The Structure and Use of a Test Generating System Designed to Facilitate Individually Paced Instruction

Roger M. Palay
Instructor, Exact Sciences
Washtenaw Community College
Ann Arbor, Michigan 48106

Abstract:
The structure and use of a test-generating system having the following parameters is described.

1. Each test is a ten item multiple choice instrument where each item has five (5) alternatives.
2. Each test is generated from a relatively small file (most often less than 150 card images).
3. Although the file contains only five (5) disjoint tests, over nine million different tests (each with its own set of correct answers) can be generated from it.
4. The correct answers to each test are stored directly on the test (in coded form).
5. Test responses are machine correctable and such correcting need not be done on the system that generated the test.
6. The reliability of different forms of tests generated from one file is theoretically high (no statistical evidence is available to support a claim of high reliability).
7. The grading program produces a total score as well as giving the correct answers for all items on that test.

One is never sure if the cry for individualizing the pace of instruction is the sophomoric scream of idealists or the dying echo of a disappointing experiment. That such individualization has been tried and been discarded seems to affect neither the volume nor the frequency of new calls-to-arms. The harmony of human development, and human nature, with such a scheme has appeared doomed to the discordant march of administrative difficulties (most of which have been the result of trying to conduct each player instead of leading the orchestra). In general, those difficulties fall into the following ranks:

a.) Generating a sufficient number of equivalent tests to allow test and re-test of all material.
b.) Providing test security over all test forms.
c.) Administering and grading multiple forms of tests, which probably have different answers.
d.) Providing sufficient feed-back of correct answers to students.

The remainder of this paper describes a computer-based system that was developed to diminish the administrative difficulties of individually paced instruction.

The central idea of such a system is that students will move through a course of study at their own pace. In order to insure both progress with understanding and the existence of a feed-back loop for catching and correcting misconceptions, students are required to "pass" a test on every section of the curriculum. Students take these tests when they feel that they have mastered the material in that section. Test-taking is done during class and office hours, but other arrangements could be made. There is no limit to the number of times a student may attempt a section test, but each time a test is taken it must be taken on a "new" form of the test. (For all practical purposes the system produces an unlimited number of different forms of each test.) Answers are marked on the test and, depending upon the scoring procedure, either the test is handed in or the answers and other test information are transferred to an optically marked data card that is handed in. In either case, the student's answers are used as input data

to a program that does the grading. The grading process not only totals the number of correct responses, but also indicates the correct answers. All tests are graded by the following class period; immediate grading would be possible if a terminal were available. The instructor's contact time is spent collecting student responses (1% of the time), posting graded results (1% of the time), and helping individual students with current problems that they are having with the material (98% of the time).

Even a quick consideration of the system just described should reveal that its keystone is the generation of a large number of machine gradable section tests. The decision to create a computer program to aid in such a generation is not particularily novel; it isn't even insightful. Test, worksheet, and problem generating programs have filled the educational program library for years. Those programs tend to fall into one of two categories. One genre of generating programs selects (possibly randomly) questions from some master file; the other uses random numbers to vary quantities within a problem whose form is specified as part of the generating program. Either version can exist at almost any level of sophistication. Also, either form can be adapted to "on-line" and "off-line" environments. "On-line" or interactive use requires numerous terminals and ports, and a computer system that is dependably "up". The necessary number of terminals is a function of the number of students to be served per hour and the length of time that each student is given to solve problems. Five minute sessions may be adequate for drill-and-practice of arithmetic facts, but more complicated problems naturally require more time to solve. To give even thirty (30) students an average of twenty (20) minutes on the machine each class period (50 minutes) would require twelve (12) wholely dedicated terminals. Even with terminal costs declining, such an expenditure would be hard to justify. This is particularly true when one considers that the terminal is inactive from the time it presents the problem until the time that the student enters a response.

In an "off-line" mode of operation the user disconnects from the system after the problem set has been produced. In an hour or less a single low-speed terminal can produce numerous forms of a worksheet or test for later duplication. If a line-printer is available then one can create reams of questions in a relatively short time. These forms can be used as they are needed, possibly days, weeks, or even months after they were produced.

The whole "off-line" operation has both cost-effective and pedagogic appeal until one considers the problem of grading the work. In an "on-line" mode the correct answer is readily available to the program that produces the problem. A student can enter a response and it can be graded immediately. In an "off-line" mode, each form of a worksheet or test may have a different answer key. Grading requires either the re-calculation of answers or their being looked up through the use of some indexing scheme. Re-calculation by hand is a most unappealing solution. Re-calculation by machine involves the development of specialized programs and the re-entering of the randomly generated data. Keeping physical files of answer sheets becomes an enormous clerical task. Keeping records of answers on the machine involves the development and use of highly sophisticated, often machine dependent, file manipulating capabilities. Furthermore, if such files reside on one computer system then that must be the system that is used for grading.

The difficulties inherent in an "off-line" system can be controlled however; and this can be done without compromising its basic economy. In the first place, the type of question to be generated can be limited to one form. Consider multiple choice items. They are among the easiest to grade. They are one of the more powerful and versatile forms of questions. They allow statistical evaluation of item-difficulty. Although they are not the easiest question form to produce, they are readily modified and improved. The distractors can be numbers, letters, words, phrases, even entire sentences. The stem of a multiple choice question can contain any idea from a simple calculation through an analysis of abstract concepts.

One is still faced with the problem of storing the correct answers for each test. One solution to this is to print the answer list on the generated test itself. Naturally the answers will have to be coded. A method for such coding is given below, but at this point consider the beauty of such a procedure. Answers printed on a test (in coded form) are as permanent as the test itself. The correct answers are available for grading on any system that has been programmed to decode them. The link between a test and its appropriate set of answers is obvious and indestructable. And, there is no requirement of a filing system with its associated clerical and/or computer

headaches.

A description of the coding system must be prefaced with the following remarks on its origins. This system for individualizing the pace of instruction was used by over one hundred students a day, three days per week. Students took tests on anywhere from one to four sections each class period. That volume dictated a batch grading of the tests. The University of Michigan Computing Center provides a sixteen (16) column optically marked data card. This sixteen column limit forced a decision to make every test ten items long because those ten items and a four character test code and a two character student code completely filled the data card. There was no other reason for limiting the test to ten items, although that proved to be more than sufficient to completely cover all material in each section of the curriculum.

The actual coding system used enhanced the advantages listed above. The method was to number the choices for each question from one to five. One could then view the correct answer list as a ten digit number. The value of each digit of that number was decreased by one. The result was a ten digit number, each digit of which was in the range of zero to four, inclusive. That is the form of a base five number. Expressed in base ten it will be between zero and 9765624 inclusive.

Now it so happens that there are at least fifty-six (56) separate characters that can be key-punched, entered via teletype terminal, marked on optically marked cards, and printed on the line printer. Those fifty-six characters can be thought of as digits in a base fifty-six numeration system. Furthermore, any base ten number between zero and 9834495 can be represented as a four digit base fifty-six number. That is precisely the coding that was used. Each test carried with it a four character code that actually held all ten of the correct answers.

Students would take a test, mark their answers on it, copy them onto an optically marked data card, add their two character student code, and finally enter the test code from the test itself. An entire deck of these cards was then read as data by a grading program. It converted the base fifty-six number (the test code) back to base five, added one to each digit and, therefore, had the list of right answers. It was simple then to compare these right answers to a student's responses, calculate and print the score, along with the student's answers, and the right answers. In practice this worked better than expected. During the last month of the course over 250 tests were being corrected in this fashion each day. Machine time for correcting was less than 1 second, elapsed time for correcting and printing results was under ten minutes. The grading program was written in ALGOLW and was run on the University of Michigan's IBM 370. Additionally, an interactive program, written in BASIC and running on Washtenaw Intermediate School District's Hewlett Packard 2000F, was available for immediate checking of individual tests.

A grading system is useless, however, if there is no procedure for generating tests. The format has been chosen: a ten item multiple choice instrument with five alternatives for each question. That format does not lend itself well to the use of random numbers to vary quantities within the questions. Rather, random numbers can be used to select questions from a master file generated manually by the user. This, too, is not a novel idea. Many such procedures exist. Unfortunately, most of them virtually demand an extremely long list of possible questions from which to choose A single good question is hard to create, how much more so is a long list of them. Other issues must also be considered. For example, will the tests so generated be equivalent in difficulty and scope? And, can the user be certain that all of the major topics in the curriculum will be covered?

The following master file structure goes a long way toward resolving these issues. A master file consists of ten (10) sets of five (5) questions and five (5) answers, where each set of questions is designed to test one concept. Within a set, the five possible answers must be good choices for each of the five questions, but the correct response may (and should) change with each question. Although such a construction may seem prohibitive, experience indicates that it is not significantly harder to produce one such set of five questions than it is to produce one <u>good</u> multiple choice question. In fact, the general procedure is to generate one form and then modify it four times to get a total of five versions. That paradigm, along with the requirement that the same five answers serve as distractors for all five questions in the set, facilitates an equivalency of difficulty and scope within each set. As a point of information, it should be noted that a complete master file (usually under 150 lines long) takes about two (2) hours to compose and three more to enter into the machine (including proof-reading and editting).

Obviously, the resulting file can produce at most five (5) disjoint tests. But, a full 9765625 different tests are possible through the random selection of one question from each set. This should be sufficient for most situations. An entire class can be given the "same" test and yet no two tests need be identical, and no two answer lists need match. A different version of the "same" test can be given every day of the school year. For all practical purposes there is no limit to the number of times that a student may obtain a new section test. One might expect that a student, after numerous re-takes, would become familiar enough with the master questions (there are only fifty, total) that he could pass the test solely by the recall of answers. Strangely, just the opposite is true. The similarity of questions within a set causes so much learning-set interference that the more often a student repeats a test, the more thoroughly the student will work and check the problems.

In summation, it can be reported that the system, as described above, did work. Questions were composed, master files were created, tests were produced and taken, data cards were marked and graded, students did pass and progress, and, most importantly, they did learn. The very activity of taking a test was a learning experience. It would be unfair to compare the performance of these students with the performance of those who took the course in a lecture format. Students in the self-paced sections spent at least two and a half hours a week doing the course work -- those in lecture sections were expected to fill seats, an activity that was done with declining regularity. Clearly, some statistical measures are needed to substantiate the statement above. Although no statistical study was done in the initial phase of designing and implementing the system, such a study should be an integral part of any future use of the procedure.

Appendix

Below is a reproduction of the data card discussed above. The card was designed to be used to code FORTRAN programs. It was adapted for use in this testing procedure by having students mark their answers to questions 1-10 into columns 1-10, their student code into columns 11 and 12, and the test code into columns 13-16. The "statement number" area of the card was used to indicate chapter and section numbers. This particular card is coded for a test from chapter 4 section 18; the student's answers are 1234514352; the student's code is EN; and the test code is A%/#