Design of a Microcomputer-Based Adaptive Testing System

C. David Vale
Assessment Systems Corporation

Research leading toward the implementation of adaptive testing has been actively underway for over 10 years. Initially, this research focused on developing and evaluating psychometric models and item selection strategies applicable to the adaptive testing process. Now that much of the basic research has been done, work on adaptive testing is moving in two directions. One direction emphasizes the development of increasingly sophisticated latent trait models; the other is directed toward the engineering problems of making the implementation of adaptive testing feasible.

Although the need for a computer system to support adaptive testing has been recognized from the beginning, relatively little emphasis has been placed on the development of a turnkey testing system. The computer programming required to develop, administer, and analyze an adaptive test represents a considerable investment in time and effort. Although a few computerized testing systems have been developed, they have been either small special purpose systems designed to administer and score a single test or large general purpose research systems. A general purpose self-contained adaptive testing system has not yet been developed.

Until recently, a major obstacle to development of such a system was a lack of sufficient demand for such a system to warrant anyone committing the resources necessary for its development and support. A capable computerized adaptive testing (CAT) system is usually somewhat dependent on the computer hardware, and to develop systems for all of the computers in use by adaptive testers would have been very impractical.

The availability of microcomputer systems and components have changed this, however. Capable microcomputers are sufficiently inexpensive that they can be dedicated exclusively to testing. It is now possible to design a testing system around a single computer and still have a system that is cost effective for testing.

The project described in this paper was sponsored by the Office of Naval Research. Its goal was to explore the feasibility of developing a single-user microcomputer-based testing system by (1) considering the needs of the potential consumers, (2) evaluating the available hardware, and (3) designing software that would run on available hardware and meet the needs of the consumers.

Evaluating the Needs of the Consumers

The evaluation of user needs proceeded along two lines. In the first approach, characteristics of current psychological tests were considered. Since the basic element of a test is the test item, a survey of the testing literature was done to discover what types of items might be used in the system. Furthermore, since a testing strategy is important to an adaptive test, the literature was also reviewed to compile a list of all of the strategies that might be used in such a system. In the second approach, potential users were surveyed. Several were interviewed, and a questionnaire was developed and distributed to determine what features they would like to see in the testing system.

Item Types

Five types of items were identified in the literature. The most basic was the knowledge item. A knowledge item is a question that seeks to determine whether the examinee possesses a knowledge of something. The most common of this type of item is the dichotomously scored multiple-choice item. Although some of these items require graphic displays, processing the responses requires only that the response given be matched to a small number of possibilities (e.g., five). Variations of the basic knowledge item include probabilistic response items (deFinneti, 1965), answer-until-correct items (Hanna, 1975), confidence-weighted items (Shuford, Albert, & Massengill, 1966), and free-response items (Vale, 1978). Items of these latter types require somewhat more complex processing of the responses.

The second type, the cognitive process item, assesses whether an examinee possesses a skill without, in theory, requiring the examinee to possess any knowledge. Examples of this type of item were contained in Cory's (1977, 1978) Memory for Objects Test. Items in this test tachistoscopically presented frames containing four to nine objects. The examinee's task was either to recall and to type the names of the objects or to recognize the objects from a word list. Administratively, the cognitive process items add a requirement to precisely time the stimulus; the remaining requirements are similar to those of the knowledge items.

Perceptual motor items comprised the third item type. An example of a perceptual motor item is the computerized version of the Two-Hand Coordination Test described by Hunter (1978). This test requires the examinee to move a cursor to follow a target as it moved around the screen. Items of this type add two additional requirements. First, sufficient processing power is required to manipulate a moving stimulus. Second, an analog input device, such a joystick, may be required.

A fourth type of item was the simulation. Simulations can be as complex as a high fidelity aircraft simulator. Such simulations must be considered beyond the current scope of a microcomputer-based testing machine. Less complex simulations, such as the medical simulation described by Prestwood (1980), are within the capabilities of current microcomputer systems, however. In the simulation described by Prestwood, medical residents were allowed to find symptoms, to administer treatments, and to cure (or kill) patients. This process was implemented as a branched sequence of scenarios with multiple-choice responses. Administratively, items such as this require no more system resources than knowledge items. To be easily implementable, they will require an authoring system that allows the branching among items to be easily specified.

The final item type consisted of noncognitive items. This type includes everything that does not assess a cognitive ability and includes personality items, interest items, and value items. Administratively, these items require no capabilities other than those required by the previous item types.

Testing Models

Most of the tests administered in paper-and-pencil form are conventional linear tests that administer a fixed sequence of items to all examinees. Most strategies of interest for a microcomputer-based testing system are adaptive. Vale (1981) grouped most of the adaptive strategies of interest into three categories: interitem branching strategies, intersubtest branching strategies, and model-based branching strategies.

The interitem branching strategies are implemented by structuring an item pool so that testing begins with one item, and each response to each item leads to the administration of a specific item. In the typical interitem branching strategy, a correct response to one item leads to a more difficult item and an incorrect response leads to a less difficult item. Examples of interitem branching strategies are the pyramidal and Robbins-Monro strategies (Weiss, 1974).

The intersubtest branching strategies are similar in concept to the interitem branching strategies except that the branching is from subtest to subtest rather than from item to item, a subtest being simply a group of items. Vale (1981) further divided this class of strategies into reentrant and nonreentrant forms. A reentrant form is one in which administration can branch to a subtest, out of that subtest to another subtest, and back into the original subtest. A nonreentrant form does not allow a subtest to be exited and then reentered. The two-stage test (Angoff & Huddleston, 1958) is one example of a nonreentrant strategy. Examples of reentrant strategies are Lord's (1971) flexilevel strategy and Weiss's (Vale & Weiss, 1978) stradaptive strategy.

The model-based strategies usually build on item response theory (IRT) and select items and score responses to those items in order to optimize the testing process according to the model. Procedurally, they start with an estimate of ability, select the item that is expected to most refine that estimate of ability, refine the estimate using the response to the item, and repeat the process until the estimate is adequately refined. One popular form of this type of strategy is Owen's (1975) Bayesian strategy. This strategy starts with a Bayesian prior distribution representing an initial estimate of ability. An item is selected that is expected to minimize the variance of the posterior distribution. This item is administered and used to produce the posterior distribution. A popular variant of this strategy selects items based on statistical information. This accomplishes nearly the same result but requires less computation.

For the interitem and intersubtest branching models, the difficult task in system design is to provide a convenient method of specifying the branching algorithms. For the model-based strategies, the greatest challenge is in selecting and designing a set of statistical programs that will allow the testing strategies to be implemented efficiently.

A Survey of Consumers

Twenty-seven individuals from seven different organizations were interviewed to gain further insight into the needs of the tester. In these interviews each individual was given a brief description of what was being developed and was then asked to suggest what features would be particularly useful to his/her application and to suggest what other features not included in the description would be useful. From these interviews, as well as from the literature reviewed, a questionnaire was developed to determine what features potential users would like to see in a microcomputer testing system and what features their tests would require. The questionnaire was divided into four sections. The first asked about characteristics of current tests and tests that might not be implemented on the proposed system. The second section addressed characteristics of the test administration process. The third asked what features would be desirable on a system, considered in light of the cost. The final section asked how much computer experience the respondent had.

Questionnaires were sent to 108 individuals. These individuals were selected for having attended conferences on computerized testing or having written articles in the area. Fifty-five questionnaires were returned, fifty with useful data (the other five individuals declined to respond.)

The test characteristics section of the questionnaire addressed four issues: the size of the item pools, the types of items, the testing strategies, and the types of scoring. Analyses of the questionnaire suggested that to accommodate item pools sufficiently large to satisfy the needs of 75% of the respondents, the system would have to store approximately one million characters just for the items. To accommodate this percentage of respondents, the system would also have to provide a way of timing the item and would have to allow graphics of at least the line-drawing variety. Almost all respondents said that they would administer conventional tests on the system, 68% said they would use model-based adaptive procedures, and 36% said that they would use interitem or intersubtest branching strategies. Seventy percent said that they would use some form of IRT scoring.

The test administration section also addressed four issues: what population would take the tests, whether the administrations would be proctored, what potential problems the system should monitor, and how much abuse the system was likely to get. Although 26% said that the system would be used for a grade-school populations, the most use was anticipated for high school (77%) and college (65%) groups. Eighty-eight percent of the respondents said that a proctor would always be available, 12% said that a proctor would usually be available, and only one respondent said that a proctor would not be available. Most of the respondents felt that the system should be able to detect unreasonably long response delays, random responding, and aberrant responding. On the issue of abuse, most felt that given the oportunity, examinees might push extra buttons and might spill things on the testing terminal; approximately one fourth thought that the examinees might be unnecessarily rough or might inflict deliberate damage on the system.

The system procurement section asked what optional features the respondents would like for the system. More than half of the respondents said that they would like a display capable of black-and-white line drawings, a videodisc or videotape interface, a simplified keybord, and a language (e.g., FORTRAN) compiler.

The final section focused on the computer skills of the potential test developer. Of the 49 individuals who responded to this section, all but two had run computer programs and all but three had written them. Ninety-two percent had written computer programs in a language like BASIC, Pascal, or FORTRAN. When asked whether they would like to develop tests in an author language or using a menu system, one third preferred the author language, one third preferred the menu system, and one third had no preference.

Design of a System to Meet User Needs

The design of a system to meet the needs of the users consisted of (1) selection of the hardware and (2) design of the software. The hardware search has been described by Vale, Albing, Foote-Lennox, and Foote-Lennox (1982) and will not be discussed here, with the exception of two points. First, at the time of the study there were a variety of microcomputers available that were adequately suitable. Second, the typical deficiency that prevented a system from being acceptable was the lack of sufficient disc storage for the item banks.

Existing Software

It is the software that makes a testing system different from any other microcomputer application. Work on the design of this part of the system began with a review of existing testing systems. Unfortunately, very few adaptive testing systems existed, and even fewer of those were documented in the literature. The most relevant system documented in the literature was the TCL system (Vale, 1981). This system was a self-contained test specification and administration system designed for a small single-user microcomputer. The system supported three testing functions: item banking, test development, and test administration. The item banking system was rudimentary; it simply read items from a sequential source file that could be edited by the system text editor and inserted them into a randomly accessible item bank. Tests were specified using an author language: a programming language developed especially to specify adaptive test structures. A test compiler translated the language into an easily executable form, collected the items required by the test, and produced an executable file for administration. The test administrator read the executable test file and administered the test. As much of the time-consuming computation as possible was done during test compilation so that it would execute faster when the test was administered. As an example of this type of computation, items for a maximum information testing strategy were sorted into a table organized by ability level during the compilation. When the test was executed, the time-consuming search was reduced to a fast table-lookup procedure.

The TCL system had some shortcomings in its design, however. Foremost among these was the somewhat clumsy nature of the author language. Although the language provided a means of test specification much simpler than FORTRAN programming, the test specifications were rather difficult to read. Furthermore, the implementation of the language allowed a reference to a specific item to appear only once in the test specification. This made some strategies (e.g., Robbins-Monro) impossible to specify without creating several copies of an item with different reference numbers in the item bank. Finally, the TCL language blurred the distinctions among item reference numbers, variables, functions (i.e., scoring procedures), and statement labels. Although this allowed the skilled user to develop "clever" specifications, it invited trouble when used by a more typical test developer.

Since so few testing systems existed, computerized instructional systems (CAI) were also considered. The most useful design idea came from the Control Data PLATO system. This idea was two-level authoring. For the skilled lesson author, an author language was provided. This language allowed great flexibility in lesson design. For the subject-matter expert who was not greatly skilled in programming, a menu system was available to allow lesson material to be inserted into an existing lesson format. This approach allowed a division of labor between the lesson writers and the programmers.

Design of a Complete Self-Contained System

The most complete testing to date, the TCL system, had three components: item banking, test specification, and test administration. A complete system needs at least one and probably two more components. Since much of adaptive testing is built around the item characteristics, a complete system must include a test analysis system so that, minimally, the important characteristics of the items can be estimated. Additionally, such a system should also include a method for reporting the results to the examinees. The software system designed in this project had all five components.

The complete design is detailed in the report by Vale et al. (1982). From the test developer's point of view, the most interesting part of the design is the test authoring system. Similar to PLATO, the test authoring system was a two-level system. The most flexible and comprehensive level is the author-language level. All tests developed in this system must at some point be specified in the author language. Everything that the system is capable of doing, in terms of test administration, can be controlled through the author language. Because the author language is comprehensive, it can be difficult for an inexperienced test developer to use, especially if she/he has not programmed before. To assist test authors who are not proficient programmers, a menu system forms the second level in the system. This level does not allow much flexibility in test design but it does allow a test developer to construct tests without having to program, even in an author language.

The author language is moderately structured and consists of eight types of statements. A test specification in the language is composed of test modules.

Subtests can be nested within tests (the system makes no formal distinction between a test and a subtest). To allow tests and subtests to be distinguished, two module delimiter statements are provided. The TEST statement denotes the beginning of a test or a subtest module and assigns it a name. The ENDTEST statement denotes the end of a test or a subtest module.

Variables are allowed in the language and are used to hold constants, scores, or other numerical values. In the language, as designed, variables are implemented as words of up to 10 characters. Local variables are defined only within a test module; local variables with the same name in different test modules will be different variables. Global variables are defined throughout all test modules; global variables with the same name in different test modules will be the same variable, and values will thus transfer from one module to another. Global variables are distinguished from local variables by beginning with an "$" sign. The second type of statement, the assignment statement, allows the test developer to assign the results of expressions to variables. The assignment statement in the author language is called SET.

The third type comprises the item statement. This statement, consisting minimally of a "#" sign followed by an item reference number, causes an item to be administered or included in an adaptive structure. It can control, to some extent, how the item is administered and can control what happens after the item is administered.

The declarative statements, SETSCORE and TERMINATE, determine what scores are calculated, what variables are assigned to them, and under what conditions the testing is to terminate. These statements are called declarative because they do not cause anything to happen; they just set up what will happen when the test is administered.

Each administration of a test will create a data file for the examinee. The output control statements, KEEP and AUTOKEEP, determine what is written to this file. KEEP writes specified identifying information and the values of specified variables to the file each time it is executed. AUTOKEEP is a declarative statement that works like KEEP but is set once at the beginning of the test module and thereafter executes automatically each time an item is administered.

The conditional statements, IF, ELSEIF, and ENDIF, allow logical expressions to determine whether a section of a module will be executed. The IF statement begins a logical clause. The ELSEIF statement continues the logical clause until a logical expression is satisfied. The ENDIF statement denotes the end of the logical clause.

The adaptive statements SEARCH, ENDSEARCH, SEQUENCE, and ENDSEQUENCE allow model-based and reentrant adaptive structures to be built. SEARCH executes a maximum information item search based on a specified variable and includes in the search all items listed between the SEARCH and ENDSEARCH statements. The SEQUENCE statement is used for reentrant intersubtest branching strategies. A SEQUENCE statement, similar to a SEARCH statement, uses the items listed between the SEQUENCE and the ENDSEQUENCE statements. The SEQUENCE statement administers one item, starting at the top of the list, every time it is executed. It keeps a pointer so that it does not administer any item more than once. Strata for the stradaptive testing strategy may be set up as sequences.

Finally, the menu statements allow a test template with blanks in it to be written. Menu statements are not really elements of the author language but rather are statements that control a template preprocessor. The INSTRUCT statement causes the preprocessor to print a line of instructions at the test developer's terminal when the preprocessor is run. Blanks (actually underline characters) signify that user input should be read by the preprocessor and inserted into the author language. Qualified blanks limit the kind of information that will be accepted (e.g., item reference numbers) or the number of items that will be accepted. A template containing these statements can be processed by the preprocessor to collect the information needed to complete the test specification from the test developer.

An Example of Test Specification

All test specification is done in the author language. Figure 1 presents an author language specification for a Bayesian adaptive test. This specification consists of a single test module, a test named BAYES. The specification begins by setting up the scores to be kept and the conditions for termination. This test will compute a Bayesian score and will keep the posterior mean and variance in the variables MEAN and VARIANCE. Termination will occur when the posterior variance drops below .01 (or when all of the items have been administered). The prior mean is initialized at 0.0 and the prior variance at 1.0.

Figure 1
Author Language Program for Bayesian Test

```
TEST BAYES ! BAYESIAN ALGEBRA TEST
!
   SETSCORE BAYESIAN (MEAN, VARIANCE) ! SET THINGS UP
   TERMINATE (VARIANCE < 0.01)
   SET MEAN = 0.0
   SET VARIANCE = 1.0
!
   SEARCH MEAN ! FIND ITEM MOST INFORMATIVE AT MEAN
      #MATH001    ! ASSOCIATIVE RULE
      #MATH013    ! COMMUTATIVE RULE
      #MATH144    ! SOLVE FOR X
      *MAPOOL     ! OLD ALGEBRA POOL
      *HARDPL     ! NEW DIFFICULT ALGEBRA POOL
      #MATH552    ! ROOT ITEM
      #MATH603    ! SET ITEM
   ENDSEARCH
!
   KEEP "ALGEBRA ", "MEAN AND", "VARIANCE", MEAN, VARIANCE
!
ENDTEST
```

The items listed between SEARCH AND ENDSEARCH will be considered, as well as the items included in the pools MAPOOL and HARDPL. Each item will be selected to maximize the information for an examinee with an ability level of MEAN, the current ability estimate. When the test ends, the two scores will be written to the data file, along with the label "ALGEBRA MEAN AND VARIANCE."

Although the author language specification for this test is not particularly complex, specification of the same test can be simplified considerably in the menu mode. Figure 2 shows how a test developer would develop the same test using the template preprocessor. The test developer has only to answer the questions as they are asked. The preprocessor produces an author language specification equivalent to the one shown in Figure 1.

Figure 2
Menu Type Test Specification

```
TEST DEVELOPMENT SYSTEM -- TEST COMPLETER PROGRAM      Ver. 1.0

What is the name of the template you would like to use?  BAYES

This template creates an adaptive test based on Owen's Bayesian
algorithm.  It assumes a N(0,1) prior and keeps the final posterior
mean and variance as scores.

Enter a title for the test :  ALGEBRA 101 MIDQUARTER -- 4/82

Now enter the items or pools you would like to include in
the test, one on each line.  Leave a blank line when you are
done.

? #MATH001
? #MATH013
? #MATH144
? *MAPOOL
? *HARDPL
? #MATH552
? #MATH603
? 

Construction of the test is complete.

Under what name would you like to save the new test?  ALG482

The test has been saved as ALG482.  Would you like to create
another?  NO

Test development is then over.  Remember to say BYE to the computer.
```

Figure 3 shows the template that the preprocessor requires to produce the author language equivalent to that in Figure 1 from the interactive session shown in Figure 2. It differs in two respects from the specification in Figure

1. First, it has blanks where item numbers and other constants will go. Second, it has instructions for the test developer that tell what is to be entered when the preprocessor prompts the user for a response. When this template is preprocessed, the blanks are replaced with information supplied by the test developer, and the menu statements are removed.

Figure 3
Template for Bayesian Test

```
! BAYESIAN TEST TEMPLATE
!
INSTRUCT This template creates an adaptive test based on Owen's Bayesian
INSTRUCT algorithm.  It assumes a N(0,1) prior and keeps the final posterior
INSTRUCT mean and variance as scores.
INSTRUCT
INSTRUCT Enter a title for the test :
!
TEST BAYES !  __
!
    SETSCORE BAYESIAN (MEAN, VARIANCE) ! SET THINGS UP
    TERMINATE (VARIANCE < 0.01)
    SET MEAN = 0.0
    SET VARIANCE = 1.0
!
INSTRUCT
INSTRUCT Now enter the items or pools you would like to include in
INSTRUCT the test, one on each line.  Leave a blank line when you are
INSTRUCT done.
INSTRUCT
!
    SEARCH MEAN ! FIND ITEM MOST INFORMATIVE AT MEAN
    #__#
    ENDSEARCH
!
    KEEP "ALGEBRA ", "MEAN AND", "VARIANCE", MEAN, VARIANCE
!
ENDTEST
!
INSTRUCT
INSTRUCT Construction of the test is complete.
INSTRUCT
```

Current Status of System Development

The design of the CAT system described in this paper is complete. It appears to be a useful design for an operational CAT development and administration. Work on development of the system is currently proceeding in two directions. First, a system very similar in design to the one discussed here is under development for implementation on a Digital Equipment Corporation PDP-11 computer system. This system will be a multi-user testing system incorporating most of the features of the microcomputer design as well as several additional features that were feasible on the larger PDP-11 system. This system is being written in Pascal and will be complete by the end of 1982.

Development of the microcomputer-based system described here has not yet begun. This development is scheduled to begin in mid-1983 as a second phase for the project in which the design was developed. The microcomputer-based system will be developed on a state-of-the-art microcomputer. Prototype systems will be given to test developers to try out, and modifications will be made to the system in order to enhance its utility. A CAT system should thus be available that will make the implementation of an adaptive test as simple as the implementation of a conventional test.

REFERENCES

Angoff, W. H., & Huddleston, E. M. _The multi-level experiment: A study of a two-level test system for the College Board Scholastic Aptitude Test_ (Statistical Report 58-21). Princeton NJ: Educational Testing Service, 1958.

Cory, C. Relative utility of computerized versus paper-and-pencil tests for predicting job performance. _Applied Psychological Measurement_, 1977, _1_, 551-564.

Cory, C. Interactive testing using novel item formats. In D. J. Weiss (Ed.), _Proceedings of the 1977 Computerized Adaptive Testing Conference_. Minneapolis: University of Minnesota, Department of Psychology, Psychometric Methods Program, 1978.

de Finetti, B. Methods for discriminating level of partial knowledge concerning a test item. _The British Journal of Mathematical and Statistical Psychology_, 1965, _18_, 87-123.

Hanna, G. S. Incremental reliability and validity of multiple-choice tests with an answer-until-correct procedure. _Journal of Educational Measurement_, 1975, _12_, 175-178.

Hunter, D. R. Research on computer-based perceptual testing. In D. J. Weiss (Ed.), _Proceedings of the 1977 Computerized Adaptive Testing Conference_. Minneapolis: University of Minnesota, Department of Psychology, Psychometric Methods Program, 1978.

Lord, F. M. The self-scoring flexilevel test. _Journal of Educational Measurement_, 1971, _8_, 147-151.

Owen, R. J. A Bayesian seuential procedure for quantal response in the context of adaptive mental testing. _Journal of the American Statistical Association_, 1975, _70_, 351-356.

Prestwood, J. S. The development of a computerized, domain-referenced assessment system for family medicine. In R. L. Holloway (Ed.), _Some issues surrounding the implementation of objectives-based education in family medi-_

cine. Minneapolis: University of Minnesota, Department of Family Practice and Community Health, 1980.

Shuford, E. H., Albert, A., & Massengill, H. E. Admissible probability measurement procedures. Psychometrika, 1966, 31, 125-145.

Vale, C. D. Computerized administration of free-response items. In D. J. Weiss (Ed.), Proceedings of the 1977 Computerized Adaptive Testing Conference. Minneapolis: University of Minnesota, Department of Psychology, Psychometric Methods Program, 1978.

Vale, C. D. Design and implementation of a microcomputer-based adaptive testing system. Behavior Research Methods and Instrumentation, 1981, 13, 399-406.

Vale, C. D., Albing, C., Foote-Lennox, L., & Foote-Lennox, T. Development of a microcomputer-based adaptive testing system. St. Paul MN: Assessment Systems Corporation, June 1982.

Vale, C. D., & Weiss, D. J. The stratified adaptive ability test as a tool for personnel selection and placement. TIMS Studies in the Management Sciences, 1978, 8, 135-151.

Weiss, D. J. Strategies of adaptive ability measurement (Research Report 74-5). Minneapolis: University of Minnesota, Department of Psychology, Psychometric Methods Program, November 1974.

ACKNOWLEDGMENT

This research was sponsored by the Office of Naval Research under Contract N00014-82-C-0132 through the Defense Small Business Advanced Technology Program.