Behavior Research Methods & Instrumentation
1981, Vol. 13 (4), 399-406

Design and implementation of a microcomputer-based adaptive testing system

C. DAVID VALE
Assessment Systems Corporation, St. Paul, Minnesota 55114, and
University of Minnesota, Minneapolis, Minnesota 55455

Adaptive testing is a relatively new form of test administration in which a test is tailored to the individual taking it by choosing items most informative about that person. Methods for determining which items are most appropriate take on a variety of forms, some requiring extensive computation, and almost all requiring administration by a computer. The increasing availability of inexpensive microcomputer systems has made adaptive testing possible when access to larger computer systems is impractical. To make implementation of a variety of adaptive testing methods feasible on a microcomputer, a system efficient from both the examinee's and the test constructor's perspectives is necessary. This paper begins by briefly outlining the strategies of adaptive testing developed to date and showing how, structurally, they can be grouped into three general categories. Considerations in design of a test-specification subsystem are then discussed as they relate to this categorization. Finally, a specific implementation of a subsystem for use under the CP/M microcomputer operating system is described. Techniques used to make the extensive computations required by adaptive testing feasible on a microcomputer are presented.

Adaptive, or tailored, testing is a relatively new form of psychological testing in which a test is tailored to an individual during the testing process, such that those items that are most appropriate to the individual and most informative about the characteristic being measured are administered. The tailoring is accomplished during, rather than prior to testing because the appropriateness of the items is dependent on the individual's status on the characteristic being assessed. In the area of ability measurement, on which most adaptive testing research has focused, this amounts to administering easy items to low-ability individuals and difficult items to high-ability individuals.

Assessment of ability using a test (although adaptive testing technology is general across many psychological characteristics, for convenience, discussion here will be limited to ability) requires that items be administered, and the choice of the item requires that the ability level of the examinee be known. Simultaneous selection of items and estimation of ability does not have a single optimal solution, but a variety of testing strategies to approximate the optimal solution have been developed. There has not yet been firm agreement among researchers regarding which strategy is best. There has, however, been almost consensual agreement that all of the strategies are sufficiently complex to require that they be administered by a computer.

The microcomputer-based testing system described in this paper was developed at the University of Minnesota under Contract N00123-79-C-1273 from the Navy Personnel Research and Development Center, San Diego, California. The author wishes to thank David J. Weiss for his critical review of an early draft of this manuscript.

In computerized administration of a test, the examinee sits in front of an interactive computer terminal on which questions are presented in textual or graphic form. The examinee responds to an item by pressing one or more keys on a response panel that may vary in complexity from a simple two-button panel (for yes-no, true-false, etc.) to a panel closely resembling a typewriter keyboard. Following an examinee's response to an item, the computer evaluates the response, selects the next item, and presents it. At the end of the test, the computer scores the test. Feedback may or may not be provided to the examinee during or at the end of the test.

Development of a computer system to administer adaptive tests requires several hardware and software system components. Considerations in selection of hardware have been discussed by Underwood (Note 1). General considerations of software have been discussed by Dewitt and Weiss (1976), who described software systems for implementing adaptive testing on a general-purpose timesharing computer and a real-time minicomputer. This paper, in part, describes a software system developed to implement adaptive testing on a microcomputer. More generally, it classifies adaptive testing procedures into three categories and describes a testing system that, without modification, can be used to specify and administer adaptive tests under a wide variety of specific strategies. The paper is divided into three sections. The first section describes the major adaptive testing strategies developed to date and integrates them into the classification scheme. The second section describes a control language, derived from the classification, that is useful for specifying adaptive

0005-7878/81/040399-08$01.05/0

testing strategies. Finally, the third section describes a software system in which the control language is incorporated into a self-contained microcomputer-based adaptive testing system.

A CLASSIFICATION OF ADAPTIVE TESTING STRATEGIES

An adaptive testing strategy consists of a method of selecting items to administer and a method of converting the responses to those items into a test score. Item selection algorithms reduce to one of three types of branching: from item to item in a predetermined structure, from subtest to subtest, or as a function of a complex rule specified by a mathematical testing model. Scoring algorithms reduce to a few of general utility. In the subsections that follow, major testing strategies are used to illustrate the basic branching and scoring types.

Interitem Branching

The interitem branching strategies are conceptually the simplest of the item selection algorithms. In ability testing, these strategies are implemented by structuring an item pool such that each item leads to one of two other items, depending on its item score (i.e., correct or incorrect). If the response to an item is correct, testing continues with a specific prespecified item of, typically, greater difficulty. If the response is incorrect, testing continues with a prespecified item of, typically, lesser difficulty. The defining characteristic of this class of strategies is that each individual item leads to an invariant set of items dependent solely upon the response to that item.

Strategies in this category often appear to have triangular or pyramidal structures when drawn graphically. Figures 1 and 2 show two such strategies, a simple pyramidal strategy and a Robbins-Monro strategy (Weiss, Note 2). In all figures representing test strategies presented in this paper, the horizontal dimension represents difficulty levels ranging from easy on the left to difficult on the right. Each node in the diagrams represents an item. Plus stands for a correct response and minus for an incorrect response. The Robbins-Monro process of Figure 2 is actually capable of representing

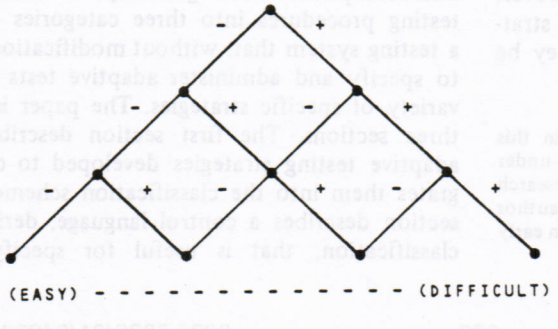


Figure 1. A pyramidal testing strategy.

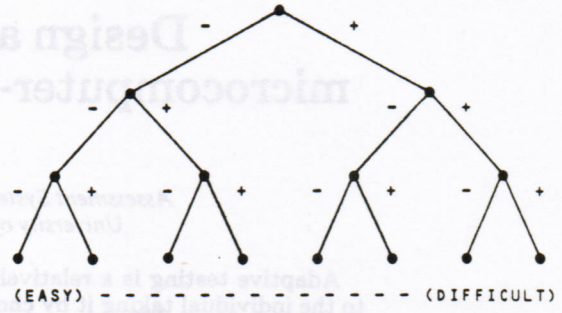


Figure 2. A Robbins-Monro process.

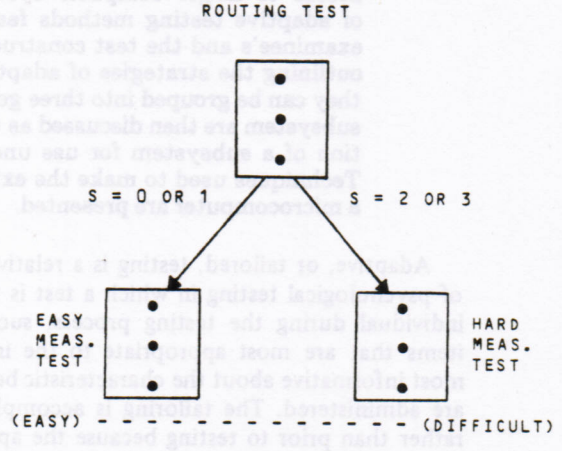


Figure 3. A two-stage testing strategy.

any adaptive testing strategy based on dichotomously scored items, although that is not its intent as presented here.

Intersubtest Branching

Intersubtest branching strategies are similar in concept to the interitem strategies. The conceptual differences stem from the fact that each node in the diagrams now consists of several items rather than one. This typically leads to fewer nodes and allows for the possibility of reentrant nodes in which only a portion of the items in a subtest are administered before branching to another. The remaining items in the subtest may be administered at a later time.

The nonreentrant form of this class of item selection algorithms is exemplified by the multistage strategy or its simplest case, shown in Figure 3, the two-stage strategy. In the two-stage strategy, a single routing test is administered to all examinees. Responses to these items are scored, and on the basis of the score, one of several second-stage tests containing items of more appropriate difficulty is administered. Items in each of the subtests are typically administered in a linear fashion, simply starting at the first item and proceeding, without reconsideration, to the end of the subtest.

In the reentrant form, appropriateness of the sub-

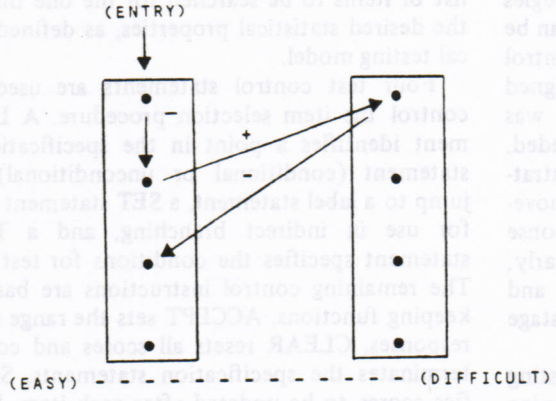


Figure 4. A flexilevel testing strategy.

tests is reevaluated periodically, and if the subtest appears inappropriate, administration transfers to another, keeping a pointer for later return to the subtest. A simple example of a test of this type is the stradaptive (stratified adaptive) strategy, in which appropriateness is evaluated after each item (Vale & Weiss, 1978). If the response is correct and a more difficult subtest is available, administration transfers to that subtest. Similarly, if the response is incorrect, administration transfers to an easier subtest. The flexilevel test (Lord, 1971), an elementary form of the stradaptive strategy having only two subtests, is shown in Figure 4. In the flexilevel test, items are grouped into two subtests, one containing easy items and the other containing difficult items. In the easy subtest, items are arranged in descending order with respect to difficulty. In the difficult subtest, items are arranged in ascending order of difficulty. Testing begins with the first item in one of the subtests and transfers to the other if indicated by the item response (e.g., to the difficult subtest after a correct response in the easy subtest). This continues until a predetermined number of items has been administered.

Model-Based Branching

Model-based branching strategies are most often based on item response, or latent trait, theory (Birnbaum, 1968; Lord & Novick, 1968), which assumes that item responses are probabilistically related by a specified function to a continuous underlying trait or ability. In these strategies, the score consists of an estimate of the level of ability on the underlying trait that characterizes the individual. Testing begins with administration of a generally appropriate item. The trait estimate is updated after each item is administered, and the next item, at each stage, is determined by deciding which item, of those available, will best improve the trait estimate. The test structure resulting from these strategies is relatively amorphous, so no attempt to diagram such a structure is provided here.

Two general strategies belonging to this class have been proposed to date. One is Bayesian in form and begins testing by assuming a prior distribution of ability (Owen, 1975). After each item is administered, a Bayesian posterior ability distribution is formed from the prior and the likelihood function characterizing the item relative to the underlying ability. This posterior distribution is then used as the prior distribution for the next stage, and an item is selected to best minimize the variance of the next expected posterior distribution.

The other strategy is similar, except that it selects items providing the most statistical information (Birnbaum, 1968; Lord & Novick, 1968) at the current estimate of the trait. Assumption of the Bayesian model is not required but may be used if a Bayesian estimate of the trait is desired. Computationally, this strategy is somewhat less taxing than the Bayesian strategy.

Scoring

Scoring methods used for adaptive testing can be grouped into three general categories: simple, model based, and custom. Simple methods are exemplified by the proportion correct, the average difficulty of all items administered, and the difficulty of the last item administered. These are all relatively easy to compute but, unfortunately, have limited interpretability across the various item selection methods. They are truly appropriate only for some of the interitem branching methods. The model-based scoring methods consist of least squares Bayesian (Owen, 1975), modal Bayesian (Samejima, 1969), and maximum likelihood (Birnbaum, 1968) scoring. Custom scoring methods are represented by procedures developed for a single testing strategy, such as the all-item score for the pyramidal strategy (Weiss, Note 2) and the interpolated stratum-difficulty score for the stradaptive strategy (Vale & Weiss, Note 3).

With the exception of the custom scoring methods, any scoring method can, computationally, be coupled with any item selection strategy. This may require imposition of a mathematical model on an item selection strategy that would not otherwise require it (e.g., to use Bayesian scoring with a pyramidal item selection strategy), or it may result in a score generally unrelated to what the items are assessing (e.g., when the proportion-correct score is used with Bayesian item selection). The scoring procedures can still reasonably be computed. There may, however, be some difficulty interpreting them.

A TEST-SPECIFICATION CONTROL LANGUAGE

In the design of any computerized testing system, one early concern is how flexible the system will be in its ability to specify subtly different test forms. The four strategies presented in the previous section were only examples and give little indication that a much wider variety of strategies is possible. In designing a testing system for research use, a good deal of flexibility is

needed to allow for those variations. If, on the other hand, the system is for operational use and the strategies are all prespecified and few in number, flexibility can be sacrificed for operational simplicity. The test control language to be discussed in this section was designed primarily for use in a research environment. It was designed to allow the flexibility that would be needed, for example, to change the stradaptive branching strategy from a simple up-one/down-one stratum movement to a movement of one up after a correct response and two down after an incorrect response. Similarly, it was designed such that the length, number, and internal branching characteristics of the multistage modules could be varied.

In development of the microcomputer-based testing system, two primary concerns guided the initial planning of the system: (1) It had to provide the individuals developing tests an efficient and general means of specifying test structures, and (2) it had to provide test administration with minimal delay between a response and the presentation of the next item. To make administration delays minimal, as much of the computation as possible had to be accomplished before testing began. To keep test-development time to a minimum, these computations had to be done after the test developer had completed specification of the test. To accommodate these speed requirements at both ends of the process, a batch-type test-specification system was designed that allowed the test constructor to prepare a set of specifications and then submit the specifications for batch processing. During the batch processing, which required no intervention from the test developer, the heavy computations were done and a compact testing module for use during the administration phase was produced.

The test specifications were designed to be similar to the control language used by SPSS (Nie, Hull, Jenkins, Steinbrenner, & Bent, 1975). That is, the general structure and readability of SPSS was retained. The control statements were entirely different, however, and a free-format input and FORTRAN-like jump statements were added.

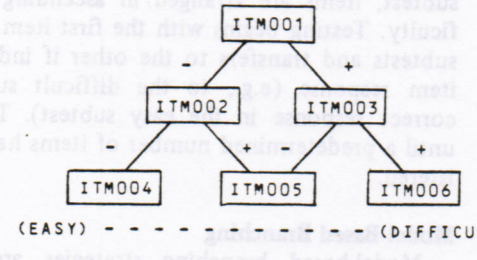
Control Statements

The control statements are divided into two classes: test control statements and module-specification statements. The module-specification statements set up modules of items required by the test structures discussed in the previous section. Three module types (not directly paralleling the three test structures) were created. A LINEAR structure includes a list of test items to be administered from beginning to end, unless terminated, and is not reentrant. A BRANCH module specifies a set of items interrelated by branch directions. Branching can be either item to item or item to subtest, and this specification can be used for both the interitem branching strategies and some of the intersubtest branching strategies. Finally, a SEARCH structure specifies a list of items to be searched for the one that best meets the desired statistical properties, as defined by a statistical testing model.

Four test control statements are used to actually control the item selection procedure. A LABEL statement identifies a point in the specifications, a JUMP statement (conditional or unconditional) executes a jump to a label statement, a SET statement sets a pointer for use in indirect branching, and a TERMINATE statement specifies the conditions for test termination. The remaining control instructions are basically housekeeping functions. ACCEPT sets the range of acceptable responses, CLEAR resets all scores and counters, END terminates the specification statements, SCORE specifies scores to be updated after each item, STATISTICS specifies scores to be written to a file, WRITE specifies the amount of information to be written, + denotes a continuation, and $ denotes a comment.

Specification of Interitem Branching

Test structures defined by interitem branching are specified using the SET and BRANCH instructions. A variable, ITEMP, is set to the initial item number (e.g., the top item in the pyramid), and each item administered specifies branching actions for each possible response. Symbolically, the specification is similar to that shown in Figure 5. In this example, three items are to be administered. Everyone is to take ITM001. Those answering incorrectly proceed to ITM002. Those answering incorrectly proceed to ITM003. The procedure will terminate after administering the third item (i.e., ITM004, ITM005, or ITM006). Specification of the structure begins by specification of the initial item. Branching




```
$
$ SPECIFICATION OF A PYRAMIDAL TEST
$
SET ITEMP = #ITM001
BRANCH
#ITM001 INCORRECT:#ITM002  CORRECT:#ITM003
#ITM002 INCORRECT:#ITM004  CORRECT:#ITM005
#ITM003 INCORRECT:#ITM005  CORRECT:#ITM006
#ITM004
#ITM005
#ITM006
$
$ END OF SPECIFICATION
$
END
```

Figure 5. Specification of a pyramidal test.

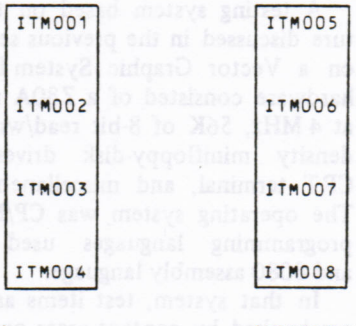
is specified for the first two stages. Lack of branch specifications in the last stage informs the system, by default, that the module is complete.

Specification of a Robbins-Monro process would be similar in form. The difference would occur in that each item would branch to two unique items, with no folding back as was observed in the pyramidal strategy, in which ITM002 and ITM003 led to ITM005.

Specification of Reentrant Intersubtest Branching

Reentrant intersubtest branching is also implemented using the BRANCH instruction. This is illustrated using a flexilevel test diagramed in Figure 6. Eight items are grouped into two subtests, an easy one consisting of Items 1-4 and a difficult one consisting of Items 5-8. Specification begins by setting two subtest pointers, P1 and P2, to the beginnings of the two subtests and by setting the initial pointer to Item 1. The test illustrated is to be four items long, so the procedure is set to terminate after administration of four items. The branch specifications are similar to those used with the pyramidal strategy. The important difference is that when a subtest is exited, the branch is to the pointer indicating the reentry point in the other subtest and, at the same time, a return pointer is set indicating the return point in the subtest exited. Branching within a subtest is done linearly, using the standard interitem branching method. Branching was not specified for Items 4 and 8, as it was unnecessary and there was no appropriate item to branch to. (Actually, Item 8 will never be reached under the termination rule used and was included only for symmetry, so that testing could be initialized with Item 5 as well as with Item 1.)

The stradaptive strategy would be implemented in a similar fashion. One pointer would be set for each stratum (i.e., subtest), and branching via pointers would occur in both directions from the inner strata.




```
$
$ SPECIFICATION OF A FLEXILEVEL TEST
$
SET  P1=#ITM001, P2=#ITM005, ITEMP=#ITM001
TERMINATE  NADMIN=4
BRANCH
$
$ EASY STRATUM (LEFT)
$
#ITM001 INCORRECT:#ITM002  CORRECT:P2, P1=#ITM002
#ITM002 INCORRECT:#ITM003  CORRECT:P2, P1=#ITM003
#ITM003 INCORRECT:#ITM004  CORRECT:P2, P1=#ITM004
#ITM004
$
$ DIFFICULT STRATUM (RIGHT)
$
#ITM005 INCORRECT:P1, P2=#ITM006  CORRECT:#ITM006
#ITM006 INCORRECT:P1, P2=#ITM007  CORRECT:#ITM007
#ITM007 INCORRECT:P1, P2=#ITM008  CORRECT:#ITM008
#ITM008
$
$ END OF TEST SPECIFICATION
$
END
```

Figure 6. Specification of a flexilevel test.

Specification of Nonreentrant Intersubtest Branching

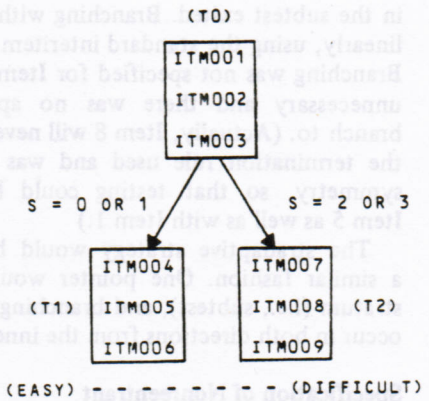
Conceptually, nonreentrant branching could have been implemented using the reentrant branching capabilities. Practically, however, operational reentrant strategies in general branch on the basis of a single-item response, whereas the nonreentrant strategies typically branch on a more complex rule. As the control language is defined, the branch statement is used only for single-response branching. More complex rules are implemented using the JUMP statement.

Figure 7 diagrams the specification of a two-stage testing strategy. The test consists of a routing test (T0) and two measurement tests (T1 and T2), one of which will be administered. The choice of measurement test depends on whether two or more items are answered correctly in the routing test. If so, Measurement Test T2 will be administered. Otherwise, Measurement Test T1 will be administered. Testing will terminate with the end of the test specification.

Although linear subtests were used as examples here, subtests in such a multistage strategy do not have to be linear. They could as well be pyramidal, Robbins-Monro, or one of the model-based procedures to be discussed below.

Specification of Model-Based Branching

Model-based branching strategies typically incorporate such complex mathematical operations that a language to explicitly specify the operations would have to be as basic as FORTRAN. The intent of this specification language was to create something a good deal simpler than FORTRAN, so the model-based search procedures were limited to single commands. Because of the extensive computations involved in these procedures, only one was deemed amenable to a microcomputer-based system. This was the maximum-information item search procedure. In this procedure, an ability estimate is obtained after each item is administered. Items remaining unadministered at that point are then evaluated on the basis of the statistical information they provide about the underlying ability level. (Information, as it is used here, is inversely related to the standard error of measurement for the individual being tested.) The item expected to provide the most information is selected for administration.




```
$
$ SPECIFICATION OF A TWO-STAGE TEST
$
$ ROUTING TEST (T0) :
$
LABEL T0
LINEAR
#ITM001
#ITM002
#ITM003
JUMP T1 (NCORR LT 2)
JUMP T2 (NCORR GE 2)
$
$ FIRST MEASUREMENT TEST (T1) :
$
LABEL T1
LINEAR
#ITM004
#ITM005
#ITM006
JUMP E1
$
$ SECOND MEASUREMENT TEST (T2) :
$
LABEL T2
LINEAR
#ITM007
#ITM008
#ITM009
JUMP E1
$
$ END OF TEST SPECIFICATION
$
LABEL E1
END
```

Figure 7. Specification of a two-stage test.

A single statement, SEARCH, is used to implement this item selection procedure. In the search statement, the ability estimate to be used in evaluating the information is specified and the items to be included are listed. Figure 8 illustrates a Bayesian testing strategy (Owen, 1975) in which items are selected on the basis of expected information. After specifying a termination rule, the search procedure is invoked, specifying that the Bayesian score should be used in evaluating information. The items to be included in the search are simply listed following the search statement.

The model-based search is the least flexible of all the branching strategies. Variations in procedure can still be introduced, however, in the score on which the search is based and in the criterion used to terminate the test.

Specification of Scoring Algorithms

Scoring can be done at two points in the testing process. It can be done after each item, or it can be done after each test module (e.g., specification statement). Two separate statements are used to specify these two options. The SCORE statement lists the scores that are to be calculated after each item is administered. The STATISTICS statement specifies the scores that are to be written to a file and, by implication, specifies the scores that must be calculated before the statistics are written. It is important that the distinction be made, because scores forming the basis of a search must be reevaluated after each item. Some scores require so much computation, however, that it is undesirable to calculate them after every item when they are not needed for the search.

A total of eight scores were included in implementation of the system. These were the testing time required, the proportion of correct responses, and three latent trait scores (a least squares Bayesian estimate, a modal Bayesian estimate, and a maximum-likelihood estimate) and their variances.

DEVELOPMENT OF A MICROCOMPUTER-BASED TESTING SYSTEM

Implementation

A testing system based on the language and structure discussed in the previous section was implemented on a Vector Graphic System B microcomputer. The hardware consisted of a Z80A microprocessor running at 4 MHz, 56K of 8-bit read/write memory, two quad-density minifloppy-disk drives, a memory-mapped CRT terminal, and miscellaneous interface hardware. The operating system was CP/M Version 2.1, and the programming languages used were FORTRAN IV and 8080 assembly language.

In that system, test items and instructional screens are banked by content areas on several CP/M files, one file per content area. Instructions, for example, may be banked on a single file and identified as INS001 to INS999. These files are developed and maintained as ASCII source-code files. All entry and editing is done using the system editor.

The first program in the system, CONVRT, converts the serial access source files to random-access binary files. Two versions of each item file are kept. Whenever modifications are made to a source file, the file is completely reconverted to a binary file; no individual modifications are made to the binary file. The binary file is then used for all further processing within the system. The decision to keep a separate source file for each item file was made primarily for programming convenience. As implemented, a separate editor did not have to be written and test developers do not have to learn to use a second editor.

Test-specification statements are entered onto another source file, using the system editor. The test constructor enters the statements in a manner similar to any other

```
$
$ SPECIFICATION OF A BAYESIAN TEST
$
TERMINATE  (BVAR LT 0.1 OR NADMIN EQ 5)
SCORE  BSCORE, BVAR
SEARCH  BSCORE
#ITM001
#ITM002
#ITM003
#ITM004
#ITM005
$
$ END OF TEST SPECIFICATION
$
END
```

Figure 8. Specification of a Bayesian test.

program. This file contains the complete specification of the test to be administered. Modifications in the testing algorithm can thus be implemented by making changes in this source file.

The second program in the system, TCLCOM, reads the instructions from the test-specification file and constructs the test. This requires translating the language instructions into an object code more easily processed by a computer, collecting the specified items from various binary files (possibly residing on several different disks), and creating a single object file containing all of the items and test information necessary to administer the test. Branching information logically belongs with an item but is, for flexibility, contained in the test specifications. When the object file is created, this information is inserted in the appropriate item records. If the specification involves a search, information required for the search is set aside on another file for further processing.

A third program, conceptually part of the second, continues the processing if a search operation is specified. To speed the test administration process, the search computations are done by this program and the results are put into a large file-resident look-up table. In this table, items are rank ordered on their acceptability at each of 51 locations along the ability scale. During administration, item selection is simply a matter of reading the appropriate segment of the table and searching down it until an unadministered item is found.

The final program in the system, ADMIN, obtains its required information from the object file and the search-table file. It then simply interprets the instructions on the object file and administers the specified test. As designed, all of the administration system resides on one of the disks. This leaves the other disk free for examinee information. This design allows for the situation in which an examinee might be routed through an extensive evaluation process at several different stations. In this situation, the examinee could carry his/her evaluation data from one station to the next on a diskette and simply insert the diskette into the system at each station. The testing system, on an appropriate file, would write all of the specified score information onto this second diskette.

Evaluation

The system developed has thus far been used primarily as an adaptive testing demonstration system. Preliminary evaluation of the system suggests that a microcomputer of the size used here can reasonably be used to administer adaptive tests of most strategies of interest. All of the general strategies discussed in this paper have been implemented on the system with satisfactory results. The only area of difference expected between this system and a main-frame computer-based system was the time delay between items. This delay ranged from 1 to 3 sec on this system, depending on the strategies evaluated. The search strategy required the most time because a second disk access was required to search the table. This delay should not depend on the size of the test and thus should be no longer than 3 sec, regardless of the number of items searched. Some improvements suggested but not yet implemented promise to reduce this time to less than 2 sec. Replacement of the floppy disks with a hard disk would reduce all delays to well under 1 sec. It should be noted, however, that even a 3-sec delay is less than must typically be expected on a larger timeshared system.

As was expected, the amount of time required to convert source items and source specifications to the object file was large. A file of 100 items administered in branched mode required 5 min to convert the source items to binary items and nearly 8 min to translate the specifications to the object file. A typical adaptive test containing 300 items could be expected to take three times as long. A test requiring construction of a search table would take even longer. Fortunately, intervention by the test constructor is not required in this batch-oriented system. This time is thus inconsequential. If, on the other hand, a fully interactive test construction system had been developed, this time would be broken into short intervals between specification statements. The test constructor would thus be tied to the machine throughout the process. A batch-oriented process thus appears to be the appropriate mode for this environment.

In general, this system has demonstrated that a fully capable adaptive testing system can be implemented on a relatively small microcomputer without apparent degradation of performance. Cost effectiveness of such an implementation will depend on the environment in which it is used. It will also depend on the reliability of the equipment in such an environment, an aspect yet to be evaluated.

REFERENCE NOTES

1. Underwood, M. A. Computerized adaptive testing and personnel accessioning system design. In D. J. Weiss (Ed.), *Proceedings of the 1977 computerized adaptive testing conference.* Minneapolis: University of Minnesota, Department of Psychology, Psychometric Methods Program, July 1978.

2. Weiss, D. J. *Strategies of adaptive ability measurement* (Research Report 74-5). Minneapolis: University of Minnesota,

Department of Psychology, Psychometric Methods Program, December 1974.

3. Vale, C. D., & Weiss, D. J. *A simulation study of stradaptive ability testing* (Research Report 75-6). Minneapolis: University of Minnesota, Department of Psychology, Psychometric Methods Program, December 1975.

REFERENCES

Birnbaum, A. Some latent trait models and their use in inferring an examinee's ability. In F. M. Lord & M. R. Novick (Eds.), *Statistical theories of mental test scores*. Reading, Mass: Addison-Wesley, 1968.

Dewitt, L. J., & Weiss, D. J. Hardware and software evolution of an adaptive ability measurement system. *Behavior Research Methods & Instrumentation*, 1976, **8**, 104-107.

Lord, F. M. The self-scoring flexilevel test. *Journal of Educational Measurement*, 1971, **8**, 147-151.

Lord, F. M., & Novick, M. R. *Statistical theories of mental test scores*. Reading, Mass: Addison-Wesley, 1968.

Nie, N. H., Hull, C. H., Jenkins, J. G., Steinbrenner, K., & Bent, D. H. *Statistical package for the social sciences*. New York: McGraw-Hill, 1975.

Owen, R. J. A Bayesian sequential procedure for quantal response in the context of adaptive mental testing. *Journal of the American Statistical Association*, 1975, **70**, 351-356.

Samejima, F. Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph Supplement*, 1969, **17**, 1-100.

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.