev
                       inequality

Types: None

Variables: i - a general loop counter
           n - the number of TF items
           sum - the sum of the lengths
           sumsqu - the sum of the squared lengths
           check_one - the item being checked
           flag - flag for the detection of the factor
           TFdataflag - detection flags for each item
           sequential - the next item counter
           row, column - the row/column print position of the item
                         length
```

```
************
Program Name: title
************


File Location: title.pas

Called By: analysis

Calling: None

Defining: None

Parameters: test_mode - is set if the user selects to perform a test
                        analysis at the end of the session.

Purpose: This procedure displays the system title screen and polls
         the user for a selection of operating mode

Constants: None

Types: None

Variables: keynumber - the function key number selected
           error - is set if an inactive function key was selected
```

```
************
Program Name: vowelstart
************


File Location: analyzei.pas

Called By: analyze1

Calling: None

Defining: None

Parameters: startline - a line to be scanned
            RETURNS - a true is the line starts with a vowel

Purpose: This function scans a line to see if it starts with a vowel

Constants: None

Types: None

Variables: None
```

```
************
Program Name: wordlength
************


File Location: analyzei.pas

Called By: analyze6

Calling: None

Defining: None

Parameters: the_list - the pointer to a list of words
            RETURNS - the number of words in the line

Purpose: This function counts the number of words in a line

Constants: None

Types: None

Variables: wlength - the cumulative length of the line
           count_head - the indexed head of the list
```

```
************
Program Name: wrap_check
************


File Location: editi.pas

Called By: command_processor

Calling: None

Defining: None

Parameters: curcol, curline- the current position in the edit field
            number_of_lines- the number of lines in the currently
                             active edit field
            line_length- the length of lines in the current edit field
            edit_field- the index number of the currently active edit
                        field
            exit_field- the code signifying that the field has been
                        left.

Purpose: This procedure checks to see if a line wrap has occurred.
         If so, the cursor is moved following the protocol:
              off the right  -  to the beginning of the next line
              off the left   -  to the end of the previous line
              off the top    -  to the previous line
              off the bottom -  to the next line
         If the cursor moves out of the field, exit_field will
         contain the field index number to be edited next

Constants: None

Types: None

Variables: None
```

Appendix D

Procedure/Function Declaration Structure

```
ANALYSIS
    READ_FUNCTION_KEY
    TITLE
    CLEAR_DIALOGUE
    ITEM_ANALYSIS
        CLEAR_ITEM
        SWAP_ITEM_TYPE
            DISPLAY_TF
            DISPLAY_MC
        DISPLAY_ITEM
            CLEAR_LINE
        ANALYZE_ITEM
            ANALYSIS_SCREEN
            REPAIR_SCREEN
            MAKE_LIST
                CLIPBLANKS
                GET_NEXT_WORD
                LINE_TO_LIST
                JOIN_LISTS
            DISPOSEIT
            DISPOSEALL
            REVERSE_WORD
            REVERSE_LIST
            DISPLAY_RECORD
            VOWELSTART
            WORDLENGTH
            SAVEDATA
```

ANALYZE1
ANALYZE2
AND_IN
END_S
ANALYZE3
ANALYZE4
ANALYZE5
ANALYZE6
ANALYZE7A
ANALYZE7B
ANALYZE8
MAKE_SIG_LIST
SEARCH_OPTIONS
EDIT_ITEM
GET_CHAR
INVALID_CHAR
COMMAND_PROCESSOR
WRAP_CHECK
PROCESS_CHARACTER
EDIT_MC
EDIT_STEM
EDIT_OPTIONS
EDIT_MC_ANSWER
EDIT_TF
EDIT_STATEMENT
EDIT_TF_ANSWER
TEST_ANALYSIS
TEST_ANALYSIS_SCREEN
TEST_ANALYZE1
TEST_ANALYZE2
TEST_ANALYZE3

Appendix E

Procedure/Function Calling Structure

ANALYSIS
- TITLE
- ITEM_ANALYSIS
- TEST_ANALYSIS

READ_FUNCTION_KEY
- *none

TITLE
- *none

CLEAR_DIALOGUE
- *none

ITEM_ANALYSIS
- SWAP_ITEM_TYPE
- EDIT_ITEM
- ANALYZE_ITEM
- CLEAR_ITEM

TEST_ANALYSIS
- TEST_ANALYSIS_SCREEN
- READ_FUNCTION_KEY
- TEST_ANALYZE1
- TEST_ANALYZE2
- TEST_ANALYZE3

CLEAR_ITEM
*none

SWAP_ITEM_TYPE
CLEAR_ITEM
DISPLAY_MC
DISPLAY_TF

DISPLAY_MC
*none

DISPLAY_TF
*none

DISPLAY_ITEM
CLEAR_LINE

CLEAR_LINE
*none

ANALYZE_ITEM
ANALYSIS_SCREEN
MAKE_LIST
READ_FUNCTION_KEY
DISPLAY_ITEM
ANALYZE1
ANALYZE2
ANALYZE3
ANALYZE4
ANALYZE5
ANALYZE6
ANALYZE7A
ANALYZE7B
ANALYZE8
SAVE_DATA
REPAIR_SCREEN
DISPOSE_ALL

ANALYSIS_SCREEN
*none

REPAIR_SCREEN
    CLEAR_DIALOGUE

MAKE_LIST
    CLIPBLANKS
    LINE_TO_LIST
    JOIN_LISTS

CLIPBLANKS
    *none

GET_NEXT_WORD
    *none

LINE_TO_LIST
    *none

JOIN_LISTS
    *none

DISPOSEIT
    *none

DISPOSE_ALL
    DISPOSEIT

REVERSE_WORD
    *none

REVERSE_LIST
    REVERSE_WORD

DISPLAY_RECORD
    *none

VOWELSTART
    *none

WORDLENGTH
    *none

SAVE_DATA
*none

ANALYZE1
CLEAR_DIALOGUE
VOWELSTART
REVERSE_LIST

ANALYZE2
CLEAR_DIALOGUE
AND_IN
END_S

AND_IN
*none

END_S
*none

ANALYZE3
CLEAR_DIALOGUE
REVERSE_WORD

ANALYZE4
CLEAR_DIALOGUE
REVERSE_WORD

ANALYZE5
CLEAR_DIALOGUE

ANALYZE6
CLEAR_DIALOGUE
WORDLENGTH

ANALYZE7A
CLEAR_DIALOGUE
REVERSE_LIST

ANALYZE7B
CLEAR_DIALOGUE
REVERSE_LIST

ANALYZE8

CLEAR_DIALOGUE
MAKE_SIG_LIST
SEARCH_OPTIONS
DISPOSEIT

EDIT_ITEM

DISPLAY_ITEM
EDIT_MC
EDIT_TF

GET_CHAR

*none

INVALID_CHAR

*none

COMMAND_PROCESSOR

WRAP_CHECK
INVALID_CHAR

WRAP_CHECK

*none

PROCESS_CHARACTER

*none

EDIT_MC

EDIT_STEM
EDIT_OPTIONS
EDIT_MC_ANSWER

EDIT_STEM

GET_CHAR
COMMAND_PROCESSOR
PROCESS_CHARACTER

EDIT_OPTIONS

GET_CHAR
COMMAND_PROCESSOR
PROCESS_CHARACTER

EDIT_MC_ANSWER
    GET_CHAR
    INVALID_CHAR

EDIT_TF
    EDIT_STATEMENT
    EDIT_TF_ANSWER

EDIT_STATEMENT
    GET_CHAR
    COMMAND_PROCESSOR
    PROCESS_CHARACTER

EDIT_TF_ANSWER
    GET_CHAR
    INVALID_CHAR

TEST_ANALYSIS_SCREEN
    *none

TEST_ANALYZE1
    *none

TEST_ANALYZE2
    *none

TEST_ANALYZE3
    *none

The vita has been removed from the scanned document