7. Implementation Decisions in Designing Computer-Based Instructional Testing Programs

John V. Noonan
*Applied Learning International, Naperville, IL*

Paul D. Sarvela
*Department of Health Education, Southern Illinois University, Carbondale, IL*

Follow this and additional works at: [http://digitalcommons.unl.edu/buroscomputerdecision](http://digitalcommons.unl.edu/buroscomputerdecision)

Part of the [Educational Assessment, Evaluation, and Research Commons](https://digitalcommons.unl.edu/assessment_evaluation_and_research_commons) and the [Educational Methods Commons](https://digitalcommons.unl.edu/educational_methods_commons)


[http://digitalcommons.unl.edu/buroscomputerdecision/9](http://digitalcommons.unl.edu/buroscomputerdecision/9)

This Article is brought to you for free and open access by the Buros-Nebraska Series on Measurement and Testing at DigitalCommons@University of Nebraska - Lincoln. It has been accepted for inclusion in The Computer and the Decision-Making Process by an authorized administrator of DigitalCommons@University of Nebraska - Lincoln.
INTRODUCTION

From preschool to graduate school, computer-based instruction (CBI) has become an increasingly common event in today’s education and training community. The interactive characteristics of CBI and its ability to simulate advanced concepts and operations, such as patient management simulations for medical students (Whiteside & Whiteside, 1987/88) or the maneuvering of a jet airplane (Conkright, 1982), make CBI an attractive new instructional delivery system for educators working in many different fields.

Because of these qualities, the computer has tremendous potential in educational and psychological measurement. For example, Millman & Arter (1984) describe how the computer aids in maintaining test-item banks. Item forms can be used by test specialists to develop computer-generated items from a set of well-defined item characteristics (Hambleton, 1984), which saves valuable time in item construction. Millman and Outlaw (1978) suggest that an additional advantage of item forms is that more items can be produced than those stored on a computer. Computers can also be used to administer tests. The advantages of using computer-administered tests range from the ability to individualize testing to increasing the efficiency and economy of analyzing testing information (Ward, 1984). Finally, computers can be used to score tests, report results, and conduct statistical analyses on the scores (Noonan & Dugliss, 1985).

Although the computer has a wide variety of instructional applications, computer technology is not a panacea for solving all educational problems. For instance, although there are a number of ways in which the computer could possibly improve the quality of instruction in our schools, there is currently a
paucity of high-quality courseware available for educational purposes. Some educational software evaluation specialists suggest that up to 90% of the educational software available today is not worth purchasing (Olds, 1983). Measurement and evaluation specialists face similar problems. The costs associated with the design and development of good computer-based testing (CBT) programs are often prohibitively expensive. For this reason, when the computer is chosen as the testing delivery system, careful analysis of implementation questions and issues must take place.

The purpose of this chapter is to identify a number of practical implementation decisions that must be made when designing and developing criterion-referenced tests (CRTs) as a part of a larger system of computer-based instruction. Many of the concepts discussed generalize beyond large-scale courseware development efforts and apply to areas such as CBT in professional certification or licensing examinations, minimal competency testing at the local or state level, and norm-referenced testing. This chapter extends earlier guidelines that addressed microcomputer-based testing (Mizokawa & Hamlin, 1984) and computer use for various stages of the testing process (Noonan & Dugliss, 1985).

We have clustered CBT development decision areas into four categories: test construction, test security, item presentation, and response capturing and scoring. Many of the decisions are interrelated, since the actions resulting from one decision limit choices at another decision point (i.e., a decision to allow a student to preview items at the start of a test generally precludes the option of adaptive testing when deciding item sequencing, since item presentation strategies in adaptive testing are dependent on the student's history of responses to previous items). The chapter concludes by introducing a checklist (Appendix A) designed to aid courseware developers and measurement specialists in making appropriate CBT implementation decisions.

Test Construction

A number of issues must be considered when constructing tests to be used for computer-based testing and instruction systems. This section will discuss areas related to the following test construction decisions: the decision to use either diagnostic or mastery tests; routing; how and which objectives are to be tested; item type; the use of embedded or block tests; size of item pools; test-taking policy; and item tryout and analysis.

Diagnostic Versus Mastery Tests. The test designer must determine whether tests to be developed are to be used to diagnose areas of difficulty or simply provide more global measures of mastery. Because diagnostic and mastery tests are used for different purposes, the methods used to construct these types of tests are also different. For example, a diagnostic test (sometimes called or used as a placement test) implemented on a CBI system would usually use an elaborate set
of routing decisions, where the testing sequence is directly related to performance on earlier subsets of items. If incorrect answers are given, the student could be routed to a set of items structured to identify or classify the types of errors the student has made. The diagnostic information could then be used to tailor the CBI to the student’s needs. In a mastery test, the student might simply proceed through the test, and either pass or fail the examination; no branching decisions take place until the student completes the test.

In addition to these differences, discontinue criteria can also be applied differently for mastery and diagnostic tests. Discontinue criteria are those standards which determine when students leave the test; students may meet the discontinue criteria by either passing the test or receiving too many errors on the test. (Discontinue criteria will be discussed in detail in a later section.) In a mastery test, once the student passes the minimal number of items or objectives required to establish mastery, or once the student fails a certain amount of the material, the testing could be stopped and the student would be returned to instructional material. In diagnostic tests, failure at a certain test level might move the student to new and less difficult material. Given the elaborate possibilities for branching students based on their responses, the decision to use either mastery or diagnostic type tests is a major concern in test construction.

Other problems related to the differences between mastery and diagnostic tests are the ways in which test items and test objectives are matched. In a mastery test, subscoreing of objectives might not be needed; however, in a diagnostic testing scenario, test items and their associated objectives must be carefully matched so that decisions can be made concerning the branching of students to appropriate sections of the test. This impacts the complexity with which the tests are programmed.

Finally, the way in which response analysis is to be used must be considered. Sophisticated analyses of student errors, particularly when using diagnostic testing procedures, are indeed desirable. However, valuable computer-programming time is needed to produce the complicated scoring routines. Therefore, one must be certain that the benefits derived from an elaborate response analysis program outweigh the costs associated with constructing such a system.

Routing Decisions. The CBT test designer needs to consider routing (also known as branching) decisions that have to be made. The designer needs to determine if the student will be remediated when incorrect answers are given, as well as determine where remediation takes place. If poor performance is indicated, it should be decided if the student will be prevented from entering future lessons. Finally, one must determine if students who perform well on pretests (if there is a pretest) may bypass the lesson.

Objectives Tested. CBI programs are usually linked to well-defined instructional objectives, and it is the responsibility of the test designer to decide how
mastery of the objectives will be tested. One might simply conclude that each objective should be tested at the end of the unit or lesson in which the content is covered. However, there are situations in which this strategy is not advisable. Testing numbers of objectives can consume too much time, both for the student and the programmer. In many cases, the designer may want to replace some of the testing with lesson practice items that have some sort of mastery criteria. In addition, the designer should analyze the hierarchy of learning objectives to see if any of the objectives can be subsumed by testing higher-level objectives. In other settings, when critical or important information is to be learned, retesting two or three times is necessary to determine if mastery has been retained over the course of instruction.

The type of learning objective to be tested should also be considered, since traditional instructional theory (e.g., Gagné & Briggs, 1974) suggests that the learning objective determines, in part, the method of testing. For example, the Instructional Quality Inventory (IQI), an instructional systems quality assurance model currently used by the Department of Defense in the design and development of their training programs (Wulfeck, Ellis, Richards, Wood, & Merrill, 1978), carefully considers the learning objectives when designing and developing curriculum materials, instructional methods, and tests. Using the IQI system, one can classify learning objectives on the basis of the task to be performed and the type of information that must be learned. Any given objective can be classified as a fact, category, procedure, rule, or principle. An objective can further be classified as one which must be either recalled (from memory) or recognized, or, performed either with a job aid ("use-aided" IQI classification) or without a job aid ("use-unaided" IQI classification). If one uses IQI in the design and development of tests, recall-fact type of objectives would be tested in a manner quite different from recognize-fact type of objectives. For instance, if the objectives are recall-fact type of objectives, theoretically, only constructed-response items (short answer, essay, fill-ins) can be used. If the objectives are recognize-fact, selected-response items (such as multiple-choice, true-false, or matching) can be used. These issues not only have an impact on the method in which the test is programmed into the computer, but also affect the types and numbers of items which need to be constructed for each test.

**Item Type.** Most CBT software programs and authoring systems are well equipped to handle selected-response items. The programming for these item types is relatively easy, and the response analysis for correct and incorrect items is also fairly easy to construct and implement. On the other hand, constructed-response items are extremely difficult to design, put "on-line," and score on the computer. Since most CBT delivery systems do not have natural language processing (artificial intelligence), it becomes extremely difficult to specify and program all possible correct student-constructed answers. Therefore, the testing system is at risk of unfairly penalizing students who actually provide a correct
answer (false negative). At the same time, the system might mistakenly interpret an incorrect answer as a correct one, and unfairly give a student credit (false positive).

**Embedded Versus Block Tests.** It is sometimes desirable to test the student while he or she is working through the instruction, through a series of items which are administered throughout the lesson (embedded tests). Embedded testing might occur where there is a large amount of information which needs to be learned, or when formal postinstruction testing (block tests) is not feasible. It may also be useful in the beginning stages of learning, where frequent checks on student understanding of fundamental concepts is necessary.

If embedded tests are to be used, the test designer should determine if the students will be told that they are being tested. There are advantages and disadvantages of informing (or not informing) an individual that he or she is being tested. For example, if a student believes the embedded test is actually just a series of practice items, he or she might bypass them or answer them carelessly. Conversely, embedded tests can be used to reduce test anxiety. In this case it could be inappropriate to tell an individual that he or she is being tested. Also, one must consider the type of learning that is taking place. An objective that synthesizes prior objectives would be tested at the end of instruction. One would not use an embedded test strategy in this situation.

Finally, the decision to use embedded or block tests can be influenced by requirements for parallel or equivalent forms of tests. If strict psychometric specifications are put into place, it may be better to use block rather than embedded tests, because psychometric analyses of tests (e.g., reliability, discrimination, and difficulty) are based on assumptions related to tests that are delivered in “block” form. If tests are administered in an “embedded” manner, it may be difficult to compute parallelism between measures. (This problem is eliminated if item analysis and reliability assessment is conducted before the tests are incorporated into the courseware.)

**Item Pools.** Several factors influence the size of the item pools for computer-based tests. Requirements for parallel and equivalent forms of the test must be considered. If students who fail a test are to be retested, it may be appropriate to offer a second form of the test. In this case, a larger pool of items will need to be developed.

Larger item pools will probably be needed if the test is diagnostic in nature. For example, a test designer will need to develop more items if he or she is testing six objectives with five items per objective than if the test designer only samples one or two items across the six objectives.

The method of presenting test items also impacts the size of the item pool. For example, if test specifications call for three forms of a test with no item overlap,
then a larger item pool is needed than would be required if items can be randomly selected from a pool and some overlap among tests is considered acceptable.

*Test-taking Policy.* When determining testing requirements and test specifications, the test-taking policy must also be carefully considered. Will a student be allowed to retake a test once he or she has failed it? If retesting occurs, is it to be the same test, or a parallel or equivalent form? It should also be determined how many times the student will be allowed to take the test before remediation or administrative action outside the CBI environment takes place. These issues impact not only the number of items which need to be developed (see item pool discussion) but also influence the manner in which the test is programmed onto the computer.

Another issue related to test-taking policy is the method in which it is decided that a student will take a test. It may be determined that students should have the option to take a test whenever they feel ready to be tested. Or, tests could be made available only after completion of each unit of instruction, with all students being required to take the same test at that time. These issues not only have an impact on the test-taking policy, but also have a large effect on the evaluation of the courseware. Tests administered throughout the course of instruction, or administered at student request, will create situations where gain scores and item statistics will be difficult to compute and analyze (Sarvela & Noonan, 1988).

*Item Tryout and Analysis.* There are several problems associated with the use of CBT when attempting to analyze the quality of the tests. Because of the unique nature of testing in CBT scenarios (e.g., random selection of items from a pool), it is possible that all students will not be tested on the same items (therefore, the students will not have taken the “same” test) and that the students did not experience the same instructional treatment (because of branching variations). In this situation, meaningful item analysis, reliability and validity measures, and pre–post gain scores are difficult to compute and interpret (Sarvela & Noonan, 1988).

**Test Security**

Test security is most often concerned with the access students have to a test. For a variety of reasons (e.g., evaluation of pre- and posttest gain scores, reducing student cheating), it is desirable to limit student access to tests. The following issues are discussed in this section: student access to tests, test preview, and test review.

*Access Limitations.* The most important consideration in test security is deciding when students can access tests. One possibility, though perhaps the least likely, is to allow the student to take any test at anytime, with no mastery criteria
and no special access controls. A more typical procedure is to: (1) Allow the student to take pretests only before entering a lesson or unit of instruction and (2) Limit access to posttests to students who have completed the lesson or unit. In other words, pretests can only be taken before any instruction and posttests taken only after all components of the lesson or unit have been completed. There are variations on this strategy, but implementation of the variations could be difficult to achieve because the programming would become more complicated and expensive. In addition, different approaches jeopardize evaluation efforts; for example, if students can take pretests or posttests at any time, an evaluation strategy that uses gain or change scores is thwarted by the inequality of the pre- and posttest groups. The evaluator cannot reasonably assume that students within a posttest group have had the same treatment or that students in a pretest group have had equal exposure to the instructional material.

Once decisions have been reached on when the student can access a test, specific coding procedures for limiting access have to be implemented. There are generally two options: (1) internal coding flags or (2) passwords. With internal coding flags, the code is usually written such that access to a test is dependent upon a flag being “ON” (set to 1) or “OFF” (set to 0). The password option requires the student or proctor to enter a password once a test point has been reached. Passwords require greater involvement and monitoring by a proctor or tutor, and, hence, are usually only feasible in large-scale CBI.

Test Preview. Some curriculum specialists argue that it may be instructionally beneficial to allow students to preview a test before starting a lesson, or, having completed a lesson, before taking the test for credit (i.e., Gebhardt & Munn, 1985). With the former, students can see exactly what will be expected of them; the test preview functions somewhat like a presentation of the lesson objectives. With the latter, students can self-assess their readiness for the test and, if needed, re-enter the lesson for extra study. A disadvantage of test preview lies in the potential compromise of the test items. If the items are written as a representative (perhaps random) sample of a domain of knowledge, then access to the items can bias the test results. If the student only studies to answer specific questions, then there is no assurance that whatever learning occurred will generalize to the broader domain of knowledge.

In addition, programming issues arise. Extra programming will be needed to keep track of when the students are in the “test” mode and when they are in “preview” mode. This extra programming would have to disable student-input, scoring, and feedback functions. Also, if the number of test attempts is controlled, then extra programming might be needed to bypass or disable the counter for test attempts.

Test Review. After a student has completed a test, he or she should be presented with the test results. This could be as simple as notification of pass or
failure, or it could include a listing of the number of items attempted, number correct, and mastery criteria. Still another option is to allow the student to review the actual items, with notation of which ones were answered correctly and incorrectly. The review might also include the correct answer and remediation for incorrect responses. Such a review can be beneficial to students in helping them pinpoint specific problem areas. The danger is, again, in item contamination. If the identical items are used in a second attempt at the test, then the student may learn how to answer specific items without having mastered the entire domain of knowledge. Allowing test review with item-level feedback is more defensible if parallel forms of a posttest are available.

The particular review options that one provides will influence the complexity with which the test is programmed. For example, if one allows students to review actual items with corresponding correct/incorrect item feedback, then it might be necessary to create and track extra scoring variables to redisplay the students' answers, the item scores, and the corresponding feedback. In addition, extra programming might be needed to disable student-input, scoring, and counters for test attempts—so that the review does not inadvertently end up as another fully scored test attempt.

Item Presentation

The manner in which items are presented to students in CBT situations is an important implementation decision. This section identifies and discusses the following CBT item presentation issues: access to test directions; item skipping; random, sequential, and adaptive item selection; screen display conventions; time-out; feedback; student discontinue criteria; and log-off procedures.

Access to Directions. Test directions and sample items are standard elements in paper-pencil tests. Students are presented with the directions and sample items at the start of the test, and they can review them at anytime during the test. Special actions must be taken by test designers to afford this same option to students when using computer-based testing. Directions and sample items can still be presented at the start of a test, but special keys or functions might have to be programmed in order to enable access to the directions and sample items once the student has begun to see test items. An icon or line of text could be displayed on the screen (perhaps on a bottom menu line) throughout item presentation to remind the student of the keystrokes needed to access the directions and sample items. Sample items become especially important in CBT because students must be told how to answer each item type. For example, a multiple-choice item could require students to enter the letter of the option they choose and then press "ENTER" or "RETURN." Or, students may have to TAB among the options until the cursor is beside their answer and then press "ENTER" or "RETURN" to register their response. Coding must be written so that once students access
directions, they go back to the same item upon returning to item presentation. Test designers have to plan for cases where students are being presented items and need assistance in remembering how to respond to a particular item type. Nothing could be more frustrating to a student than to know an answer to an item but be unable to register the response in the computer.

*Item Skipping.* Test designers must decide whether or not students will be able to preview or skip items once they are taking the test. A common student test-taking strategy for paper-pencil tests is to: (1) Preview the items to gain an idea of the scope and content of the test, (2) Go back and answer the “easy” items, (3) Allot the remaining time among the items which require greater thought and study, and (4) Review the answers at the completion of the test. Designing CBT to accommodate this strategy can be a programming nightmare. If skipping is allowed, then test designers must decide when responses are scored. If the items are scored immediately (before presentation of the next item), then precautions will have to be taken about coding “null” responses (when a student elects to skip an item). The test designer must determine when such a null response will be scored as incorrect. Also, the designer has to decide upon a key or key function that students use to skip an item. This again must be included as an icon or line of text to remind the students how they can skip items.

Another consideration relates to how skipped items are recycled. If a student gets to the end of an initial item cycling, and has skipped items during the test, the student should receive a prompt concerning the unanswered items and instructions on how to move to and answer the skipped items. Also, the designer has to decide if all items or only the skipped items will be seen again. If all items are seen again, the designer must decide if students can change answers.

Options of allowing item preview or skipping also relate to item selection strategies. If items are selected randomly from a pool, then all of the random selection must occur before item presentation begins. A decision to use item preview or skipping impacts on other presentation decisions. For instance, one could not utilize computer-adaptive testing (CAT) if item preview is used. With CAT, items are selected on the basis of the student’s responses to previous items; the computer is programmed to select the item that will provide the most information about the student’s level of performance. CAT relies upon a response to each item as it is presented, therefore item preview cannot be used with CAT.

*Item Selection.* Decisions must be made regarding the procedures for item selection. Several options are open to the test designer. Items could be selected randomly from a pool. They could be presented sequentially, as in a paper-pencil or individually administered test. Or, one could use adaptive testing, where the item selection depends on the student’s success or failure on previous items. Each strategy has its own advantages and disadvantages, and a decision to use one strategy impacts other design decisions.

If items are selected randomly from a pool, then complications arise if the test
designer wants to allow item preview or skipping. To accomplish this, all of the items would have to be randomly preselected at the start of the test. One could not randomly select items as they are administered. And, if test review is allowed, the courseware must be coded to store in memory the particular items chosen for each student. If different items are to be seen on a retest, then code must be written to "lockout" those items seen on the first administration.

Aside from these coding complications, there are two serious conceptual problems with random item selection. The first problem is the implicit assumption that the items administered to one student will be equal in difficulty to items that are presented to another student. Imagine that a pool of items has an average \( p \)-value (difficulty index) of .80 and a standard deviation of \( p \)-values of .12. For most courseware environments the item pool is relatively small, so also assume that there are 15 items in the pool and 5 will be selected for administration. If the test is going to be fair to students, the items that one student sees should be comparable in difficulty with the items on which another student is tested. In the long term, random selection will produce comparable tests, but one certainly would expect that at times one student would receive all of the easier items and another would receive the harder items. The frequency with which this occurs will depend on the degree of variance in item difficulty. It is clear that with a random selection of items, problems occasionally will arise concerning test difficulty. One possible control for this undesirable effect is to randomly select items within strata of difficulty. For example, 1 item could be randomly selected from the \( p \)-value range of .90–1.00, three items from the range of .80–.89, and 1 item from the range .00–.79.

The second conceptual difficulty with random item selection relates to compromises on program and test evaluation. If students see different items it becomes extremely difficult to compute item and test statistics (e.g., total score, point biserial, KR–20). The major problem is that there is no sensible total score. With random item selection, a total test score only becomes defensible for item analysis if every item is of equal difficulty and equal discrimination (otherwise, the students have not seen the "same test"). And, pretest and posttest comparisons presume parallel forms of a test (equal means, standard deviations, reliabilities, and validity coefficients). With random item selection, parallel test criteria can only be met if each item in the test domain pool is of equal difficulty and discrimination, a highly improbable condition (Sarvela & Noonan, 1988).

Many of the problems mentioned disappear if items are presented in sequence. Usually, a sequential item delivery is used with a fixed-length test; a set number of items are presented in a particular order. This format is most closely analogous to a paper-pencil test. Total test scores fit well into the logic of test theory and less concern can be given to establishing equal item difficulty and discrimination. Also, fewer items are needed and the test designer is not forced to choose a particular option on other decision points (e.g., item preview, back-up, answer changing, when scoring occurs, etc.).
Parallel advances in computer technology and item response theory (IRT) (Green, Bock, Humphreys, Linn, & Reckase, 1984; Jaeger, 1987; Lord, 1980) have generated a considerable degree of interest in CAT. In CAT, an ability estimate is computed after each item is presented and answered by the student (Weiss & Kingsbury, 1984). This ability estimate is used to select the item that will produce the most information for the next ability estimate (technically, an item that the estimate predicts the student will have a 50% probability of answering correctly). Items are presented and ability estimates are computed until a discontinue criterion is reached (usually an error limit associated with the ability estimate). The primary advantage of CAT has been in a reduction in testing time (Ward, 1984). Interestingly, CAT has not been implemented in CBT–CBI environments. The primary hindrance to its use is that the item parameters that are needed require extensive item tryout and analyses on very large samples. This kind of test development effort is normally not supported in traditional courseware development environments. IRT also assumes that items are unidimensional (the items all measure a single underlying attribute). For many CBI environments, training is aimed at multiple objectives; the resulting tests are, by design, not unidimensional.

Also, a decision to use CAT forces, by default, the test designer to choose particular options at other decision points. One cannot allow test preview, item preview or skipping, or back-up and changing of answers. Items must be scored immediately and a CAT discontinue criterion must be used.

**Screen Display Conventions.** Screen design is an important consideration in all aspects of CBI courseware development (Sweeters, 1985) and should be carefully considered when developing CBT programs because presentation of items in traditional (paper-pencil) formats differ significantly from CBT item presentation. For instance, a “matching” test item can usually be placed on one page of a paper-pencil test. It may be difficult to fit the same matching item on one computer screen due to display constraints. Because of the “terseness” that is required in CBT development, the test designer could be limited in the types of items that can be developed.

**Time Out.** One of the often-cited advantages of CBI is that the computer is infinitely patient. The computer will wait for an input without generating the social pressure to respond that often occurs in a traditional classroom setting. In certain test settings, however, it is often desirable to set time limits for responding to individual items. If a time limit is set for the test as a whole, then time limits on individual items help the student move through the test. This would be especially important if item preview or skipping is now allowed. Also, time limits provide a safeguard against students’ simply leaving the terminal and having the item(s) open for viewing by other students. The difficulty is in deciding when it is reasonable to conclude that the student has left the terminal. One
alternative is to select an amount of time, say 180 seconds, and then prompt the student to respond if no response has been made in the allotted time. The prompt could be “Please answer now!”; the student could then have additional time to answer, say 30 seconds, before the test is discontinued. If no time limits are set, the test designer risks having a student sit for extended periods of time without answering.

**Item Feedback.** One of the primary advantages of CBI is the potential for immediate feedback during a lesson. As students answer practice questions, they can receive immediate information on their answers. Given the instructional advantages to immediate feedback, there is a great temptation to provide item feedback during a test. From an instructional perspective, it makes perfect sense to correct an error during a test. (For purposes of scoring, an incorrect item could still be counted wrong.) However, there is a danger of contaminating future items if all items are not totally independent. That is, the student could use the feedback as an aid in answering future items. The reply from the instructional perspective is that it really does not matter where the students learned the material, the lesson or the test, as long as the students show that they have mastered the material.

The research of Wise and his associates suggest caution in using item feedback. In a study with elementary schoolchildren (Wise & Wise, 1987), they found that item feedback on a computer-administered test increased state anxiety among high-achieving math students. In another study they found item feedback to interact with item arrangement (Wise, Plake, Eastman, Boettcher, & Lukin, 1986); item feedback did not affect anxiety or performance level when items were presented in an easy-to-hard order, but anxiety increased and performance decreased with random presentation of items. Other research on item feedback is mixed; some have found positive effects (Morris & Fulmer, 1976; Rocklin & Thompson, 1985), while others have found debilitating effects (Strang & Rust, 1973). In summarizing the research, Wise and Wise (1987) go so far as to say that “the use of such feedback in computer-administered tests is not recommended until its effects are better understood” (p. 19).

Another factor to consider is student motivation. If a student is consistently answering items incorrectly, the negative feedback can be detrimental to motivation on future items. Likewise, a series of correct-answer feedbacks can promote greater motivation in future items. The danger here is the differential effects of item feedback across high and low achieving students. Most, if not all, individually administered tests do not include item feedback in their instructions. Moreover, test directions often caution about the motivational dangers of giving subtle cues about the correctness of the student’s responses (Wechsler, 1974).

**Discontinue Criteria.** In a fixed-length test, the student is presented with all of the items on a form of a test (i.e., all students see all of the 40 items on a test).
The computer can allow the test designer to stop testing once the student has passed or failed the test. If a test has 40 items and 30 has been set as the passing score, the computer could be programmed to discontinue the test once the student passes (provides a 30th correct answer) or fails (provides an 11th incorrect answer). Discontinue rules could be set up according to a fixed number of correct or incorrect responses, a percentage correct, or consecutive right or wrong. For CAT, the discontinue criteria are normally some limit or error associated with an ability estimate. If the test is to be discontinued early, the test designer must specify and program the decision rules.

Discontinue rules are often contraindicated if the testing is diagnostic in nature. There might be cases where entire sets of items must be presented in order to assess mastery of subskills. For example, suppose a 30-item test covers 6 objectives (5 items per objective), and the designer has specified mastery scores of 4 out of 5 items for each objective. If the test is stopped before information is collected on the last set of five items, the system might not have the information to route the student past or into the corresponding segment of instruction.

If discontinue rules are used in conjunction with backing up and changing answers, the student would have to be cautioned about casual answer changing. It would be possible for a student to back up, change an answer, and then suddenly satisfy a discontinue rule for early failure. In other words, a student could change a correct response into an incorrect answer and then receive notice about failing a test.

If discontinue rules are used, the designer must be wary of the possible compromises to program evaluation, mentioned earlier under item selection. One needs a comparable or sensible total score in order to compute item statistics or use gain scores in program evaluation and discontinue criteria may make these calculations difficult.

Finally, the designer will have to decide whether or not the students will be informed of the discontinue criteria. Normally, students would be told up front. However, if complicated discontinue rules are used, the designer might opt to withhold an explanation of the criteria.

**Student Log-off.** The test designer will have to address the difficult issues related to student log-off in the middle of a test. If a student leaves in the middle of a test, will the test be failed? Will only the last item seen be counted wrong? Will items seen but not answered be counted wrong? What sort of warning will the student receive? Which items will the student see when he or she returns to the test? Will the counter for correct answers be reset to 0 when the student logs back on to the test? Will a parallel form of the test be provided on the next attempt? Will the student be allowed to change answers given prior to the early log-off?

The simplest procedure would appear to be counting the test as failed and providing a parallel form upon returning to the test. When a student tries to log
off during a test, he or she could be told that the test will be failed and then asked if they want to return to the test. In this case, programming is complicated because the normal log-off has to be intercepted and a procedure for returning to the test, without penalty, must be coded. If there are negative consequences to logging off, then students should be given some idea of time estimates for the test before they enter the test.

Response Capturing and Scoring

The final cluster of issues to be discussed concerning CBT implementation decisions are response capturing and scoring considerations. The CBT designer must decide when answers are to be registered, if backing up and changing of answers will be allowed, how error trapping will occur, how response latency analysis will occur, and finally, the types of response analysis and scoring that will be used.

Answer Registration. For almost all interactions with a computer, the student must somehow signal the end of an input to the computer. Normally, ENTER or RETURN keys are used for this purpose. Regarding answers to test questions, there must be a procedure for the students to mark the end of their answers. For single character responses (e.g., true-false or multiple-choice items) the system could be set up to accept the single character input and then proceed to the next item. Alternately, and perhaps preferably, the student would make a double keystroke; press a letter for the answer, and then press RETURN or ENTER to register the response and trigger the next item. The advantage to the double keystroke response is that accidental or stray keystrokes are not counted as inputs. A designer could conceivably even offer greater student control by presenting “Are you sure? y/n” after the “answer and ENTER” input. These additional safeguards could become more of a nuisance than they are worth, but they might have application if more than one item is shown on a page (e.g., a matching exercise).

Backing up and Changing Answers. In paper-pencil tests, students often go back to items they have already answered and change their responses. A recent review of research (Benjamin, 1984) suggests that, more often than not, the answer changing is from an incorrect answer to a correct answer. If these features are going to be afforded to students in a CBT environment, then complications will arise in coding. The designer has to provide for the student returning to the appropriate item after the back-up has been completed. Also, procedures for determining exactly how the back-up is accomplished need to be developed and coded. Will a designated key back-up items one-at-a-time? Or, will a request for back-up produce a menu in which the student is prompted to enter the number of the item to which they want to return?
There are some arguments for disallowing this feature on CBTs. If there is extensive routing within the test, as in CAT or diagnostic testing, then items must be scored as they are answered. What happens if a student has been routed to a particular subtest because of failure on some routing test and then the student opts to back up and change an answer on the routing test? With CAT, answer changing seriously complicates the algorithm for generating ability estimates between item presentation: a test-wise student could notice that the items are getting easier and decide to go back and change earlier answers. Also, students could try to look back at items continually in order to get clues that help them answer other items (e.g., help eliminate distractors on a multiple-choice item).

Finally, it is conceivable that the test designer could permit students to back up and see earlier items but not allow them to change the answer. These two features can be kept distinct. However, it could be overly frustrating (perhaps unfair as well) for a student to back up and find an error, and then not be allowed to change the answer.

**Error Trapping.** Computers are usually programmed to expect particular types of inputs. The most simple cases would be inputs of numerical and string variables. If the system is awaiting an input of a numerical variable and the student types a letter, the program could crash. Programmers usually include error trapping routines to avoid these problems. If the system is awaiting a numerical input, the system is programmed to determine if the real input is numerical before it tries to assign the input to the predesignated variable label. Similarly, test designers need to include error traps to make sure that the response is of an appropriate type or within a particular limit. For instance, if a multiple-choice item has the options of “a,” “b,” “c,” or “d,” then the program should ask for a reanswer if any other input is made. Likewise, true–false items could be programmed to only accept inputs of “t” (true) or “f” (false). If a number is expected, then letter inputs should not be accepted. Error traps also guard against accidental keystrokes if answer registration uses a single keystroke. If the CBT system does not already provide these error traps, then the test designer or programmer must code for them.

**Response Latency Analysis.** Response latency is the time between presentation of a test item and the student’s response. The test designer should decide if response times are going to be collected and, if so, how the data will be analyzed. Latency analysis has been proposed as a promising area for computer-based testing (Space, 1981). One would expect that longer response times are associated with “uncertainty” in achievement and ability testing; for personality testing, longer response times might be expected for items that are more “ego-involved” and, hence, generate emotional blocking. Dunn, Lushene, and O’Neil (1972) conducted early research on the feasibility of latency analysis in personality assessment. They administered the MMPI via computers to 165 college
students. Response times were averaged across students and entered as the dependent variable in stepwise multiple regression analysis. Predictor variables included a number of item characteristics, such as item length, social desirability, ambiguity, tense, and voice. They found that item length accounted for 47% to 58% of the variance, while three other variables—ambiguity, social desirability, and social desirability dispersion—accounted for only an additional 3% to 8% of the variance. One difficulty in interpreting the research of Dunn and his associates is that they did not look at intraindividual differences. One wonders what results would have been found if response times were analyzed for individual examinees—where psychological blocking on particular items would not be lost in aggregated data.

Using response latency analyses in computer-based instructional testing poses additional problems. If latency analyses are going to be conducted, then the following cautions are in order. First, latency analyses presumes a rather high degree of vigilance on the part of the students. This might not be as much of a problem for stand-alone ability and personality tests, where testing times can be rather short. But, for large-scale computer-based training, students could be at a terminal for several hours at a time. Variations in attention during longer sessions at a computer could produce highly variable response times, and the test designer should be cautious about overinterpreting response latencies. What if a student sneezes or helps out another student at a nearby station?

Secondly, latency analysis requires a very simple response format, such as a single-letter input. It would be very difficult to interpret response times for constructed response items, because additional time must be allowed for typing in an answer. Students could arrive at answers quickly and then have their latencies misinterpreted because of slowness in typing in the answers.

Finally, response time can be easily confounded with reading speed, reading comprehension, and item length. The test designer has to be cautious about decisions or judgments that are made on the basis of a short or long response time to a particular item.

Latency analysis might be appropriate for learning objectives that focus on teaching students how to perform already learned skills more quickly (e.g., drill-and-practice exercises). If students have learned a skill to the point of being “correct, but hesitant” (Rosenshine & Stevens, 1986), latency analyses would be entirely appropriate for measuring learning objectives that are designed to bring the students to full automatization of the skill.

Response Analysis and Scoring. Once a student has registered an answer and the input has passed the error traps, the system must analyze the input for correctness and score the item accordingly. Response analysis can be the most complicated coding aspect of CBT. Response analysis is least difficult in a selected-response mode and most difficult in a constructed-response mode. Checking the input for a match to “a,” “b,” “c,” or “d” (even upper- or
lowercase) on a multiple-choice item, or “t” or “f” (again upper- or lowercase) in a true-false item is relatively straightforward.

Constructed responses require considerably more complex analyses. Decisions must be made about handling such things as upper- and lowercase, spelling errors, punctuation errors, and extra spaces in the input. Once the designer decides upon how these elements are scored/analyzed, the code must be written for the actual analysis. The first major difficulty arises in trying to detail all possible correct answers. As an example, consider the following constructed-response item: “What are the two steps in preparing the XYZ radio tuner?” Suppose that the two steps are: (1) turning the power on and (2) turning the mode selector dial to “tune.” Further suppose that the order of these steps is not important. The following are some correct answers:

- Turn it on and turn the mode dial to tune.
- Set mode switch to tune and then turn on the power.
- First you press the power switch, then you rotate the other dial to “tune.”
- I think you flip the power switch and turn the dial selector to tune.

The list could obviously go on ad infinitum. The second major problem is in programming time. Imagine, without some kind of artificial intelligence, how much programming is involved for even a partial subset of all possible correct answers. If diagnostic tests are used, then extra code is needed for error analyses.

A second issue in response analysis and scoring is deciding when scoring will occur. In cases of diagnostic or adaptive testing, scoring must be done before the next item is presented because the student’s history of successes and failures is used to route the student to particular subtests or items. If discontinue criteria are utilized, the system must keep a running count of correct and incorrect answers. If early student log-off is allowed, it might be advisable to score items immediately so that response data are not lost with the log-off. If item feedback is provided immediately, then the item must be scored immediately.

There are also times when it would be advisable to delay scoring until the test is completed. The interests of test security might dictate that scoring be delayed to the end of the test. This is more likely to occur in microcomputer configurations involving floppy diskettes; enterprising students might figure out a way to retrieve correct answers from the diskette. In a response to this potential problem, test designers at Psychological Corporation created an item presentation diskette and a scoring diskette on a microcomputer version of the Ohio Vocational Interest Survey (OVIS, 1984). The product is configured in such a way that the student never handles the scoring diskette. The presentation diskette (called the Survey diskette) presents the items and stores responses in a file. The Scoring diskette, which is used exclusively by the test administrator, reads the file, scores the instrument, and writes the scores onto a student file.
It might also be advisable to score items at the end of a test if the student will be allowed to back up and change answers. If there is going to be answer changing, scoring immediately could result in a lot of extra or wasted processing.

Finally, the test designer has to assign points to items. Usually, one point is given for each correct item; however differential weighting is possible and sometimes desirable since research suggests that it increases the reliability of tests (Haladyna, 1984). If weights are used, the programming usually involves another variable (a weighting variable) that is applied to the items.

SUMMARY

Although the computer has a number of potential applications in the testing environment, the costs associated with the design and development of computer-based tests are quite high. When the computer is selected as the testing delivery system, careful analysis of the implementation issues and questions must take place. This chapter has identified four decision areas which need to be addressed when designing CBT programs as a part of computer-based instruction courseware development efforts: test construction, test security, item presentation, and response capturing and scoring. A checklist which can be used during the CBT development effort, covering these major decision areas, appears in Appendix A.

ACKNOWLEDGMENT

The authors would like to thank William Sweeters, of Ford Aerospace and Communications Corporation, for his helpful comments on an earlier draft of this chapter.

REFERENCES


APPENDIX A

Decision Points in Developing Computer-Based Testing Programs

A. TEST CONSTRUCTION
   Diagnostic or mastery tests
   Routing
      within the test
      within the courseware
   Types of learning objectives
   Item types
      selected-response
      constructed-response
   Embedded or block tests
   Size of item pools
   Test-taking policy
   Item tryout and analyses

B. TEST SECURITY
   Access limitations
   Test preview
   Test review

C. ITEM PRESENTATION
   Access to directions
   Item skipping (preview)
   Item selection
      random
      sequential
      adaptive
   Display conventions
      format
      color
      headings, titles
      highlighting
      menus and icons
   Time out
   Item feedback
   Discontinue criteria
   Student log-off

D. RESPONSE CAPTURING AND SCORING
   Answer registration
   Backing up and changing answers
   Error trapping
Response latency analysis
Response analysis and scoring
  selected-constructed response
  when scoring occurs
  points per item