DOCUMENT RESUME

ED 111 356 IR 002 404

AUTHOR McClain, Donald H.; Wessels, Stephen W.
TITLE IPSIM: Additional System Enhancements Utilized in a Chemistry Application.
PUB DATE Jun 75
NOTE 17p.; Paper presented at the Conference on Computers in the Undergraduate Curricula (6th, Fort Worth, Texas, June 16-18, 1975); For related document see IR 002 399
AVAILABLE FROM Entire Proceedings; Ted Sjoerdsma, Treasurer, CCUC, 1248 Lindquist Center for Measurement, Iowa City, Iowa 52242 ($10.00, Checks payable to the University of Iowa)

EDRS PRICE MF-$0.76 HC-$1.58 Plus Postage
DESCRIPTORS Academic Records; Chemistry; *Computer Oriented Programs; *Computer Storage Devices; Individual Tests; Multiple Choice Tests; Performance Tests; *Recordkeeping; Student Records; Student Testing; *Test Construction; *Testing Programs; Test Scoring Machines
IDENTIFIERS IPSIM; University of Iowa

ABSTRACT

The University of Iowa has been involved with the development, implementation, and operation of computer-based test-item pools and a test construction and course management system titled IPSIM (Iowa's Item Pool System for Instructional Management), originally devised for a freshman medical course in the Pathology Department of the College of Medicine. The IPSIM system provides the user with the ability to: (1) create and manage item pools; (2) produce "paper-and-pencil" tests, including computer-generated forms; (3) provide Coursewriter III code for interactive test administration; (4) construct multiple-choice questions from a prototype statement; (5) score tests; (6) analyze student performance data; and (7) compose student progress reports. The system appears to have wide ranging potential for other subject matter areas. (Author/KKC)

IPSIM--ADDITIONAL SYSTEM ENHANCEMENTS UTILIZED IN A CHEMISTRY APPLICATION

Donald H. McClain* and Stephen W. Wessels
University Computer Center
The University of Iowa
Iowa City, Iowa 52242
(319) 353-3170

Kenneth M. Sando
Department of Chemistry
The University of Iowa
Iowa City, Iowa 52242
(319) 353-3788

*will deliver talk and handle correspondence

INTRODUCTION

Throughout the past twenty-four months The University of Iowa has been involved with development, implementation, and operation of computer based test item pools and a test-construction and course-management system titled IPSIM (Iowa's Item Pool System for Instructional Management). The evolution of this system has been influential in convincing us that test item pools are a valuable resource for educators and that course-and pool-management programs are an effective utilization of the computer in the instructional process, particularly beneficial for self-paced, individualized courses.

This type of computer application to the instructional process appears to possess wide ranging potential and to be viewed by most users (classroom instructors) as a needed, cost-effective investment of time, effort, and money. The computer aids the instructor in several of the most time-consuming and least-valued parts of his/her job--the production and grading of tests and the analyses of their results, the evaluation of student performance, and the keeping of class records. Additionally, information concerning class performance is provided to assist instructors in identifying and correcting weak parts of the curriculum and in adjusting instruction to the performance and needs of the students.

The IPSIM system, originally devised for a freshman medical course in the Pathology Department of the College of Medicine, is implemented on The University of Iowa's IBM 360 Model 65 computer. The Pathology item pool evolved from the efforts of members of a medical educators' group called GRIPE (Group for Research and Instruction in Pathology Education) and is comprised of over 4,000 questions and student-performance data, with 30 medical schools in the

United States and Canada presently utilizing its capabilities. The basic system has been extended to other academic areas at The University of Iowa through the development of item pools in Materials Science for the College of Engineering, in Introductory Chemistry for the Chemistry Department, and through work on an item pool for the Division of Educational Media.

A detailed description of the original system developed to manage the medical item pool and the self-paced Pathology course is contained in a paper presented at the Fifth Conference on Computers in the Undergraduate Curricula held in Pullman, Washington, June 24-26, 1974. The following section will briefly summarize the features of this system with the remainder of the paper detailing the significant enhancements made to the IPSIM system for the application in Chemistry.

REVIEW OF IPSIM

The IPSIM system which consists of approximately twenty computer programs written in PL/1 (Programming Language One) is totally operational, and we are fully utilizing its capabilities. The item pools, the core of the system, are stored as keyed indexed sequential files with variable length records utilizing IBM's PL/1 data base management routines and disk storage conventions. An item pool is usually comprised of test questions as well as other descriptive information (e.g., key phrases or words, cross references, statistical characteristics, etc.) which are filed by the system.

In computerizing a pool of questions, a classification scheme for organizing and indexing items must be developed. Even though the system will also permit alphabetic letters to be used if desired, an eight-digit number is most often employed for cataloguing and retrieving items in our work at The University of Iowa. The first two

digits of an item pool number are utilized to designate the MCA (major-content area) and the following two digits denote subcategories, the TD's (topic descriptors), under each MCA. The remaining four digits represent the sequence number of the item (three digits) and the version number (final digit).

Employing the generic key capability of file management, items can be selected by specifying either the entire number or any generic part of the MCA-TD number. Illustrating the generic concept through use of alphabetic letters for simplicity, all names in a telephone book file beginning with the letter A would be retrieved by calling for A, all names beginning with Al by AL, names with the first three letters of Alb by ALB, all names having the first four letters Albe by ALBE, etc. Or an entire specific name such as Albert would be retrieved by specifying ALBERT. Since most item retrieval is either for a relatively small number of specific entries or for a large number of entries from a generic subset, the generic key capability has been extremely useful.

In addition to the item number, the structure of a typical item pool entry may include the following components: (1) the question and/or textual material; (2) the correct answer; (3) the question format, a number used by the system for questions having a common set of attributes; (4) the MCA and TD phrases; (5) the SCA (secondary-content area) which indicates a cross reference; (6) the KP (key phrase), a description or reference of the knowledge required by a student to correctly answer the question; and (7) the performance data, with one line of data per each occasion the item is used. Comprising the performance data is information such as the percentage of students selecting each response (foil) or failing to indicate a choice of answers; the number of students tested; the MPL (minimum-

pass level); the discrimination index; and the difficulty of a question.

To create and manage the item pools, computer programs have been written to bank (add questions or other materials/information to a pool), delete, edit (correct errors), update (add performance data), and rename (change classification number) the test questions. An item can be banked which contains any number or all of the components listed in the preceding paragraph, a feature enabling the IPSIM system to be flexible and thus permitting it to be utilized in creating various pools (e.g., test questions, course objectives, text references, etc.). The sequence and version number of an item can either be supplied by the user upon banking the question or be automatically added sequentially to the MCA-TD number by the system.

The system provides three basic forms of output--an item listing, an index, and a test. A listing is simply a printed copy of an item containing one or any number of the seven components previously discussed. An index, a catalog with each entry a brief description of the item, provides the user with a quick method of identifying appropriate questions for the particular application. A test, being defined as a series of questions regardless of its use or purpose, can be generated in two ways: a printed copy for reproduction as a "paper-and-pencil" test or a series of questions coded in CWIII (Coursewriter III), IBM's author language for interactive administration by the computer via a terminal.

Regardless of how a test is administered, a very important and often omitted follow-up procedure is the collection of performance data. The IPSIM system is programmed to utilize input from either mark scan cards or CWIII student records to perform an item and tes-

analysis. The student-performance data obtained is added to the pool, the update making it more informative.

To accommodate self-paced, individualized courses computer programs have been devised to record student data and provide status reports detailing student progress on individual units. These reports can be run at specified periods of time and are valuable assets to course management.

In summary, the IPSIM system is designed to: (1) create and manage item pools; (2) generate tests; (3) score tests; (4) analyze student-performance data; and (5) provide student-progress reports.

CROSS REFERENCING

It has been our experience at The University of Iowa that selection of a classification scheme is by far the single most-important aspect in building any item pool. The classification scheme has probably failed, if the user can ever make a simple request for information from a pool and receive a major percentage of the pool's entries in response. The scheme needs to act as a very definite filtering process.

A dominant philosophy in our work has been that if a specific item does indeed fit into more than one category of the classification scheme, either the scheme has failed or the item should be handled as two absolutely distinct items. Unfortunately, the multiple-entry principle has proven to be not entirely satisfactory for all applications. One solution is to allow liberal cross referencing within a pool. However, there is potential danger here in that the desired filtering process can easily be negated by indiscriminate cross referencing.

Our solution at The University of Iowa, which we term the "parent/alias concept" and advise our users to use with caution,

involves one-way referencing. The parent is comprised of the question or text, any of the descriptors, and any performance data. Furthermore, it includes a count of the aliases which reference the parent (used to assure that the parent is not deleted or renamed before all aliases are removed from the pool). Each alias contains only a reference to the parent, the parent's item number. There is no doubt but that this method can be abused, but we believe it provides just enough flexibility so as to handle the exceptions in item classification without negating the filtering process imposed by the generic classification scheme. In our opinion, cross referencing is definitely not an alternative to a classification scheme.

COMPUTER GENERATED MULTIPLE CHOICE QUESTIONS

To be of greatest benefit and value for a self-paced course an item pool should be large enough so there is no need for test security. In fact, the pool of questions might be an important learning aid for the student if it were to be made available to him/her. But with static multiple-choice questions, probably more than 2,000 items for a one-semester course would be needed. It is believed, however, that a computer-generated question could be equivalent to a large number of static questions in the item pool, depending upon the creativity of the writer.

The IPSIM system was expanded to not only be used as an aid in preparing tests from a pool of items but also to be employed as an aid in producing the test item itself. Even though it is most effective for developing questions with numerical answers, computer-question generation can also be utilized for devising non-numerical questions.

The system retrieves a skeleton or prototype of an item from the pool in a manner identical to that for regular test-question selection. The prototype states the rules for creating the specific

variable components in both a question and its multiple-choice foils. The rules specify exactly how to generate entries for the variable parts of items: the type of entry (alphabetic, decimal, or integer), the range of numeric values, the number of significant digits, and the algorithms for selecting or calculating variables. Only this prototype is banked in the item pool, with the system constructing the actual question during test production.

The effectiveness of computer-generated questions can best be illustrated through use of a simple example. The instructor writes the item prototype as follows:

What volume of 'R 3 .1 .5' M. 'T HCL; HBr; HI; HClO4;'
will neutralize 'N 10 40' mls. of 'R 3 .1 .5" M.
'T NaOH; KOH; NH3;'?
(A) 'RA 3 X3*X4/X1' mls. or 'TA X2'
(B) 'RA 3 2*X6' mls. of 'TA X2'
(C) 'RA 3 X6/2' mls. of 'TA X2'
(D) 'RA 3 1/X6' mls. of 'TA X2'
(E) 'RA 3 X3*X1/X4' mls. of 'TA X2'

Enclosed within each set of single quotation marks is a rule for creating the variable and is replaced by the computer-generated entry during test production. Immediately following the first quote mark is one of the three characters which informs the computer of the type of text to be created: an R for real decimal numbers, an N for integer numbers, and a T for textual (alphabetic) materials. Next is either a blank space for randomly generated variables or the letter A for text-generated by an algorithm. Thus, the six forms of computer-generated variables are R, N, T, RA, NA, and TA.

In the above example the first variable item, denoted by the system as X1, is a randomly generated decimal number of three

significant digits having a numeric value between .1 and .5 in magnitude. The second item to be generated, X2, signifies randomly chosen textual material from a given list. X3 is a random integer between 10 and 40. The computer-generated variable for X4 and X5 are similar in form to X1 and X2, respectively.

In the first foil, the variable X6 is a decimal number with three significant digits calculated by the given algorithm X3*X4/X1. It is important to note that the algorithm (formula) can utilize any of the arithmetic operations as well as some basic functions (e.g., SQR, LOG, SIN, and COS). Furthermore, the algorithm can reference any variable in the item preceding or following it. The second text of the A foil is a simple reference to the textual material selected in X2.

When the generated multiple-choice question is printed as part of an examination, the order of the foils is scrambled. A sample execution of the aforementioned item with the foils unscrambled is:

What volume of .217 M. HBr will neutralize 26 mls. of .346 M. NaOH?

(A) 41.5 mls. of HBr

(B) 82.9 mls. of HBr

(C) 20.7 mls. of HBr

(D) .0241 mls. of HBr

(E) 16.3 mls. of HBr

The computer-generated question system was designed so that: (1) questions can be constructed by instructors who have a minimal knowledge of computer programming, and (2) as the prototypes are written in the question format accepted by IPSIM, computer-generated items can be mixed with and selected in the same manner as the static entries.

AUTOMATIC-TEST GENERATION

For its self-paced courses, The University of Iowa's undergraduate chemistry division had a need for generating a large number of unique tests over specific subject-matter areas. To satisfy this need and as a response to the increasing interest in automatic-test generation, IPSIM was augmented to perform automatic-test generation in a manner which has proven to be quite successful.

According to directions given by the user, test generation typically proceeds as follows: Construct three forms of a test which will be of average difficulty (difficulty index of the test about 60) but will result in a wide range of scores (variance of the difficulty index about 600).

To satisfy the request, IPSIM performs a two-phase selection process:

(1) In Phase I, all applicable items are grouped. Then for each form of a test, a random subset of a sufficient size to perform the next phase is selected.

(2) In Phase II, the items from each subset which best fit the user's test specification are chosen. Once the items are determined, the user can employ any of the test printing, automatic scoring, and analysis capabilities available with IPSIM.

This general description does not serve to demonstrate any specifics of IPSIM's automatic-test generation. In particular, what are "applicable items" in Phase I? In Phase II, what constitutes a "best fit"?

A best fit is achieved by weighting each item in a given subset by the weighting formulas. Items with the highest weight will constitute the set of items which most closely approximate the test's

desired characteristics. Elements involved in the weighting scheme are summarized as follows:

	Subset	Item	Desired
Difficulty Index	calculated avg.	actual/default (can override)	default (can override)
Variance of Above	calculated	NA	Default (can override)
Discrimination Index	calculated avg.	actual/default (can override)	NA
Past Usage	NA	actual	NA
No. of Items in a "category"	calculated	NA	NA
How badly user wants items in a "category"	NA	default (can override)	NA

In addition to the six elements in the above table, four other relevant parameters which the user can override are incorporated to allow him to experiment with and/or massage the weighting formulas.

Applicable items are ones which meet the user's specifications for inclusion. For example, DEFDISC(40),TEST(CATEG(#22-33)) is a complete test specification. The applicable items are all those found under the MCA-TD of #22-33. IPSIM utilizes its default values for everything except the descrimination index value indicated by the DEFDISC which is to be assigned to any items lacking such data.

Two other parameters which can be overriden are the number of forms (default = 1) and the number of questions in the test (default = 25). Thus, the specification to get four, thirty-five item forms of the above test is: DEFDISC(40), TEST(NOFORM(4), NOQUEST(35), CATEG(#22-33)). Modifying the above example to include items from category #88-88, it becomes DEFDISC(40), TEST(NOFORM(4), NOQUEST(35),

CATEG(UNION(#22-33,#88-88))), where UNION is the normal set union operation.

One of the primary purposes of the CATEG (category) descriptor is to equalize any disparity in the number of items from a specific MCA-TD. If #22-33 has significantly fewer items than #88-88, the probability of final inclusion of items from #22-33 is a good deal less than that of items from #88-88. In some applications this is desirable but in others it is not. The following example demonstrates how to equalize these probabilities: DEFDISC(40), TEST(NOFORM(4),NOQUEST(35),CATEG(#22-33),CATEG(#88-88)).

Thus far we have introduced the basic form of IPSIM's test-specification language and have overridden several of the test-description parameters. However, the most extensive aspect of the language is the capability of specifying criterion for the inclusion of items. In the preceding example, the criterion was merely all items of #22-33 and all of #88-88.

Adding--include items from #22-33 which are not aliases for entries in #88-88--the test specification becomes: DEFDISC(40),TEST(NOFORM(4),NOQUEST(35),CATEG(UNION(QUAL(NOT(ALIAS (#88-88)),#22-33),#88-88)). This introduces the QUAL descriptor which stands for qualification. (Items from #22-33 are qualified to not be aliases of items in #88-88.) Also, the example introduces one of the three logical operators; AND,OR are the other two. ALIAS is a so-called bottom-level descriptor because it is applied directly to the items. If, instead of specifying items which are not aliases for entries in #88-88, we specify items which are either computer-generated questions (denoted by a FORMAT of 9) or questions which are aliases, the specification changes to

DEFDISC(40),TEST(NOFORM(4),NOQUEST(35),CATEG(UNION(QUAL(OR(FORMAT(9),ALIAS),#22-33),#88-88))).

It is not difficult to see how complex a test specification can become. On the other hand, suppose the user says he wants items from #22-33 and #88-88. These items should have a difficulty of at least fifty and be computer-generated questions or aliases. First of all, the three qualifications can be logically combined in several different ways. Secondly, which parts of the qualifications are intended to apply to #22-33? Which to #88-88? Which to both #22-33 and #88-88? This illustrates the age-old dilemma of the user translating what he thinks he needs to tell the computer into what the computer needs to be told.

At this point in automatic-test generation we feel it is important to have a language which can never be ambiguous to the computer. Our test-specification language is powerful enough to express any possible inclusion criterion in an unambiguous manner and yet be as reasonable to work with as some computer programming languages. To illustrate both its heavy recursiveness and its flexibility in adapting to the user's way of thinking, suppose two users ask for the following tests:

All items from #22-33 and #88-88 with difficulty of at least 50. But those from #22-33 must be either computer-generated questions or aliases.

Items from #22-33 which have difficulty of at least 50 and be either computer-generated questions or be aliases. Also, items from #88-88 which have difficulty of at least 50.

The two test specifications are:

CATEG(QUAL(GE(DIF,50),

CATEG(UNION(QUAL(AND(GE(DIF,50),

UNION(#88-88,QUAL(CR(FORMAT(9),OR(FORMAT(9),ALIAS)),#22-33),ALIAS),#22-33)))) QUAL(GE(DIF,50),#88-88))).

The first specification shows how recursion can be employed, with the entire category qualified to have a difficulty of at least fifty. The items from #22-33 are then further qualified. In the second specification, each MCA-TD is independently qualified. Though the first specification is shorter, it is exactly the same specification as the second one. In fact, the judgment as to which is the more accurate depends upon the user's reasoning.

In conclusion, automatic-test generation is achieved at The University of Iowa by a two-phase selection scheme. In the first phase IPSIM groups all applicable items, ones which meet the inclusion criterion specified by the user, to be considered for selection in the second phase. The criteria are expressed via a language which: (1) is powerful enough to express any inclusion criterion; (2) is as reasonable to work with as some computer languages; (3) is extendable in the sense that the addition of bottom-level descriptors have been anticipated and space allocated for them; (4) is flexible to the user's way of thinking. From the subset of randomly chosen items formed in Phase I, Phase II selects the set of items which best fit the characteristics desired for the test. Random selection provides unique subsets for each form of a desired test. Fitting of items to the desired test criterion is achieved by the weighting formulas. In addition to the desired characteristics of a test, all the parameters in these formulas can be overridden by the user.

IPSIM's automatic test-generation system has initially performed better than we had anticipated. Due to the flexibility in massaging the fitting criterion, the existence of varied item descriptors in the item pools, and the completeness of the inclusion criterion aspect of

the language, we believe that this system has even greater potential than has yet been utilized.

THE CHEMISTRY APPLICATION

An application utilizing the enhancements of the IPSIM system has been the development of a self-paced introductory course in chemistry. A modified Keller plan for chemical principles was introduced on a trial basis Fall semester, 1974, in three first-year chemistry courses, with a test group of 200 subjects out of a chemistry student population of 1,000. General use of the plan was scheduled to begin Fall semester, 1975; but because of the success of the initial effort, two courses with a student enrollment numbering approximately 600 employed this plan Spring semester, 1975, with the assistance of the IPSIM system.

In the modified Keller plan, the contents of each of the two spring semester chemistry courses is divided into six or more units. A student is permitted to take an examination over a unit as many times as and wherever he/she wishes. It is believed to be more important for a student to acquire a strong general knowledge of chemical principles than to acquire mastery of a few topics. Therefore, the simple mastery system of the Keller plan is not used as the standard for grading; instead, a student's evaluation is determined by totaling the highest scores earned for each of the unit examinations.

Throughout the course, students are aided by a text, supplementary autotutorial materials, conventional lectures, discussions, and open tutorial sessions. It is anticipated that in future years students will also be assisted through use of audiovisual materials and computer-assisted instruction techniques. The ultimate

goal is to provide the student with a truly individualized program while maintaining a high standard of quality.

It should be clear that implementation of a self-paced program in an introductory course of nearly 1,000 students is impossible without the effective utilization of technological innovations. Using the IPSIM system, we have been able to generate, score, and record results from the more than 200 different examinations required in the program. An essential feature has been the ability to generate multiple examinations of equal difficulty.

SUMMARY

Briefly, the IPSIM system provides the user with the ability to: (1) create and manage item pools; (2) produce "paper-and-pencil" tests, including computer-generated forms; (3) provide CWIII code for interactive test administration; (4) construct multiple-choice questions from a prototype statement; (5) score tests; (6) analyze student-performance data; and (7) compose student-progress reports. The three enhancements to the system (cross referencing, computer-generated multiple-choice questions, and automatic test generation) were added to provide capabilities for the chemistry application, but they appear to have wide ranging potential for other subject-matter areas as well. With the additions to the original system, we have a flexible system which has proven to be a valuable asset to its users in a variety of instructional applications.