

THE FORMATIVE EVALUATION AND REVISION OF AN  
INSTRUCTIONAL MANAGEMENT SYSTEM  
FOR BUSINESS COMPUTER COMPETENCIES

by

Andrea Emmot Eason

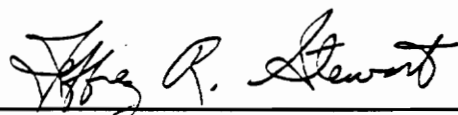
Dissertation submitted to the Faculty of the  
Virginia Polytechnic Institute and State University  
in partial fulfillment of the requirements for the degree of

DOCTOR OF EDUCATION

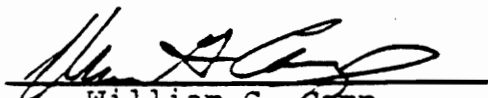
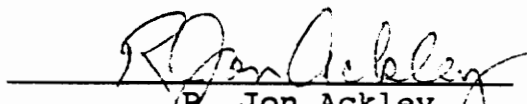
in

Vocational and Technical Education

APPROVED:



Jeffrey R. Stewart, Chair

  
B. June Schmidt  
Daisy L. Stewart  
William G. Camp  
R. Jon Ackley

March 31, 1993

Blacksburg, Virginia

C.2

LD

5655

V856

1993

E276

C.2

## ACKNOWLEDGEMENTS

First, I wish to thank God for every blessing He has bestowed on me. I realize that all things come from the Lord, and I thank Him for His loving sustaining grace throughout the many months of work on this project.

I wish to extend gratitude to my husband, Mack Eason, and my daughters, Frances Hope Eason, Andrea Suzanne Eason, and Lisa Torrence Mills for the love, support, and sacrifices they have had to make during the years of this project. A special blessing to Frances and Suzanne who had to shoulder all household duties while I met the residency requirement at Virginia Tech.

An extra special thank you is extended to George L. Hazelton, a colleague and the co-programmer of LessonBank, who provided friendship, encouragement, and constant help. I learned a lot about programming from George and without his expertise and experience this project would have been much more difficult.

I wish to thank the members of my committee for their support and encouragement during the completion of this study: Dr. R. Jon Ackley, College of Business, Virginia Commonwealth University; Dr. William G. Camp, Dr. B. June Schmidt, Dr. Daisy L. Stewart, and Dr. Jeffrey R. Stewart, College of Education, Virginia Tech.

A special debt of gratitude goes to Dr. Jeffrey R. Stewart who was ultimately responsible for the idea of building the database and application upon which this project was established. His guidance and expertise in business education made this project possible.

My parents, Andrea and Walter (Kayo) Emmot of Independence, Kansas, share a special place in these acknowledgments. They have always encouraged me to do my best all along the way as I pursued an advanced degree.

Thanks are due to my colleagues at Chowan College for their friendship and encouragement.

To Dr. B. Franklin Lowe, Vice President of Academic Affairs, and Dr. Jerry Jackson, President, Chowan College, who provided financial support and encouragement, I owe special thanks.

This acknowledgment would not be complete without a genuine thanks to the Burroughs-Wellcome Foundation who provided me with scholarship funding on several occasions to pursue and complete this degree. Without their support the completion of this degree would be unrealized.

# TABLE OF CONTENTS

Page

LIST OF TABLES . . . . .	ix
LIST OF FIGURES . . . . .	x
CHAPTER 1	
STUDY OVERVIEW . . . . .	1
Statement of the Problem . . . . .	5
Background of Established System . . . . .	7
Assumptions . . . . .	11
Purpose Statement . . . . .	11
Project Development Objectives . . . . .	13
Definitions of Terms . . . . .	14
Limitations . . . . .	16
Need for the Project . . . . .	17
Project Organization . . . . .	17
CHAPTER 2	
REVIEW OF LITERATURE . . . . .	18
Computer Managed Instruction . . . . .	18
Background . . . . .	18
CMI Defined . . . . .	19
Themes of CMI . . . . .	20
CMI Features Enhance Instruction . . . . .	22
Effectiveness and efficiency . . . . .	25
Testing and measurement . . . . .	25
Record keeping . . . . .	26
Designing a CMI System . . . . .	27
Realities of the Classroom Situation . . . . .	29
Information Systems . . . . .	29
Elements of a System . . . . .	30
Hardware . . . . .	30
Programs . . . . .	30
Data . . . . .	31
Personnel . . . . .	32
Procedures . . . . .	34
The Role of Software in Problem Solving . . . . .	34
Software Design Philosophy . . . . .	35
Software Issues . . . . .	35
Data Input . . . . .	35
Output . . . . .	36
Steps in Database Design . . . . .	36
Flexibility . . . . .	37
System Life Cycle . . . . .	38
Documentation . . . . .	39

Formative Evaluation . . . . .	40
Formative Evaluation Defined . . . . .	40
Terminology . . . . .	43
Purpose of Formative Evaluation . . . . .	44
Special Considerations . . . . .	45
Process and Procedures of Formative Evaluation . .	46
Summary . . . . .	49
CHAPTER 3	
ELEMENTS OF <u>LESSONBANK</u> . . . . .	51
Course and Competency Banking . . . . .	51
Task Banking . . . . .	51
Question Banking . . . . .	52
Performance Test Banking . . . . .	52
Test Generation . . . . .	53
Reporting . . . . .	53
Sample Menus, Screens, and Reports . . . . .	54
Example One . . . . .	55
Example Two . . . . .	61
Design Elements . . . . .	71
CHAPTER 4	
FORMATIVE EVALUATION PROCEDURES . . . . .	84
Project Design . . . . .	84
Formative Evaluation Design . . . . .	85
Phase One Evaluation . . . . .	86
Program Revisions . . . . .	87
Phase Two Evaluation . . . . .	87
Phase Three Evaluation . . . . .	88
Subjects . . . . .	89
Data Collection . . . . .	89
Developer's Observations . . . . .	89
Structured Application Assignments . . . . .	90
Data Analysis . . . . .	90
CHAPTER 5	
RESULTS . . . . .	91
Objective 1 . . . . .	91
Objective 2 . . . . .	92
Suggested Changes for Data Input (Screen Forms) .	93
Suggested Changes for User's Manual . . . . .	96
Suggested Changes for Reports . . . . .	100
Programming Errors . . . . .	101
Suggestions for New Features . . . . .	102
Objective 3 . . . . .	102
Objective 4 . . . . .	104
Objective 5 . . . . .	104

Objective 6 . . . . .	105
Suggested Changes for Data Input (Screen Forms) . . . . .	105
Suggested Changes for User's Manual . . . . .	107
Suggested Changes for Output (Reports) . . . . .	107
Processing Errors . . . . .	108
Miscellaneous Errors . . . . .	109
Suggestions for New Features . . . . .	111
Objective 7 . . . . .	111
Task Banking . . . . .	115
Competency Banking . . . . .	118
Course Banking . . . . .	121
Assigning Tasks to Competencies . . . . .	121
Printing Reports . . . . .	125
Creating Lesson Plans . . . . .	126
Multiple Choice Question Banking . . . . .	128
Matching Question Banking . . . . .	132
Performance Test Banking . . . . .	134
Utilities . . . . .	135
Overall Evaluation of <u>LessonBank</u> . . . . .	136
Summary . . . . .	139
CHAPTER 6	
SUMMARY AND RECOMMENDATIONS . . . . .	141
Background . . . . .	141
Project Summary and Discussion . . . . .	143
Recommendations . . . . .	153
Instructional and Research Recommendations . . . . .	154
Potential Uses . . . . .	155
Suggestions for Getting Teachers to Use the System . . . . .	156
REFERENCES . . . . .	158
APPENDICES . . . . .	163
APPENDIX A	
ELEMENTS OF	
LESSONBANK: THE INSTRUCTIONAL MANAGEMENT SYSTEM . . . . .	164
APPENDIX B	
MENU CHOICES FROM LESSONBANK . . . . .	166
APPENDIX C	
LETTERS TO SUPERVISORS AND TEACHERS . . . . .	170
APPENDIX D	
STRUCTURED APPLICATION ASSIGNMENT . . . . .	175
APPENDIX E	
VOLUNTEER SOLICITATION FORM . . . . .	193

APPENDIX F  
SUMMARY EVALUATION FORM . . . . . 195

APPENDIX G  
DATA FROM CHECK SHEET AND EVALUATION FORM  
OF SIX SUBJECTS . . . . . 198

ABSTRACT . . . . . 206

VITA . . . . . 208

## LIST OF TABLES

1	Summary of testing and CMI features . . . . .	23
2	Differences between formative and summative evaluation . . . . .	42
3	A comparison of database and file processing terms	73
4	Program completion time and successful completion rate of structured assignment -- Phase 3 . . . . .	114
5	Overall evaluation of <u>LessonBank</u> . . . . .	138

## LIST OF FIGURES

1.	Select software type, function, and difficulty forms . . . . .	56
2.	Enter a new task form . . . . .	57
3.	Edit task description form . . . . .	58
4.	Selecting a task to change the function form . . .	59
5.	Change the function of a task form . . . . .	59
6.	Change the difficulty of a task form . . . . .	60
7.	Course selection form . . . . .	62
8.	Competency selection form . . . . .	62
9.	Message display screen . . . . .	63
10.	Random selection option screen . . . . .	63
11.	Random selection heading form . . . . .	64
12.	Print menu . . . . .	64
13.	Save to a file screen . . . . .	65
14.	Multiple choice test . . . . .	66
15.	Multiple choice answer key . . . . .	66
16.	Test basis report . . . . .	66
17.	User selects questions heading form . . . . .	67
18.	Multiple choice question selection form . . . . .	67
19.	Popup menu displaying question count . . . . .	68
20.	Select a software type form . . . . .	69
21.	Select a function form . . . . .	70
22.	Select a difficulty form . . . . .	70

## CHAPTER 1

### STUDY OVERVIEW

Microcomputers are now appearing in more of the nation's classrooms. The Office of Technology Assessment (1988) reports that nearly all United States public schools have a computer, and the acquisition of computers by schools is continuing. The availability of computers in the classroom has expanded dramatically in the 1980s. In 1981 only 18 percent of schools had purchased computers. By 1987 computers had been purchased by more than 95 percent of all schools.

According to a survey by the research division of the National Education Association (1991), more than two-thirds (68.1 percent) of the teachers reported they had a personal computer available at their work site for their use. Teachers are beginning to recognize the possibilities a computer offers for both instructing and managing instruction. More than half (61.5 percent) of all teachers responding to the NEA survey concerning use of selected teaching resources (1991) showed they used personal computers regularly. With the increased availability of classroom computers, teachers will have the technology necessary to improve the management of instruction. As the cost of microcomputers has continued to fall, they become feasible in financial terms. As better software becomes

available, microcomputers become more feasible in technical terms.

A teacher can be categorized as both a deliverer of instruction and a manager of instruction. A computer managed instruction (CMI) system will help the teacher with the management of the classroom. As the deliverer of instruction, the teacher may give lectures or demonstrations to aid the learning process. As the manager of instruction, the teacher sets goals, monitors progress toward the goals, plans instruction, organizes and controls the activities of the classroom.

No doubt some teachers have made restricted use of computers for the management of instruction. Keeping records of student progress on a computer using a spreadsheet, for example, is one method of managing information. Using a word processor to store lesson plans or type a test is another. Editing and updating handouts, pupil worksheets, and similar documents is time-saving with a word processor. Output from the reports in a database system can be saved to a text file and retrieved in a word processor for further manipulation. This capability should add to the efficiency and effectiveness of the teacher.

With the emphasis on accountability and competency-based instruction, keeping track of competencies and associating or relating these competencies to the tasks that

must be mastered in each course can be a complicated record keeping problem. Creating tests to measure the competencies taught is a challenge, and knowing which tasks or competencies have been measured can be a time-consuming irritation. Computer technology has the potential to help teachers increase their productivity as managers of instruction as they prepare for their classes. If teaching can be viewed as a production function, efficiency is important.

According to Lancaster (1985), there are three major uses of microcomputers within school systems: (a) teaching computer studies as an academic subject, (b) computer assisted learning in other academic areas, and (c) school administration. Teachers have been involved with computer technology primarily in the first two areas while principals and other administrators have used computers for management activities such as record keeping, e.g., attendance, scheduling, and budgeting.

Faculty use of computers for computer managed instruction has the benefits of increased educational efficiency and effectiveness in the classroom and capability of demonstrating the use of computers in a real setting. If students are taught that microcomputers can be used easily and beneficially for a wide range of administrative applications, they might expect such claims to be reflected

in the administration and teaching of the school (Lancaster, 1985).

Problems in managing instruction efficiently and coordinating large amounts of data so that teaching effectiveness can improve show a need for the development of a system in which all data are integrated. A relational database has the power and sophistication to allow data to be processed as an integrated whole (Kroenke & Nilson, 1986).

Geisert and Futrell (1990) point out that a teacher who uses a computer before and during the instruction process is able to supplement teaching effectiveness and promote student learning. Teachers can use the computer both in the testing process and in the record keeping process.

Two reasons for the growth of microcomputers for management functions in schools include ease of tracking students for purpose of accountability and the need for better information to simplify decision making. The quality of the information is an important factor for decision makers. Decision support information systems can be used for performance evaluation. Computers and the appropriate programs may improve the efficiency with which these tasks are performed.

The challenge is to establish an instructional information system that contains information about courses,

objectives, tasks, and tests, and to relate all the data in multiple views. A database application could be developed to address each of these instructional elements. As a minimum, users of the system should be able to (a) put information into the system, (b) retrieve information from the system, and (c) keep the system up to date (Geisert & Futrell, 1990).

In 1991 a curriculum guide for Virginia business educators was developed which contained data related to 22 business courses and 140 computer software related competencies. In addition, the guide included a taxonomy of 1,100 tasks to be demonstrated to achieve a competency. This information was published in the Business Computer Software Curriculum Series by the Virginia Department of Education. The purpose of the curriculum series was to assist business educators plan instruction on the sizable number of computer operations required for occupations in business.

#### Statement of the Problem

In Virginia there exists a need for a method of organizing, maintaining, manipulating, and processing the many courses, competencies, tasks, and tests that had been organized for teachers by the State Department of Education. If an automated system were developed, tested, and revised, a useful program for educators could be produced.

Virginia business educators could use an automated method of organizing instruction around the sizable number of computer competencies and tasks required for occupations in business. A database containing the data would allow teachers to produce information easily as an aid to managing instruction. No system with this capability currently exists. According to the Office of Technology Assessment (1988), teachers believe that technology eases some aspects of classroom management.

A system that would have the capacity to store the volumes of data related to teaching computer software competencies and allow teachers to maintain and update the data as times change could be a propitious aid for teachers. Teachers think that "computer programs such as spreadsheets, database managers, and desktop publishing can streamline recordkeeping and material preparation" (OTA, 1988, p. 14). For example, a system that would have the capability of generating various reports to assist in the instruction process would be a valuable supplement for a teacher. Simplifying the creation of lesson plans based on competencies and tasks would be a desirable function of the system. The ability to store many test questions to measure each competency would assist with the evaluation process of teaching. A good system would allow teachers to create

parallel versions of tests to measure the competencies taught.

A research study completed by Kohnken (1987) found that educators using database applications on microcomputers perceived benefits including time-savings and improved instruction management. Computer managed instruction was ranked as the most valuable use of computers among ten uses in the classroom (Paulson, 1985). The goal of this project would be to develop a usable CMI program through a series of user tryouts.

#### Background of Established System

A method of identifying and organizing individual software tasks and assigning these tasks to the competencies within courses in the curriculum was underwritten by the Virginia Department of Education in 1989. As a result, a research team was organized to develop a taxonomy of tasks. The taxonomy resulted in the Business Computer Software Curriculum Series published by the Virginia Department of Education in 1990. This publication was based in part on computer competencies extracted from the Business Education Suggested Course Competencies and Performance Objectives, published by the Virginia Department of Education in 1989.

The competencies in the 1989 publication were developed to help business educators with the implementation of updated competency-based courses in administrative systems,

business management, and information systems programs. The competencies were compiled from work by the Business Education Curriculum Study Committee, basic business education course outlines committees, publications of previous competency-based course outlines for business education, local business education supervisors, and members of the Business Education Service, Vocational and Adult Education, Virginia Department of Education.

A subset of competencies that measure computer software skills was extracted from the 1989 competency-based curriculum guide and a listing of tasks identifying specific actions to achieve a competency was developed. The compilation of 1,100 individual tasks was prepared and organized by a research project team. A panel of consultants validated the business computer software tasks and members of the Virginia Department of Education helped in research design and publication. The purpose of the Business Computer Software Curriculum Series was to help business educators plan instruction on the sizable number of computer operations required for occupations in business. Before the tasks were identified, the researchers categorized business application software as follows:

1. DOS and fundamentals, for internal and external disk operating system commands, terms, concepts, and procedures related to hardware and software use

2. Word processing, for text production and printing
3. Spreadsheet, for entering and organizing data into rows and columns for computation and analysis
4. Database, for records management
5. Graphics and graphing, for creation of pictures, designs, stylized text effects, and charts
6. Data communications, for interaction among computers
7. Desktop publishing, for integrated text and graphics and production that approaches the quality of commercial typeset.

In addition, the researchers inspected the documentation from software vendors for each type of software, and examined business textbooks and other materials to develop a preliminary list of computer tasks. The identified tasks were then categorized and grouped according to nine common computer software functions: (a) vocabulary, (b) access software, (c) data/text entry, (d) editing, (e) formatting, (f) printing, (g) file management, (h) production, and (i) troubleshooting. Finally, it was decided that the tasks would be arranged according to difficulty: (a) beginning, (b) intermediate, and (c) advanced.

The initial task list of approximately 1,200 tasks was validated in two stages. First, experts in the use of each type of software reviewed each task on the list to determine

if it should be included in the final taxonomy. Following this review, a business technical committee validated the tasks from the standpoint of business and industry requirements. Extensive revision was made to the original list that resulted in a validated task list of 1,100 tasks each identified by software type, function, and difficulty level.

During the next phase of the project the researchers reviewed the courses and competencies in the 1989 Business Education Suggested Course Competencies and Performance Objectives and determined that 22 courses and 140 competencies required computer operation. The validated computer tasks were then assigned to the appropriate courses and competencies according to the competency statement and performance objective requirements as taken from the 1989 curriculum guide.

Interviews were conducted with a microcomputer database consultant to determine the practicality and feasibility of developing a relational database linking the courses, competencies, and tasks. After determining that a database management system (DBMS) was the proper vehicle to use in managing the data, R:Base 3.1, a relational DBMS with its own programming language, developed by Microrim, Inc, Bellevue, Washington, 1992, was selected. The system design was based on methods extracted from textbooks relating to

design and development of a relational database and database application. Following the decision to use R:Base, a prototype system was designed and programming initiated. The resulting system was titled LessonBank: The Instructional Management System. The comprehensive database application would need to be field tested and revised before final distribution to business teachers for actual classroom use.

#### Assumptions

Certain assumptions were made by the developer in planning this project.

1. Following consultation with a colleague and database programmer, it was assumed that a relational database would be an appropriate method for organizing data related to courses, competencies, tasks, and test questions based on software types, software functions, and difficulty.

2. Following a review of the literature, it was concluded that formative evaluation would be an appropriate procedure for revising and improving a computer managed instructional system.

#### Purpose Statement

The purpose of the project is to evaluate and revise an automated program for managing instruction. Instructors would be able to use the system for planning course content organized around competencies that require instruction in

the use of computer software. The automated system will contain the following data: (a) courses, (b) competencies, (c) a taxonomy of tasks categorized by software type, function, and difficulty, (d) a taxonomy of tasks required to achieve a course competency, (e) objective test questions, and (f) performance tests. To develop the computer-based instructional management system that would ultimately be tested, evaluated, and revised, several supplementary procedures were undertaken. The developer participated with others in an initial project which

1. Determined the business education courses in the Commonwealth of Virginia containing curricular competencies related to software.
2. Identified types of software currently used in the business world as listed in business texts and literature.
3. Categorized tasks by software type.
4. Classified tasks by common functions.
5. Graded tasks by difficulty level.

The developer then

1. Designed in 1990 a relational database to establish a link between course, competencies, tasks, software type, function and difficulty level.
2. Expanded and revised the database, with the assistance of a colleague, to include test questions and answers (developed by other researchers) which would be

linked to competencies, software type, function, and difficulty by task in 1991-92.

3. Developed the database application to access, manipulate, and process the data contained in the database during the period from 1990-92. The resulting application and its associated database were named LessonBank.

#### Project Development Objectives

At the conclusion of the project, the developer accomplished the following objectives:

1. Developed written documentation for LessonBank in the form of a user's manual.

2. Conducted a one-to-one evaluation with an individual to obtain an initial reaction to all components of LessonBank and the user's manual.

3. Revised the user's manual into a more comprehensive document following interaction with the initial user.

4. Revised LessonBank following an analysis of the one-to-one evaluation.

5. Conducted a field test with a small number of individual users to check effectiveness of changes made following the revision process and identified remaining problems.

6. Made further revisions to the program and user's manual following responses from individual evaluations in the field test.

7. Conducted a final field test with Virginia business education teachers to appraise LessonBank individually and make certain all menu choices perform correctly.

#### Definitions of Terms

Bank: a system that allows for data collection that is both structured and formatted.

Column: a relational database term used to describe a group of characters or a single piece of data with a logical meaning. May be used interchangeably with term "field."

Competency: a computer-related occupational assignment. Several tasks may be required to achieve a competency.

Computer managed instruction: computer involvement used in assisting teachers with the making of decisions concerning the instructional process.

Computer managed instruction system: a computer program that has the capability of storing and relating a number of different types of information files (Geisert & Futrell, 1990, p. 155).

Data: raw material from which information is constructed.

Database: a self-describing collection of integrated records (Kroenke & Dolan, 1990, p. 165).

Database application: an organized set of menus, reports, forms and programs, plus the database it manipulates.

Database management system (DBMS): a set of programs that provide the tools to define a database, access the database, and define an application to use the database.

Formative evaluation: a systematic process of planning, developing, revising, and field testing instructional materials.

Integrity: logical consistency (correctness) of the data.

Intersect: a relational command that produces a new table from two source tables when the values in common columns match.

Join: a relational command that produces a new table which contains combined rows from two source tables. The value of a column in the first table is compared to a value in the second table. If the two values have a relationship (such as = or >) as specified in the join operation, then the rows of the source tables are combined to form a third table.

Key: one or more columns that uniquely determines a row (all other columns in a row are dependent on it).

Projection: a relational command that creates a new table which has only the desired columns (a subset of columns) from the source table.

Relational database: a structure composed of named tables, which may be divided horizontally into unnamed rows and vertically into named columns. Based on the relational model of data developed by E. F. Codd.

Relational operations: the ability to manipulate tables in various ways. Four common operations (defined elsewhere) are intersect, join, projection, and selection.

Row: a structure corresponding to a record in a flat file system (a collection of fields or columns about some entity).

Selection: a relational command that creates a new table which has only those rows from the source table whose columns meet conditions prescribed in the selection operation (a subset of rows).

Table: a structure corresponding to a file (with rows and columns).

Task: an activity required to produce something useful at a computer.

### Limitations

The project was limited in the following ways:

1. The primary subjects were business educators who had access to the required hardware. Hardware requirements

include an MS-DOS computer with a 286 (or greater) microprocessor, 640K of main memory, a hard disk with 5Mb of free disk space, and DOS 3.1 or higher.

2. LessonBank requires a runtime version of R:Base 3.1C. This program was purchased with unlimited distribution rights.

### Need for the Project

The exploration of a practical way to manage classroom instruction is needed because of the increasing demand placed on educators to provide competency-based curricular materials and a method for measuring the competencies. The system should be efficient if it is to help business educators. Teachers would welcome support in organizing data at their disposal. A computer managed instruction system with a collection of components should help users collect and organize data and assemble it into information which can be used when it is needed.

### Project Organization

This chapter provided an overview of the project. Chapter 2 discusses the literature. Chapter 3 explains the elements of LessonBank: The Instructional Management System and the basic database design elements. The data collection procedures are described in Chapter 4. Chapter 5 contains the results of the formative evaluation and data analyses. Chapter 6 presents a summary and recommendations.

## CHAPTER 2

### REVIEW OF LITERATURE

The major components of the review of the literature are (a) computer managed instruction, (b) information management, (c) the role of software in problem solving, and (d) formative evaluation.

#### Computer Managed Instruction

The capability of the computer to store, process, and manipulate an abundance of data has joined with the need of educators to effectively manage instruction. This union has resulted in the design and development of computer managed instruction systems. Several such systems are currently evolving. Daft and Becker (1978) point out that within a number of school districts there are professionals who, without knowledge of one another, are developing components of instructional information systems.

#### Background

The origins of the term computer managed instruction (CMI) are vague, but as early as 1965 people began using the acronym. Around 1968 CMI began appearing in the literature (Baker, 1978). Terms such as computer-based instructional management system (CBIMS) have been used but none proved to be as popular as CMI. The term Instructional Information Systems (IIS) was popularized at the UCLA Center for the Study of Evaluation (CSE). The IIS project has been

tracking selected aspects of the computer revolution in education and one of those areas is instructional management (Bank & Williams, 1987).

Although many sophisticated computer managed instruction (CMI) programs have been available for a number of years, educators have not shown a great deal of interest in them. In a study done by Scrogan (1988) through the Office of Technology Assessment, CMI was not even listed as a category.

Early CMI systems utilized the mainframe computer as the computer component. The fact that not many schools had access to a mainframe has undoubtedly had an impact on the implementation of CMI systems. The development of the microcomputer and its easy accessibility may bring a resurgence of interest in CMI.

#### CMI Defined

Baker (1978) defines CMI as a "total educational approach" in which the management functions performed by the teacher are supported by a computer-based management information system. This total educational approach "encompasses the educational goals, the curriculum, the instructional model, the teacher, and a management information system" (p. 14). Geisert & Futrell (1990) define a CMI system as "a computer program that has the capability of storing and relating a number of different

types of information and files" (p. 155). These files represent various entities that need to be managed. They may include, but are not limited to, students, objectives, goals, test questions, learning materials, and administrative information such as attendance. Geisert & Futrell (1990) indicate that computer managed instruction (CMI) refers to computer involvement in the professional decision-making process--decisions about how students are performing, which students need further instruction, and when to move students on to new topics. Baker (1978) concludes that a CMI has common capabilities for testing, diagnosing, prescribing and reporting. Yet another writer postulates that "instructional management takes place wherever and whenever people make decisions that affect the substance or organization of instruction" (Bank & Williams, 1987, p. ix).

Baker (1978) makes the point that CMI is not just computer assisted test construction (CATC) even though it can be a very important component of CMI. Most CMI systems have integral CATC components (Egan, 1973; Greisen, 1973; Lundgren, 1991; Merrill, 1974).

#### Themes of CMI

Three themes weave throughout the concept of computer managed instruction: (1) individualization, (2) behavioral objectives, and (3) technology (Baker, 1978).

Baker (1978) characterizes early CMI systems as being created to show the feasibility of using the computer to support the management of individualized instruction. Gagne and Briggs (1974) believe that instruction must be for the individual. Education needs to be optimized for individual students rather than the masses. Individualization can be supported by a comprehensive CMI system.

Expected or planned outcomes of the events of learning are called behavioral or performance objectives by Gagne (1974). Emphasis is placed on "task analysis" of a curriculum or subject matter area and breaking it down into behavioral objectives (Tyler, Gagne, & Scriven, 1967). The objectives form the basis for instructional segments designed to facilitate the attainment of the objectives by the students. Instructional information systems are needed to form the backbone of prescriptive or adaptive teaching (Bloom, 1976; Glaser, 1977). This instructional approach organizes classrooms to focus on individual student learning needs. Bank et al (1987) have shown that adaptive instruction demands that instructional objectives be defined and that objective-referenced diagnostic tests be created.

The basic assumption underlying adaptive teaching is that, with good and timely data about student performance, a teacher can accurately diagnose achievement deficiencies and, based upon the curriculum, formulate a plan (prescribe) to remediate deficient skills. Information on those skills that students have mastered and on those skills that

students have yet to master allows for better lesson planning and more purposeful grouping of students (Bank et al, 1978, pp 169-170).

The third theme underlying computer managed instruction is technology. Technology in education has encountered many false promises and dashed hopes over the years. Television was once touted as the solution to all educational problems. Teaching machines and computer assisted instruction (CAI) were extolled as the answer to improved educational results. Technological advances are generally embraced with enthusiasm and before long expectations are not reached and the enthusiasm subsides. However, Baker (1978) feels that "in a very real sense, the digital computer and its associated technology represents an increase in educational potential of many orders of magnitude over previous technologies" (p. 10).

The combining of the three themes resulted in the configuration of computer managed instruction: behavioral objectives, which yielded a style of curricular plan; individualization, which yielded an instructional model matched to the curricular plan; technology, which enables CMI developers and implementers to keep track of the many aspects of instruction.

#### CMI Features Enhance Instruction

A CMI system may be as small as one teacher using one microcomputer for word processing or it may a large

comprehensive system utilizing a database with communication capabilities for sharing data over a network of computers.

Table 1 presents a summary of the range of various CMI systems and CMI features (Geisert & Futrell, 1990, p. 158).

Table 1

Summary of testing and CMI features

Spectrum of testing and CMI programs available	Brief descriptions of the program and the computer and hardware required to use this type of program
Word processor	Need any type of computer, printer, and word processing program. The word processor makes test typing easy; tests can be easily changed and easily stored.
Test generator	Need computer, printer, and test generation program. Need to enter test questions and answers but the form for the questions and the test is provided and there may be easy generation of alternate forms. Easy to store, change, and print tests and work sheets.
Test scoring	Need computer with an attached mark-sense reader and special program. Special test answer sheets are fed into the machine. The tests are automatically scored and the results recorded and printed out for the teacher.
Testing included with a text	Need computer and printer. Publisher provides the program and test questions for the text. No questions need to be entered, and often one can pick and choose from an item bank.

Spectrum of testing and CMI programs available	Brief descriptions of the program and the computer and hardware required to use this type of program
CMI included with a drill or tutorial lesson	Need lesson program and computer to run it. Teacher needs to enter student names and set any instructional parameters, such as difficulty level of items.
Simple CMI	No special hardware required, but would need a CMI program. Teacher is required to enter aspects of the curriculum such as goals, objectives, test items, and student names.
Comprehensive CMI	This requires a powerful computer and printer capability, and a special CMI program. Extensive information is required: goals, objectives, test items, learning materials, student information, and other special aspects.

One of the goals of a CMI system should be to support quality classroom instruction. Geisert & Futrell (1990) define characteristics of quality instruction as:

1. Having clear and meaningful purpose
2. Using effective and efficient teaching methods
3. Implementing valid and reliable measures of outcomes
4. Maintaining a record keeping system "to keep track of what is happening and to sustain decision making in the other three areas" (p. 143).

Identifying an instructional goal is the first step in developing a model of instruction (Dick & Carey, 1985).

Creating a systems approach model for designing instruction should ensure the achievement of purposeful, effective instruction. The CMI system can be used to supplement instruction for efficiency and effectiveness, measuring, and record keeping.

Effectiveness and efficiency. Teachers should be the immediate beneficiaries of CMI as an effective system should reduce the paperwork and the time needed to evaluate the performance of the student (Baker, 1978; Geisert & Futrell, 1990). Further, the teacher's decision-making capability is made more effective by all the information available for use. Teachers should gain confidence in their instructional planning because of the multiple measurement opportunities provided and the assurance of accuracy in measuring mastery of objectives (Lundgren, 1991).

Testing and measurement. Some process of assessment must be used by a teacher to determine if and when the students have achieved the goals and objectives. A computer program can be used to generate tests that measure individual objectives to determine if students have mastered a particular learning goal. Mastery-learning courses (Geisert, 1974) can easily be developed utilizing the microcomputer to create parallel versions of tests. The test generating capability of a CMI system allows instructors to create nearly unlimited testing and retesting

opportunities for students. Test construction capabilities can be as simple as using a word processing program for efficiency in typing tests. Word processing programs allow the user to save documents for future use, to edit work easily by moving blocks of text around, and rearranging questions. Test generating programs can enable the instructor to select the types and numbers of questions to have in a test. The test maker may be used to generate a random selection of questions meeting certain objectives. A program of this type should allow a user to enter and edit questions so that a test-item bank could be developed and improved over time (Geisert & Futrell, 1990). In addition, there are programs that will present the test to a student on the computer, score it automatically, and save the results on a diskette (Lundgren, 1991). A disadvantage of the on-line testing program is that too much of the computer time may be utilized by students taking and retaking tests (Geisert & Futrell, 1990).

Record keeping. Adequate record keeping must be achieved in a quality classroom instruction program. Examples of record keeping include grade books in which data are recorded manually, spreadsheet data management records of students in which data are stored electronically, or a computerized database program that is tracking a large

number of students who may be working on different objectives.

### Designing a CMI System

Developers of a CMI must be aware that users of any CMI system must be able to accomplish the following:

1. Get information into the system
2. Keep the system current
3. Get information out of the system (Geisert & Futrell, 1990).

In designing a CMI system the developers and users must work together to organize the following (Geisert & Futrell, 1990):

1. Establish a curriculum listing with all the learning goals to a high level of specificity
2. Write five to ten test items for each objective
3. Associate each learning goal with a specific teacher or course.

Learning objectives should form the core of any CMI system (Futrell & Geisert, 1984; Tyre, 1989). As CMI systems have evolved in response to technological advances, the focus on objectives has remained constant. Objectives are tied to all other elements of a CMI--testing, learning resources, and student information. Defining clear objectives supports the instructional elements of measurements, teaching, and record keeping.

Lundgren (1991) enumerates the following objectives as being indispensable to the reasonable functioning of a computerizing testing system in a classroom:

1. Ease of item entry and editing
2. Random selection of a subset of items
3. Accommodation of a variety of input/output formats
4. Security of the test as a disk file
5. Efficient collection and summary of results (p. 2).

Other guidelines as suggested by Lundgren (1991) for developing a test bank are that it should contain at least three times the number of items to be selected for the actual test in order to have the capability of creating forms that are sufficiently different. On the other hand, it would not contain more than ten times the number of items to be chosen to ensure equivalency.

A good test generator will support a variety of types of questions: multiple choice, matching, true/false, performance. For output the test generator should be able to output the test to three teacher-selected formats: (a) disk file, (b) printer, (c) screen (Lundgren, 1991).

CMI needs sincere commitment on the part of teachers and administrators. Rethinking the traditional "teach and test" patterns is a way to improve the quality of education (Baker, 1978; Geisert & Futrell, 1990).

### Realities of the Classroom Situation

One of the difficulties faced by those interested in CMI is developing support at the local level for a long-term commitment to CMI. As educators work to meet the demand for accountability, the concepts of achievement gains per dollar and the cost per pupil dominate the evaluation for effectiveness (Baker, 1978). Unfortunately, as CMI systems are developed they do require additional expense in computer software and hardware.

It would seem that the issue of improved student performance should be tied directly to the instructional program that has already been developed and should be evaluated separately from CMI. The contribution in the educational process generated by CMI should be to make the instructional program function properly and to support its management. With this as a goal it is not possible to show an increase in student achievement or a reduction in cost directly attributable to a CMI system (Baker, 1978).

### Information Systems

As defined by Kroenke & Dolan (1990) "a system is a collection of components that interact to achieve some goal" (p. 24). In the same manner, a computer system is defined as "a collection of components, including a computer, that interact to achieve some goal" (p. 24). The point the

authors are attempting to make is that a computer is not a system.

### Elements of a System

A computer system is composed of five parts as defined by Kroenke & Dolan (1990). The five component model includes hardware, programs, data, procedures, and personnel. All five of these components are equally important to the success of a computer system.

Hardware. A computer is a configuration of four types of devices all connected by cables. The equipment is categorized as (a) input, (b) output, (c) processing, and (d) storage. The central processing unit (CPU) contains the "brains" of the computer. The system unit contains the CPU. The CPU contains main memory or random access memory (RAM). Main memory holds the programs to be executed and the data to be processed. Main memory is also called temporary storage and is frequently described as being volatile because when the power is turned off whatever was in main memory is gone. Main memory is measured in terms of bytes. Microcomputers generally contain between 256,000 bytes (256K) and 16 million bytes (16 Mb) of RAM. The more RAM a computer contains the larger a program it is capable of running.

Programs. Programs are the written instructions for the computer. Software is another term used to describe

programs. Programs are categorized in two basic ways: (a) system programs, i.e., DOS, OS/2, UNIX, that coordinate the execution of all other programs, and (b) application programs that solve specific problems. Powerful application programs are currently available for direct purchase "off-the-shelf." In addition, custom programs may be written in one of the over 200 programming languages available. Some well-known programming languages are BASIC, COBOL, Pascal and C. Each custom written program must be designed, coded, tested, debugged (errors removed), and retested. Custom-developed software is more expensive, but one of the benefits is that it fits the user's requirements perfectly (Kroenke & Dolan, 1990).

Data. The raw material that goes into the computer is called data. The "finished product" is information. Information is defined as "knowledge derived from facts" (Kroenke & Dolan, 1990, p. 51). Data are also categorized as (a) input, (b) output, (c) processing, and (d) storage. Input data are the raw facts entered into the computer by an input device like the keyboard. Output data are information for human use like a student's transcript or grade report. Processing data are the data loaded into main memory for further manipulation. Storage data are saved on disk for later processing.

The bit is the basic building block representing data. A bit is an abbreviation for binary digit and consists of only two symbols: 0 and 1. Data are represented in the microcomputer in a series of 8 bits in a code called ASCII (American Standard Code for Information Interchange). This code is a pattern of bits forming a byte used to represent each character. Data that has been saved in ASCII format can be read by a program other than the one which created it.

Data are arranged in a hierarchy as follows: bit, byte or character, field, record, and file. A bit is the smallest representation of data, either a 1 or a 0. A byte is a group of bits forming a character. A field is a group of related characters representing some fact. Last name of a student would be one field, the first name another field, the grade level yet another. (A field may also be referred to as a column in database terminology.) All three fields about one person compose a record. A record is a group of related fields. A record may also be referred to as a row in database terminology. A collection of related records is a file. The information about all the students will be contained in a file.

Personnel. The fourth part of a computer system involves four types of personnel. These are (a) system developers, i.e., programmers and/or systems analysts, (b)

operators of the computer system, (c) users of the system, and (d) clientele (Kroenke & Dolan, 1990). In a small interactive system using a microcomputer, all four categories may be contained in the same person. In most systems the categories are separated. The system developers write the programs after interviewing the users and determining the requirements for the new system. The programmers and systems analysts must work closely with the users of the system to develop a workable system.

The operator actually runs the computer. Operators have to know how to start the program, how to input data, how to stop the program and how to operate hardware like the printers. Baker (1978) points out that in at least one large CMI system the use of a teacher as a computer terminal operator was not cost-effective. Because the operation of the computer became a routine task, it was felt the job did not require a professional level person. As CMI systems are developed using microcomputer configurations rather than mainframe computers, teachers or aids are likely to serve in the capacity of computer operator.

The users are categorized as people who use the computer system as a tool to do their specific jobs. Users typically have expertise in some specialty. In education those specialists would be teachers, supervisors, or administrators. In business they would be accountants,

managers, or supervisors. In an interactive microcomputer system the positions of user and operator may merge into one.

Procedures. Procedures are the written instructions for the people who use the system. Procedures are categorized in two areas: normal operating procedures and failure recovery procedures. Normal operating procedures include such things as how the inputs are developed, how to use the system, how to generate output, and how to make backups. Failure recovery includes what to do in case of a system crash, equipment failure, and how to restore data from backups. Procedures must be written down in the form of documentation and followed if they are to be effective (Kroenke & Dolan, 1990). A reliable computer component of a CMI system depends on several factors. One of those factors involves procedures. The reliability of a CMI system in part "depends upon having simple, systematic procedures for both normal and abnormal operations" (Baker, 1978, p. 328).

#### The Role of Software in Problem Solving

The computer component of a CMI system is represented to a large extent by the computer application programs designed specifically for the project. The programs determine what the system will be able to do, what its features are, what reports are generated, and its flexibility for future growth and expansion.

## Software Design Philosophy

The design of a program should be based on a modular approach. Each module should be as self-contained as possible. If a modular approach to software design is used, then future program modifications are easier to incorporate into the overall plan. If a major module has several functions to perform, this module can be divided into several smaller submodules. Some functions which are used in several modules, such as printing, could be designed as utility routines that are used throughout the program (Baker, 1978).

## Software Issues

Whether one has a quality system or a weak system depends on the conceptualization of the system by the developers. Much thought must be put into the design of the database which forms the core of the system. Several issues should be considered as the system is being developed. A good CMI system will be one in which reliability of the system has been considered. The CMI program must be protected against those who use it.

Data input. The keyboard is the most popular form of data input. Data can be entered into the program using the input routine developed with the program, or, in some cases, data may be entered using a word processor which generally has easy to use editing features which may already be

familiar to the user. Some CMI systems allow data input by using optical mark readers and desk top scanners (Baker, 1978). The data input routine should provide adequate prompts on the screens so that the operator can respond easily. Baker (1978) points out that when errors are made the system should make editing as easy as possible. Sufficient checks and prompts should be used to insure that the proper sequence of operations is performed.

Output. The computer component should have the capability of generating various reports. Much of the power of any database system or management information system is the ability to generate a variety of reports from the data. To be useful, data should be available to the user in at least three formats: (a) to a disk file in ASCII or DOS text format, (b) to the display screen to be previewed, and (c) to the printer (Lundgren, 1991). The reports generated by various CMI systems depend on the subject matter, the curricular plan, and the instructional model (Baker, 1978).

#### Steps in Database Design

Undoubtedly the use of sophisticated database management software (DBMS) packages will increase the flexibility of a CMI system (Baker, 1978). Using a DBMS package to build a database requires developing a design plan or database model. Kroenke and Nilsen (1986) suggest a simple but effective way to start developing the model:

1. List the objects in the work environment you are modeling (p. 49). Objects are things to keep track of like courses, students, objectives, and tests.

2. Describe the relationships among the objects. The data model must show the relationship the objects have to one another. Relationships can be one-to-one, one-to-many, or many-to-many (p. 49).

3. Decide what facts about the objects are important (p. 49). The facts will contain the data to produce reports and will become the names of the columns.

4. Designate the key columns (p.49). Each row in a data table must be unique. A key is the column that uniquely identifies the row. If one knows the key, one can determine the data found in the rest of the row.

5. Record the relationships among the objects (p. 50). The relationships are represented by the facts that the objects have in common.

Flexibility. One characteristic of a good CMI software package is flexibility. As users become familiar with the program, they want to be able to do other things. They begin to understand what services the computer can offer, and they desire to expand these services. Flexibility should be built into the software from the beginning. Baker (1978) gives three factors for designing flexibility into the system:

1. The designer must look beyond immediate needs, concepts, and requirements when creating CMI software. By looking ahead as far as possible, and borrowing liberally from other systems, the software designer can provide the flexibility needed to meet future needs (pp. 344-345).

2. Software modularity is also fundamental to flexibility and high quality software. A proper software fractionalization results in modules which are compact and conceptually independent. With the software structured around modules, one can add new modules and delete old ones without affecting the structure of the computer program (p. 345).

3. The final key to software flexibility is good data base design. One is in a much better position to meet new requirements when the data base design has some generality (p. 346).

### System Life Cycle

What often happens to a CMI system is that little thought is given to the life span of the system. In the beginning all concern is focused on the design of the system and the implementation. Some consideration should be given to what will happen after the developmental team is disbanded (Baker, 1978). All systems have a life cycle. The life of the system begins when someone recognizes a need. With the need in mind, and the knowledge of the

resources available to meet the need, a system can be developed and put to use. The system is then used for a while and its effectiveness is evaluated and changes made.

Changes may be made for three reasons:

1. The user's needs have changed, or the user may recognize new needs.

2. The system did not meet the user's needs.

3. Technology has changed, providing new ways of dealing with the user's needs.

When changes to the system are needed, the cycle begins again. The development of an information system is an ongoing, iterative process (Kroenke & Dolan, 1990). Baker (1978) suggests that once the baseline system is operational, the improvement phase begins. The users request additional changes and the developers attempt to find areas of the program where improvements can be made. However, if the developer responds to each request for a change, there may become an endless series of small changes made which have not been evaluated properly and may have a negative impact upon the quality of the program. All changes should be carefully evaluated with the goal of making the program reasonably stable.

### Documentation

One of the products to be delivered at the end of the design stage of system development should be documentation.

Documentation should be written for the developer, the user, and the operator. The documentation written by the developer should include the schema of the data, how the data are related, copies of all the programming code contained in each module, and a data flow diagram. Documentation for the user and the operation staff should contain written instructions about how the program works (Kroenke, 1992).

### Formative Evaluation

The following section provides a review of the literature relating to formative evaluation. Included in this section is a definition of formative evaluation, a discussion of various terms, a consideration of the purpose of formative evaluation, and some special considerations of formative evaluation and computer utilization. A discussion of the process and procedures is included.

#### Formative Evaluation Defined

Formative-summative evaluation was first defined in 1967 by Michael Scriven (cited in Popham, 1988). Scriven's distinction of the two terms was that formative evaluation focuses on improvement of a product while summative evaluation compares two alternate products. Initially Scriven defined evaluation as "the systematic and objective determination of the worth or merit of an object" and labeled this as summative evaluation. He felt that

summative evaluation was more important than formative evaluation. According to Stufflebeam (1983), in a comparison of his CIPP framework with Scriven's formative-summative approach, Scriven had placed an emphasis on the comparative approach of summative evaluations because his audience included potential users of packages, and they were interested in the recommendations about which alternatives they should purchase. In direct contrast to Scriven's early insistence on the importance of summative evaluation, Stufflebeam (1983) says "the most important purpose of program evaluation is not to prove but to improve" (p. 117).

Worthen & Sanders (1987, p. 36,) summarized and distinguished between formative and summative evaluation in several ways as shown in Table 2 on the following page.

Table 2

Differences between formative and summative evaluation

Domain	Formative Evaluation	Summative Evaluation
Purpose	To improve program	To certify program utility
Audience	Program administrators and staff	Potential consumer or funding agency
Who should do it	Internal evaluator	External evaluator
Major characteristic	Timely	Convincing
Measures	Often informal	Valid/reliable
Frequency of data collection	Frequent	Limited
Sample size	Often small	Usually large
Questions asked	What is working? What needs to be improved? How can it be improved?	What results occur? With whom? Under what conditions? With what training? At what cost?
Design Constraints	What information is needed? When?	What claims do you wish to make?

Formative evaluation has been defined by Dick (1977) as a "process of systematically trying out instructional materials with learners in order to gather information and data which will be used to revise the materials." The related literature on formative evaluation describes this evaluation procedure for the purpose of improving an

instructional program to determine any needed revisions (Baker, 1974; Patterson & Bloch, 1987; Popham, 1988; Tuckman, 1985). Formative evaluation is sometimes referred to as "developmental testing" (Golas, 1983) because alterations that are made to the program are made from tryout data as development is being conducted.

Since a CMI system is designed specifically for the teacher(s) rather than the pupils (Baker, 1978), the term "learners" is replaced by the term "teachers." The term "instructional materials" will, of necessity, need to be replaced by "instructional management materials." The teachers and learners ultimately become the "users" or beneficiaries of the improved materials whether they are developed for a classroom learning situation or a classroom management situation.

Computer managed instruction (CMI) systems have received a limited amount of consideration in the literature with respect to formative evaluation. It is critical that developers of CMI and Computer-Assisted Instruction (CAI) involve themselves in formative evaluation of computer materials (Patterson & Bloch, 1987).

### Terminology

For a fundamental understanding of the term "formative" when used to describe an evaluation process, researchers (Dick, 1977; Dick & Carey, 1985; Gagne & Briggs, 1973) make

the distinction that the process is occurring while the product is being formed or developed rather than after it is developed. In addition, the term "evaluation" does not imply making a judgment about what a student or user has gained by using the materials. Summative evaluation would entail a comparison of alternate methods or systems and an evaluation of potential gains would be indicated during this process (Dick, 1977).

Formative evaluation is based upon some "object." In the past, students and teachers have been the objects of evaluation in education. Now programs or projects or curricular materials can be the objects of formative evaluation. In fact, almost everything can be an object of evaluation. It is important to identify the object clearly because this helps keep an evaluation focused (Nevo, 1986).

#### Purpose of Formative Evaluation

The focus of formative evaluation is on the techniques that are used to improve materials (Worthen & Sanders, 1987). Formative evaluation requires a "feedback phenomenon" (Baker, 1974; Scriven, 1973; Stufflebeam, 1974) as data are collected and then a judgment is made as to what improvements will be used to modify the product. The purpose of formative evaluation is not to rate the effectiveness of the materials themselves. The main purpose is to enhance the product under development. According to

Stufflebeam (1974) formative evaluation addresses questions about vocabulary level, usability, appropriateness of media, durability of materials, efficiency and other matters.

A testing/revision cycle is a necessary component of courseware development according to Robelyer (1983). This evaluation/improvement cycle will help assure quality courseware. In order to achieve this purpose, systematic, empirical evidence must be gathered.

### Special Considerations

Formative evaluation and computer utilization have some special considerations which are inherent as enumerated by several authors: (Golas, 1983; Patterson & Bloch, 1987)

1. Possible crashes of software causing frustration.
2. Possible breakdowns of hardware triggering postponement of the evaluation process.
3. Time-consuming revision process involving further programming.
4. A target audience who may be unfamiliar with a computer or at least may be apprehensive toward using it.

Evaluators should be aware that differences exist between formative evaluation of print materials and formative evaluation of interactive computer-based products. Attention must be paid to screen input formatting, report formatting, the type of hardware required, etc.

### Process and Procedures of Formative Evaluation

Formative evaluation by its very definition implies change. Baker (1974) suggests several guidelines for conducting formative evaluation:

1. Collect information on areas where something can be done.
2. Select "very few subjects for early versions of new programs and expand this number only as there is evidence that the program is working" (p. 544).

Formative evaluation will require a number of rounds or phases before completion. Three phases are recommended by Dick 1977; Dick & Carey, 1985:

1. One to one evaluation with one to three subjects.
2. Small-group evaluations with 5 to 15 subjects.
3. Field tests of 20 or more subjects.

Since reprogramming is required with CAI (and CMI) and the procedures for conducting the trials can be fairly complicated, Golas (1983) recommends that the formative evaluation consist of the following three stages: one-to-one evaluation with one subject and handwritten materials; one-to-one evaluation with one subject at the computer; and small-group evaluation with three subjects at the computer.

The most acceptable method of initial evaluation involves data gathering in a one-to-one situation and using this data as base line information for the product being

evaluated (Baker, 1974). This base line information can be used to effect change for the next round of evaluation. Since formative evaluation is conducted in a decision mode, considering what to change and what not to change is of prime importance. The base line data will aid the developer in making these decisions.

Arenson's model of formative evaluation (cited in Patterson & Bloch, 1987) suggests that the first step in formative evaluation should be a technical review in which a technical expert reviews the material for timeliness, appropriateness, quality of presentation, and content data to be followed by major revisions to the original prototype.

Golas (1983) feels that a full-scale field test is probably not necessary when considering the cost of conducting such a test may outweigh the value of the information to be collected. However, a field test may detect management problems. These problems may either require further changes to the program or the development of some additional materials such as a user's guide. Baker (1974) indicates that the field test is necessary to determine the functionality of the product under relatively normal conditions. Establishing satisfaction in using the program with the teachers and supervisors is important. One method of accomplishing satisfaction is through interview or questionnaire procedures. In order to obtain the most

complete data, it is suggested that face-to-face contacts are the most desirable even though they are relatively expensive. Data analysis should detail exactly what happened during the tryout period (Baker, 1974). Field testing should lead to an operationally ready program.

According to Baker (1974), some developers recommend one subject be involved initially as a "learner-informant" who can talk through difficulties with the program. Others suggest using just a few subjects at first and then increasing the size of the groups as the evaluation proceeds. Assuming the program is successful and no major difficulties arise, several subjects should become involved. However, Baker (1974) does not feel that it is necessary to involve 30 subjects with a field trial in formative evaluation. Too much data will have a tendency to cause unnecessary delays in the development activity.

Scriven (1973) asserts that it is important to assign the role of formative evaluation to a person who is a regular part of the program being evaluated. This would be someone who is familiar with the project and understands the details.

Researchers must consider the answer to the question "when is formative evaluation over?" Baker (1974) indicates that there is no rule limiting the number of field tests necessary. The completion depends on several factors such

as how well the program is working, the resources of the developer, or how much improvement previous revisions have been able to generate.

### Summary

As microcomputer technology becomes more powerful, less costly, and universally available in the classroom, its use for managing instruction has great potential. CMI should be of greater interest to educators as a method that can be used to improve instruction and contribute to efficiency and effectiveness in areas such as testing and measurement, record keeping, and instruction.

A system contains several components designed to achieve a goal. A computer information system is composed of five parts: hardware, programs, data, procedures, and personnel. All five of these components are equally important to the success of a CMI system. The planners and developers of a CMI system should plan for the life cycle of the system.

The formative evaluation process is a necessary step in the development of computer-based materials. Formative evaluation is generally a time-consuming process, but the evaluation and subsequent strengthening of the materials will ensure the quality of computer programs that educators can trust to achieve the desired results. Even though a many studies have been conducted utilizing formative

evaluation, none were found by the author relating specifically to computer managed instruction (CMI).

## CHAPTER 3

### ELEMENTS OF LESSONBANK

LessonBank: The Instructional Management System (Virginia Department of Education, 1993) is identified by the developers as a computer managed instruction (CMI) system. A system generally incorporates a number of components. The components of LessonBank are: (a) course and competency banking, (b) task banking, (c) question banking, (d) performance test banking, (e) test generation, and (f) reporting. The elements of the system are summarized in Appendix A.

A bank is a system that allows for data collection that is both structured and formatted. The data in all banks have the capability of being added to, revised, deleted, selected, sorted, and printed. LessonBank provides the capability of banking with four separate components:

#### Course and Competency Banking

The first bank contains a collection of structured competencies related to specific courses. Twenty-two courses and 140 competencies were established initially in LessonBank.

#### Task Banking

Another bank within the system includes a collection of a structured set of tasks. The tasks are each categorized according to variables software type, function, and

difficulty. The task bank currently contains 1,100 tasks categorized by these variables. The system allows the assignment of tasks to course competencies.

#### Question Banking

The question banking component of the system contains a collection of a structured set of multiple choice and matching questions. The questions are stored by the key element of the task bank, i.e., task number. The test questions may be used for (a) creating formal tests by the variables software type, function, difficulty, or competency, (b) reviewing topics by each variable and (c) diagnosing weaknesses by student. Questions in the bank are not numbered. A routine for numbering the questions was written for the test generation program.

Currently the multiple choice question bank contains 359 questions and the matching question bank contains 272 questions. Multiple choice and matching tests may be generated by course competency and software type.

#### Performance Test Banking

The performance test bank is a collection of structured, enumerated instructions for the printing of a performance test based on software type and difficulty level variables. The system contains 17 performance tests. In addition, this bank stores short question quizzes with questions to be answered after completing each performance

test. When adding a new performance test, the user is asked to assign tasks being measured by the performance test. A report of the tasks may be used for diagnosing weaknesses by a student or the class.

### Test Generation

The system allows the user to generate a test based on two main choices: (1) course and competency or (2) software type, function, and difficulty. Generating parallel versions of a test is possible with LessonBank. The tests may be generated randomly from the questions that match the variables chosen or the user may individually select the questions that match the criteria specified in the variables. The logic of the program creates a temporary grouping of questions that match all specified variables. From this group the user can specify the number of questions desired. A test key is printed listing correct answers as well as tasks measured. A menu choice lets the user print multiple copies of the test (without a key) for student use rather than going to another source to prepare additional copies of the test.

### Reporting

Another function of LessonBank is the ability to produce various reports. Reports available are (a) task listings by variables course and competency, software type, function and difficulty, (b) competency listing by course,

(c) competency and task listing by course, (d) lesson plans by course and competency, and (e) test bank question listing.

All reports and tests may direct the output to: (a) the screen, (b) a printer, or (c) a file. The file output may be stored on a hard disk or a floppy disk. This file is a DOS text (ASCII) file and may be retrieved and reformatted in any word processor.

The final menu choice provides the capability of completing various utility disk operating system commands including backup, format, directory, copy, check disk, and restore. A database "packing" routine is provided by the DBMS and is necessary to restrict the file size of the database from becoming inflated with unused space as the data are manipulated.

#### Sample Menus, Screens, and Reports

The system provides a main menu with seven choices as shown on the next page. Each of the seven choices from the main menu leads the user to another menu. The menu on level two may or may not generate a third level menu. Examples one and two on the following pages lead the user through a hierarchy of menus. A complete listing of menus is found in Appendix B.

### Example One

[Main menu]:

```
Task, Competency, and Course Data
Lesson Plan
Multiple Choice Tests
Matching Tests
Performance Tests
Utilities
Exit
```

[Task, Competency, and Course Data] menu:

```
Add/update task data
Add/update competency data
Add/update course data
Assign tasks to competencies within courses
Print reports
Exit to main menu
```

[Add/update task data] menu:

```
Add task data
Edit task data
Delete task data
Look at task information
Exit to previous menu
```

In order to add a new task, the user must decide on three things: one software type, one function, and one difficulty. The [Add/update task data] menu option displays three screen forms with prompts to guide the user through the process of selecting choices that assign the variables software type, function and difficulty to the task to be added as shown in Figure 1. The user does not have to type any data into the system and chance a misspelling. The system is designed to make it simple for a user to view the

existing software type, functions, and difficulty and press a key to select each. This procedure is designed to protect the integrity of the data. The DBMS uses the function keys [F7] and [F8] to move to the previous and next rows in an existing table rather than the up and down arrow keys.

Figure 1. Select software type, function, and difficulty forms.

Note: Mark only one software type.

Mark      Press [S] to mark selection; [spacebar] to unmark  
Choice    Press [F8] to move down; [F7] to move up

✓	1	DOS and Fundamentals
	2	Word Processing
	3	Spreadsheet
	4	Database
	5	Graphics and Graphing
	6	Data Communications
	7	Desktop Publishing

Press [ALT] when finished.

Note: Select only one function

Mark      Press [S] to mark selection; [spacebar] to unmark  
Choice    Press [F8] to move down; [F7] to move up

✓	1	Vocabulary
	2	Access Software/Data
	3	Data/Text Entry
	4	Editing
	5	Formatting
	6	Printing
	7	File Management
	8	Production
	9	Troubleshooting

Press [ALT] when finished.

Note: Select only one difficulty level.

Mark      Press [S] to mark selection; [spacebar] to unmark  
Choice    Press [F8] to move down; [F7] to move up

✓	1	Beginning
	2	Intermediate
	3	Advanced

Press [ALT] when finished.

After selecting software type, function, and difficulty, the user views the screen shown in Figure 2. A complete description of the new task should be entered. The system supports word wrap as would be found in a word processor.

Figure 2. Enter a new task form.

YOU ARE ADDING TASK DESCRIPTIONS FOR:

Software type: DOS and Fundamentals

Function: Vocabulary

Difficulty: Beginning

Enter the complete task description below. The program automatically assigns the task number.

Task No.	Task Description
2192	

Press [ALT] when finished and

- 1) choose Add Row to save. (User may add additional tasks)
- 2) choose Add Row and Exit if not adding more tasks.

To edit the description:

Press [ALT] to return to area. The current mode is overstrike. However, the [Insert] key may be used for editing.

[Edit task data] menu:

Change the description of a task Change the function of a task Change the difficulty of a task Exit to previous menu
---

After choosing the [Edit task data] menu option from the [Add/update task data] menu, the user is cycled through the screens shown in Figure 1 on page 56 that let the user select the variables software type, function, and difficulty

of the task to be edited. The screen that follows is shown in Figure 3. One task is displayed that matches the variables selected. The user presses the [F7] key for a previous description and [F8] to see the next description.

Figure 3. Edit task description form.

You are editing Task Descriptions for:

Software type: DOS and Fundamentals  
Function: Printing  
Difficulty: Beginning

Edit the task description(s) as desired  
Press [F8] to move down; [F7] to move up; [ALT] when finished.

Task Description
You may enter a complete description of the task here. It will wrap automatically. The description should fit in the space shown? No.

To [Change the function of a task] from the [Edit task data] menu, the user selects the three variables as shown in Figure 1 on page 56. Following the selection by the user, all tasks are displayed that match the criteria. The form used which displays the listing of tasks is shown in Figure 4 on the next page. The user selects the one to change by pressing an alphabetic key to mark the desired task. The form indicates that the user should press the [S] key; however, any alphabetic key will work. The form in Figure 5

displays the current function for the task selected and prompts the user to mark the new function by pressing the [S] key to mark. Upon exit from the form, the new function is automatically assigned and the database updated.

Figure 4. Selecting a task to change the function form.

You are picking the task to change

Press [ALT] when finished.

Mark Choice	Press [S] to mark selection; [spacebar] to unmark Press [F8] to move down; [F7] to move up
	1151 Identify backup (copy of a file to a disk)
	1152 Identify bit
	1153 Identify boot
	1154 Identify byte
	1155 Identify central processing unit
	1157 Identify the clear screen (CLS) command
	1158 Identify cold boot/start
	1159 Identify command
	1160 Identify the copy (COPY) command
	1161 Identify cursor
	1162 Identify the date (DATE) command
	1163 Identify default disk drive

Figure 5. Change the function of a task form.

You are choosing a new function for:

Software type: DOS and Fundamentals  
Function: Printing  
Difficulty: Beginning

Code: 6  
Code: 1

Mark Choice Press [S] to mark selection; [spacebar] to unmark  
Press [F8] to move down; [F7] to move up

✓	1 Vocabulary
	2 Access Software/Data
	3 Data/Text Entry
	4 Editing
	5 Formatting
	6 Printing
	7 File Management
	8 Production
	9 Troubleshooting

Press [ALT] when done finished.

To [Change the difficulty of a task] from the [Edit task data] menu, the user selects software type, function, and difficulty as shown in Figure 1 on page 56. After selecting the three variables, all tasks are displayed that match the criteria as shown in Figure 4. The user marks the one to change and Figure 6 is displayed.

Figure 6. Change the difficulty of a task form.

You are choosing a new difficulty level for:

Task:	Identify backup (copy of a file to a disk)		
Software type:	DOS and Fundamentals		
Function:	Vocabulary	Code:	1
Difficulty:	Beginning	Code:	1

Note: Select only one difficulty level.

Mark Choice      Press [S] to mark selection; [spacebar] to unmark  
Press [F8] to move down; [F7] to move up

✓	1	Beginning
	2	Intermediate
	3	Advanced

Press [ALT] when finished.

After marking the new difficulty level and exiting from the screen, changes are recorded in the task table of the database.

## Example Two

[Main menu]:

```
Task, Competency, and Course Data
Lesson Plan
Multiple Choice Tests
Matching Tests
Performance Tests
Utilities
Exit
```

[Multiple Choice Tests] menu:

```
Create a multiple choice test
Add new multiple choice questions
Print questions from test bank by software type
Edit or delete existing multiple choice questions
```

[Create a multiple choice test] menu:

```
By course and competency
By software type, function, difficulty or combination
Print student copies of previous test
Exit to previous menu
```

In example two of the menus, the user selects [Multiple Choice Tests] from the main menu in one of three ways. The user may double click with the left mouse button, type the first letter of the menu choice and press enter, or cursor up/down to the choice and press the enter key.

To create a multiple choice test based on course and competency, the user selects the course on which to base the test as shown in Figure 7. Figure 8 displays all the competencies for the course selected.

Figure 7. Course selection form.

Mark Choice	Press [S] to mark selection; spacebar to unmark Press [F8] to move down; [F7] to move up
✓	6320 Accounting
	6613 Accounting Computer Applications
	6612 Business Computer Applications
	6430 Business Supervision and Management
	6340 Clerical Accounting I
	6611 Computer Concepts
	6640 Data Processing I
	6650 Data Processing II
	6625 Information/Word Processing
	6610 Keyboarding
	6735 Legal Office Procedures
	6635 Management Information Systems

When finished, press [ALT].

Figure 8. Competency selection form.

Press [ALT] when finished.

Mark Choice	Press S to mark selection; spacebar to unmark Press [F8] to scroll down; [F7] to scroll up
✓	1 Using a spreadsheet software package, create a payroll template for salaried employees. Include soc. sec. number, name, beg date, and
	7 Develop bar graphs for expenses (salaries, travel, shipping charges, etc.) spent during a specific period
	22 Complete an electronic spreadsheet application
	23 Complete and verify business forms and records

LessonBank begins searching the database for questions in the test bank that match the choices made. If no questions are found that match the selections, the user is shown the messages in Figure 9.

Figure 9. Message display screen.

No questions in the database  
Match the criteria.

Press any key to continue.

If questions are found that match the criteria chosen, the user is prompted to indicate if the program should randomly select the questions. Figure 10 displays the screen as seen by the user.

Figure 10. Random selection option screen.

Yes  
No

Do you want the program to randomly select the questions?

If the user selects "yes" from the screen, Figure 11 appears, displaying the number of questions that match the criteria, and asks for the number of questions to be

selected. The number must be between 1 and the number shown as matching the criteria. An error message will caution the user if a number is entered that is larger than the number of possible questions. In addition, this form prompts the user to enter the lines for the heading that will appear at the top of the test.

Figure 11. Random selection heading form.

Add/discard   Go to   Exit

The number of questions that match your criteria is 137  
How many questions shall the program select for you? 20

Line 1: Word/Information Processing  
Line 2: Test I  
Line 3:

Enter the test heading with a maximum of three lines.

When finished, press [ALT] and  
(1) choose Add/Discard to save and exit.  
(2) choose GoTo and press [ENTER] to edit heading or number.

LessonBank will randomly select the test questions for you.

(Note: [ALT] key toggles between form and menu.)

Figure 12 displays the printing menu asking the user to choose the output location.

Figure 12. Print menu.

Screen  
Printer  
File  
Exit this menu

If the [File] option is chosen from the print menu, the user is asked to select a destination drive by highlighting the choice and pressing enter. Figure 13 displays the destination drive menu and a dialog which follows. The program indicates the drive letter selected and prompts the user to make sure the correct disk is in the drive. A valid DOS file name is requested. The test is saved to a DOS text (ASCII) file and may be retrieved in a word processor.

Figure 13. Save to a file screen.

Destination Drive --- Press [Esc] To Leave This Menu				
A:	B:	C:	D:	E:

Sending Report to drive A:  
Make sure the correct disk is in the drive  
Are you sure? (Y/N) y  
Enter valid DOS filename: courses.txt

Figure 14 shows a sample multiple choice test format. Figure 15 displays the answer key. Figure 16 displays a report generated for the teacher indicating the competencies selected as the basis for the test.

Figure 14. Multiple choice test.

Name \_\_\_\_\_

Accounting Computer Applications  
Test on Competency 22  
October 1, 1992

Multiple Choice: Write the letter of your answer in the answer column.

\_\_\_\_\_ 1. After creating a spreadsheet, the program will allow the user to retain the spreadsheet by:  
A. saving  
B. scrolling  
C. starting  
D. quitting

\_\_\_\_\_ 2. Normally, the user may access the \_\_\_\_\_ anytime while working in a spreadsheet.  
A. file function  
B. shift print directory  
C. scroll menu  
D. help function

Figure 15. Multiple choice answer key.

ANSWER KEY  
MULTIPLE CHOICE TEST

Question No.	Answer	Task No.	Smart No.
1.	A	1022	321
2.	D	1024	321
3.	B	1027	321
4.	D	1028	321
5.	D	1028	321
6.	A	1029	331
7.	A	1031	331
8.	A	1036	331
9.	C	1038	332
10.	D	1050	341
11.	C	1053	341
12.	B	1059	351
13.	A	1061	351
14.	D	1114	341

Figure 16. Test basis report.

Date Printed: 08/31/1992

Accounting Computer Applications  
Test on Competency 22  
October 1, 1992

The test is based on the following course and competencies:

6613 Accounting Computer Applications

22 Complete an electronic spreadsheet application

If the answer is "No" for random selection, the number of questions matching the criteria is shown and the user is prompted to enter up to three lines in the test heading as shown in Figure 17.

Figure 17. User selects questions heading form.

Add/discard Go to Exit

The number of questions that match your criteria is 10.	
Line 1:	Computer Applications
Line 2:	TEST I
Line 3:	Mrs. Smith's first period

Enter the test heading with a maximum of three lines.

When finished, press [ENTER] and

- 1) choose Add/Discard to save and exit.
- 2) choose GoTo and press [ENTER] to edit heading.

You may then choose the exact test questions.

(Note: [ALT] key toggles between form and menu.)

The form in Figure 18 is displayed and the teacher may mark as many questions as desired.

Figure 18. Multiple choice question selection form.

Press [S] to select as many questions as you desire; [spacebar] to unmark.  
Press [F8] or [ENTER] to move down; [F7] to move up through the data.

When the user wants to leave a spreadsheet program, he/she can select the:
A. leave command
*B. quit command
C. remove command
D. function command
The process of moving horizontally or vertically through the worksheet is:
A. windowing
B. menuing
*C. scrolling
D. editing

Asterisk [\*] marks correct answer.

Press [ALT] to verify the number of questions selected.

To mark, press [S] and [ENTER]; to unmark press [SPACEBAR].

To verify the number of questions already selected or prepare the test, the user must press the [Alt] key and a popup menu appears. The menu is shown in Figure 19 as it appears over the question selection form.

Figure 19. Popup menu displaying question count.

Press	Continue--Prepare test now	e; [spacebar] to unmark.
Press	Redo--Add or subtract questions	p through the data.
	Stop--Cancel and return to previous menu	
✓ Nor	Question count = 5	anytime while
wor		
A. file function		
B. shift print directory		
C. scroll menu		
*D. help function		
✓ A series of choices that appear on the control panel:		
*A. main menu		
B. main cell address		
C. label menu		
D. formula		

Asterisk [\*] marks correct answer.

Press [ALT] to verify the number of questions selected.

Make a choice and press [ENTER].

In example two of the menus, the user selects [Multiple Choice Tests] from the main menu. The user makes the appropriate menu choice to [Create a multiple choice test] based on software type, function, and difficulty or combination.

[Multiple choice tests] menu:

Create a multiple choice test Add new multiple choice questions Print questions from test bank by software type Edit or delete existing multiple choice questions
--

[Create a multiple choice test] menu:

By course and competency By software type, function, difficulty or combination Print student copies of previous test Exit to previous menu
---

The second option on this menu allows the user to build a test based on three choices: software type, function, and difficulty. In addition, users may choose a combination of any of the three options. Figure 20 displays the form for the user to select questions based on one or more types of software.

Figure 20. Select a software type form.

Mark Choice	SELECT A SOFTWARE TYPE:
	1 DOS and Fundamentals
	2 Word Processing
✓	3 Spreadsheet
	4 Database
	5 Graphics and Graphing
	6 Data Communications
	7 Desktop Publishing

Press [S] to mark selection; [spacebar] to unmark  
Press [Enter] or [F8] to move down; [F7] to move up  
Select as many choices as you desire.

Press [ALT] when finished.

Figure 21 displays the form used to select one or more functions. Selecting a function will have the effect of limiting the matching data. Selecting no function by pressing the [Alt] key will choose them all. Figure 22 displays the difficulty selection form. Users may select as many choices as desired or select none.

Figure 21. Select a function form.

Mark Choice	SELECT A FUNCTION:	
	1	Vocabulary
✓	2	Access Software/Data
✓	3	Data/Text Entry
✓	4	Editing
✓	5	Formatting
	6	Printing
	7	File Management
	8	Production
	9	Troubleshooting

Press [S] to mark selection; [spacebar] to unmark  
 Press [Enter] or [F8] to move down; [F7] to move up

Select as many choices as you desire. Making one or more selections will limit the matches. Making NO selection actually chooses them ALL.

Press [ALT] when finished to move to difficulty level.

Figure 22. Select a difficulty form.

Mark Choice	SELECT A DIFFICULTY:	
✓	1	Beginning
	2	Intermediate
	3	Advanced

Press [S] to mark selection; [spacebar] to unmark  
 Press [Enter] or [F8] to move down; [F7] to move up

Select as many choices as you desire. Making one or more selections will limit the matches. Making NO selection actually chooses them all.

Press [ALT] when finished.

For example, choosing spreadsheet (software type), vocabulary (function), and beginning (difficulty) will generate a listing of questions that match specifically those three criteria. Choosing spreadsheet, vocabulary, and no difficulty will generate questions matching ALL spreadsheet terms.

The questions matching the criteria will be searched and selected by the system. A popup menu as shown in Figure 10 on page 63 will be displayed asking the user for a yes or no response to a randomly generated test. After the user makes the yes or no selection, the program proceeds as displayed in Figures 12-19.

#### LessonBank Design Elements

This instructional management system was written to help teachers and administrators realize the potential of using a database in education. LessonBank has the purpose to help teachers, administrators, and supervisors take advantage of the capability of the computer as a tool and to assist effectively in the administration of their duties. This project is unique in its application as it is directly associated with the Business Computer Software Curriculum Series published by the Virginia Department of Education in 1991 and the Business Education Suggested Course Competencies and Performance Objectives, published by the

Virginia Department of Education in 1989. The application should help teachers to better manage their instruction.

The program uses a menu-driven application based upon a relational database management system. Users should understand there is a difference between a database, a database management system, and an application.

A database is a self-describing collection of integrated records (Kroenke, 1992); the database consists of the raw data itself (fields and records or columns and rows), the overhead data or schema (the physical description of the structure of the database), and the meta data (structure of forms and reports). The database management system (DBMS) is a sophisticated, powerful program that a user or developer purchases "off the shelf" to create a specific database and database application. The database management system should allow the developer to define the database, create various reports and forms, and the application. The application normally has menus from which a user makes selections. Using an application is a way of controlling what processing the user is able to do and allows users to store, retrieve, and manipulate data without having to understand the underlying structure of the database itself. The application allows users to input data with user-friendly on-screen forms and allows output to one of three resources (screen, a hard copy printout, or to a

text file). A relational database establishes ways of linking and integrating the data into useful information for the teachers.

This project was developed using a relational database composed of named tables, which may be divided horizontally into unnamed rows and vertically into named columns.

A relational database is structured around the concept of two dimensional tables. Information systems professionals utilize one set of the terminology in Table 3.

Table 3

A comparison of database and file processing terms

Relation	Table	File
Tuple	Row	Record
Attribute	Column	Field

The terms in column three are used in most file processing applications. Database management systems use the terminology in either columns one or two. R:Base uses table, row, and column. Teachers should be able to relate the two dimensional structure of the table to the familiar spreadsheet structure of rows and columns.

In the following examples, table names are presented in all capital letters. Column names begin with initial caps and may contain no more than eight characters. Keys to each table uniquely determine the contents of the row. Key

columns are indicated by underlining them. For example, the course table contains information about the course number and the course name:

COURSE

<u>CorsNbr</u> , CorsName
---------------------------

Databases are composed of tables that have a one-to-many (1:N) relationship allowing the user to link data from one table to the other. By storing all the information about one entity in a table, such as the information about courses, the table can be used later to retrieve information.

Competencies associated with each course must be identified. The competencies selected for this project were competencies involving the use of various software types. A competency table that contains the course number, competency number, and competency description was designed as follows:

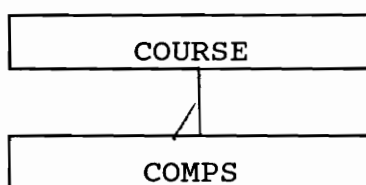
COMPS

<u>Compnbr</u> , <u>CorsNbr</u> , CompDesc
--

One course could have many competencies related to it. This is an example of a one-to-many (1:N) relationship. The competency itself has a description, a number, and a particular course to which it is related.

A graphical tool is helpful in depicting the relationship in a database. The tables in this model were represented as boxes with the table name in all capital

letters. The relationship existing between tables is shown by drawing a line between the boxes. The relationship between COURSE and COMPS is one-to-many. This relationship is indicated by putting a reverse arrow on the "many" side of the relationship.



A determination was made that in order to achieve a given competency, a student should complete a number of tasks. The task is an activity to be completed or mastered to meet the competency. In 1990 several researchers at Virginia Polytechnic Institute and State University developed a validated list of approximately 1,100 tasks covering a variety of software types. The software types form another table. The software type was given a number (for purposes of developing the key, which uniquely identifies a row in a table), a name, and a code. In addition, these tasks were also further categorized by what was called function. Nine function categories were identified. The functions then became a table with a function number as a key and a function name or description as well. The tasks were also to be categorized according to

difficulty. Three difficulty categories were chosen and they formed another table. The tasks themselves were assigned a key. Each task also had a description.

The following tables were developed as described above:

SOFTTYPE

<u>SoftType</u> , SoftName, Kode
----------------------------------

FUNCTION

<u>Function</u> , FuncName
----------------------------

DIFFICUL

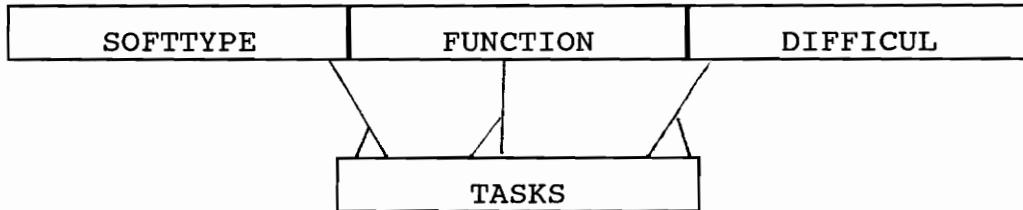
<u>Difficul</u> , DiffCode, Diffname
--------------------------------------

TASKS

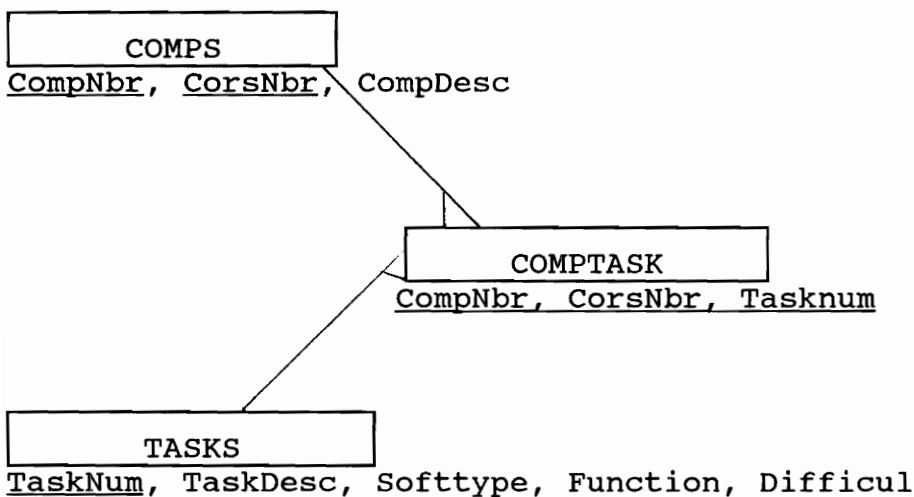
<u>TaskNum</u> , TaskDesc, SoftType, Function, Difficul
---

The developers modeled software type having a 1:N relationship with tasks as each one of the seven software types would have many tasks. A function had a 1:N relationship with tasks as one function had many tasks associated with it. Difficulty had a 1:N relationship with tasks as each difficulty level had many tasks associated with it. To model one-to-many relationships, the key from the "one" side of the relationship was placed in the table for the "many" side. The relationship is shown in the TASKS table where the keys for softtype, function, and difficulty appear as illustrated above.

The following is a model of the 1:N relationship between software type, function, difficulty, and tasks.



Modeling tasks and competencies was a little more difficult. A task could have many competencies and a competency could have many tasks. As a result the relationship would be many-to-many (M:N). Since M:N relationships are difficult to model in a relational database, the relationship needed to be changed to one-to-many (1:N). The 1:N relationship was established by creating another table that contained only the keys from the two tables in the M:N relationship.



Each of the tables, COMPS and TASKS, now have a 1:N relationship with the third associated table, COMPTASK.

The relationships between tables were established so the database can be used with either a relational command to create a new table from existing data or the program can look up data in the associated table. Relational commands, such as join, allow the developers to bring the desired columns and rows from the tables together and eliminate data redundancy. Temporary tables may be created for the purpose of printing a report that requires data from several tables.

Each table with a relationship to another must have a linking column, called the "foreign key." The foreign key is the key of a table on the "one" side of a one-to-many relationship. The 1:N relationship may be called a parent-child relationship. The key of the parent table is placed in the child table. The SOFTTYPE, FUNCTION, DIFFICUL table keys are placed in the TASK table as foreign keys. The placement of the foreign keys established a link between the tables to allow data look up of names and descriptions.

The database data is controlled by the application. This application presents the users with menu choices for maintaining the various components of the database itself. Those components are the tasks, the courses, the competencies, and the multiple choice, matching, and performance question banks. The database application should

allow users to add data to all of these areas as well as delete and modify existing data. Users should be able to print various reports.

Some developers call these reports "views" because they represent various ways of looking at the data. For example, the state supervisor might want a list of all courses taught and a list of competencies associated with each course. A teacher might want to see a list of the tasks associated with a specific competency within a particular course. Another user might want a list of tasks for a given software type. Users can select one of the software types and print a list of tasks associated with that software type (or more than one type). In addition, a teacher could narrow the task list to just one or two functions. The teacher might want to print a list of tasks by difficulty. The list could contain only the beginning tasks of a software type.

To make the database even more useful, a test bank with multiple choice and matching questions was developed. Each test question is associated with a task forming a 1:N relationship between the test question tables and the task table as illustrated on the next page.

#### MULCHOIC

<u>Stem</u> , AnswerA, AnswerB, AnswerC, AnswerD, Answer, <u>TaskNum</u>
--

#### TASKS

<u>TaskNum</u> , TaskDesc, SoftType, Function, Difficul
---

#### MATCHING

<u>MaStem</u> , MaAnswer, <u>TaskNum</u>
--

The developers made the decision that for purposes of database design there would be only one task associated with a given question. However, a task could have many questions to support it. The current systems contains 359 multiple choice questions and 272 matching items. Questions do not currently exist for all tasks. However, users should keep in mind that a test is generated based on either competencies or software type. Both of these categories will have a number of tasks assigned to them thus generating a test with a reasonable number of questions.

The questions themselves (of the multiple choice and matching variety) were separated into two tables because of their structure. The matching questions contained a definition (MaStem) and an answer. The multiple choice questions would contain a stem, four possible choices and an answer. The matching questions and the multiple choice questions would each be associated with a task, which would

have a number. The task number is the foreign key to both the MULCHOIC and MATCHING table and is on the parent side or the "one" side of the 1:N relationship.

A test bank would naturally have the capability for the teacher to add new questions, modify existing questions, and delete old questions. This particular application not only will allow users to process or maintain the test bank but also will allow for the creation of tests based on various factors. These factors are (1) software type, (2) function, (3) difficulty, (4) course and competency.

To make it easy for the teacher to construct (create) a test, the program developers created a menu allowing the choices of creating a test by (1) course and competency and (2) by software type, function, and difficulty. If the teacher chooses to create a new test by course and competency, an on-screen form appears where the user selects the appropriate course and the competency(s) desired. The application then goes to the COMPTASK table and uses the relational projection command to create a new table that contains a subset of rows from the original COMPTASK table that meet the criteria. For example, if the course number was 6615 and the competency numbers were 22 and 23, the database would create a temporary table that contains only the task number where there is a match (=) on the columns course number and competency number. This new table will

contain a subset of rows and one column (task number). Once the database management system has created the new table, then the relational intersect command is used to create another table containing all the data from the question table that match the task number. This new table is used to print the test and the key. The same basic procedure is used for selecting software type, function, and difficulty.

In addition, the teacher can select the questions individually by looking at the actual question or the program will randomly select a number of questions from among those that meet the given criteria. In this case the teacher indicates the number of questions wanted for the test.

Another part of the database includes performance tests. The performance test module can be added to, deleted, modified, or printed as well.

The application will allow a teacher to create a daily lesson plan based on competencies and tasks. The lesson plan can be distributed to the principal, parents, or students. The lesson plan can be written to a text file. This file can be retrieved in a word processor where data can be added or changed.

LessonBank: The Instructional Management System (Virginia Department of Education, 1993) is a program written for Virginia business educators, but could be used

as the shell for any system identifying courses, tasks, and competencies. The database and application programs were written and revised over a two-year period (1991-1993).

## CHAPTER 4

### FORMATIVE EVALUATION PROCEDURES

The purpose of this project was the formative evaluation and revision of a database application to be used by business teachers and program developers to provide an efficient and effective means of managing instruction in the classroom. The procedures used to complete the project are delineated in this chapter. The topics discussed include: (a) project design, (b) subjects, (c) data collection, and (d) data analysis.

#### Project Design

A formative evaluation project design was used to field test and revise the computer program LessonBank: The Instructional Management System. The program was based in part on computer competencies extracted from the Business Education Suggested Course Competencies and Performance Objectives, published by the Virginia Department of Education in 1989. The competencies in the curriculum guide were developed to assist business educators with the implementation of updated competency-based courses in administrative systems, business management, and information systems programs.

A subset of competencies which measured computer software skills was extracted from the 1989 competency-based curriculum guide and a listing of tasks identifying specific

actions to be demonstrated to achieve a competency was developed. The compilation of over 1,100 individual tasks was prepared and organized by a research project team from Virginia Polytechnic Institute and State University and resulted in the Business Computer Software Curriculum Series published by the Virginia Department of Education in 1991.

The purpose of the Business Computer Software Curriculum Series was to assist business educators plan instruction on the sizable number of computer operations required for occupations in business.

LessonBank: The Instructional Management System is a program written for Virginia business educators, but could be used as the shell for any system identifying courses, tasks, competencies, and software types. The database and application programs were written and revised over a two year period (1991-1993). The developers used the programming language that is part of R:Base for DOS, Version 3.1C. A runtime version of R:Base is provided without charge to each user of the program.

#### Formative Evaluation Design

A formative evaluation design was followed with a prototype of the program. Three phases were used: (a) initial program testing, (b) small group evaluation, and (c) field testing as suggested by Dick & Carey (1985).

### Phase One Evaluation

Because of the complexity and size of LessonBank, the developer used a one-to-one evaluation with one individual user to obtain an initial reaction to all components of the system. The developer met with the user for an orientation session in which the purpose of the instructional management program and the one-to-one testing was explained. A user's manual was developed for evaluation during phase one. The user was informed that objective reaction to the materials was important to the success of the project. The developer and user met, loaded the program in the microcomputer, and the developer explained how the program was to operate. The developer then observed as the subject used the program. The developer interacted with the user following each major choice from the main menu and made notes concerning every suggestion for improvement. No coaching was allowed. The user was given an assignment for each menu choice designed to test each component of the program. The developer observed by recording places where errors were made, questions were asked, and instructions were not clear. Notes were made of all comments and suggestions as well as alternative explanations made by the developer. The purpose of this first stage of formative evaluation was to identify and eliminate the most obvious errors and obtain initial reactions to the program (Dick & Carey, 1985). The

developer interviewed the user following the initial program analysis. The conversation was tape recorded, and the tape was reviewed prior to making revisions to the program.

### Program Revisions

Suggested revisions to the program were evaluated and implemented following the one-to-one evaluation phase. Revisions to improve the effectiveness of the program could include changes to (a) application menus, (b) instructions and directions, (c) arrangement of data and prompts on forms, (d) arrangement of data on all reports, and (e) addition of other features. Revisions were made during August and September, 1992, and the user's manual was expanded for evaluation during phase two.

### Phase Two Evaluation

Phase two of the formative evaluation contained seven subjects who were as nearly representative of the target population as possible. The subjects were chosen from members of a graduate class in instructional methods in cooperative marketing and office education programs at East Carolina University. Subjects were to load the program and complete the structured assignments on their own. The subjects were given an assignment for each menu choice designed to test each component of the program. The purpose of the small-group evaluation was to determine the effectiveness of changes made following the one-to-one

evaluation and to identify any remaining problems. A secondary purpose was to determine if subjects could use the program without interaction with the developer.

Users were to record suggestions and problems on each assignment sheet and return the completed task assignments to the developer after using the software.

### Phase Three Evaluation

Phase three, the field trial, involved distribution of the revised LessonBank program to 15 volunteer users representing the target population during January, 1993. The developer contacted a vocational business education specialist in the Virginia Department of Education in November, 1992, who provided the names and addresses of local business education supervisors and vocational administrators. The specialist marked the names of 25 supervisors who were subsequently contacted by letter (Appendix C) and asked to provide the developer with the names of two teachers who may be interested in participating in the study. Fifteen supervisors responded to the request. The developer followed up with a letter to 30 teachers (Appendix C). From that number 15 volunteered to participate in the study. The developer prepared 15 copies of the program, user's manual, structured assignment sheet, check sheet, and evaluation form and mailed the entire package on January 4, 1993.

The effectiveness of the database application may best be measured in the actual classroom setting during the field trial. Users were to complete an assignment for each menu choice designed to test each component of the program, return the assignments to the developer for checking, and complete a check sheet during use and an evaluation form after using the software. The field test should determine the functionality of the product under normal conditions.

### Subjects

The target population for this study were business education teachers. Volunteers were solicited for the one-on-one, small group, and field test stages of program development.

### Data Collection

The following instruments and procedures were used to collect data for the study.

#### Developer's Observations

During phase one of the evaluation only, the developer observed the user and recorded all problems and requests for assistance. Problems were recorded to aid in the revision process. No observations were made during phases two and three as these subjects were to use the database application on their own and return the results of the structured assignment with suggestions and comments.

### Structured Application Assignments

A structured application assignment was used to measure the ability of the users to successfully access each of the 68 menu choices of the program. In phase one, the developer observed and recorded problems as the subject completed the structured assignments. Users in phases two and three were asked to record any problems and suggestions for improvement. The structured assignment is included in Appendix D.

### Data Analysis

Data were analyzed to identify problems for revision. Problems were recorded and changes were incorporated in the program.

## CHAPTER 5

### RESULTS

The purposes of this project were to (a) evaluate and revise a computer-based instructional management system and (b) develop and revise documentation for using an instructional management system. The instructional management system consists of a database and various applications employing relational database architecture. The resulting system will be used by Virginia business teachers in implementing their curricula. Seven project objectives were addressed in the undertaking. The results of the project are detailed in this chapter. The objectives are discussed in the order in which they were addressed in the project.

#### Objective 1

The first objective was: Develop written documentation for using LessonBank in the form of a user's manual. This objective was addressed by the development of a 12-page manual containing the following topics:

(a) a detailed description of the capability and contents of LessonBank to familiarize the user with the structure of the system,

(b) a description of each of the six main menu choices,

- (c) the system requirements including the size and type of computer required to run the program and the basic configuration file necessary,
- (d) the installation procedure to be employed by the first-time user,
- (e) instructions on starting the program,
- (f) instructions on accessing the menu choices, and
- (g) a brief overview of special keys.

The manual was developed in July, 1992, to be used with the first teacher who would evaluate the program in a one-to-one setting.

#### Objective 2

The second objective was: Conduct a one-to-one evaluation with a single individual to obtain an initial reaction to all components of LessonBank and the user's manual. The program was tested by a volunteer representative of the target population in August, 1992. The teacher was an experienced high school teacher and rated herself on the scale of (a) novice, (b) casual user, (c) heavy user, or (d) expert, as a "heavy user" of computers. The developer felt that a heavy user would profile the target population since user's of this system would be teachers who taught business computer skills within a variety of courses. The user loaded the program herself following the instructions in the user's manual. A

structured assignment to test each of the menu choices was provided in writing to the subject. The developer observed the user interact with the program and made notes concerning any problems. The conversations were recorded on tape and reviewed before changes were made to the program.

The following section discusses the problems that were identified during phase one of the evaluation process and the corrective action taken. Errors were grouped according to the following categories:

1. Suggested changes for data input (screen forms).
2. Suggested changes for user's manual.
3. Suggested changes for output (reports).
4. Processing errors.
5. Suggested new features.

#### Suggested Changes for Data Input (Screen Forms)

1. Problem: Several instructions on the three-page form used to add a new performance test required further clarification. Specific suggestions were:

a. Instruction needed to explain that [F7] and [F8], and [Tab] keys are used to move from one item to another to edit a mistake. User became confused and did not know how to correct an error when discovered in an instruction already typed.

b. Initial instruction needed on second screen.

Subject thought the form should provide a prompt to the user to "Enter task number and press [Enter]."

c. Consistency is needed for data entry of numeric items. The first screen, which contained numbered test instructions, did not require that a period be placed following the enumerated items, and the third screen, which contained a numbered quiz, required that a period be placed following the number.

d. An instruction describing that the [Enter] key is used to move from question to question on the quiz form needed to be added.

e. A menu choice of "Add row and exit" needed to be added to the form.

Action taken: The instructions on the pages of the form were modified to incorporate all suggestions.

2. Problem: When choosing the menu option to edit a performance test, the user needed a screen prompt explaining how to select the test to edit or delete. User was not familiar with the cryptic information which appeared as a performance test number.

Action taken: The program module was rewritten to show the performance test number and the full title of the performance test. Instructions were added to the screen to guide the user as well.

3. Problem: When adding a matching question to the test bank, the user entered a task number. Once the task number is entered, the program displays the task description, software type, function, and difficulty. The order of the displayed data needed rearranging so it did not appear to jump around.

Action taken: Developer changed the order of look up data to provide a smoother display.

4. Problem: The form used to enter the test heading information needed to be modified to display a single choice of "Add row and exit" upon completion of entering data.

Action taken: Form menu modified.

5. Problem: The form used to enter the teacher's name and date for printing a lesson plan needed directions on how to exit and save.

Action taken: The form was modified.

6. Problem: Instructions needed to inform the user to press [Alt] key when finished on the form used for adding tasks assigned to a competency.

Action taken: The form was modified.

7. Problem: When choosing new difficulty level on tasks, the final screen should list complete task description and number.

Action taken: The code controlling this display was modified.

8. Problem: The form used to edit a competency number needed a statement indicating that the [Insert] key may be used when entering data to avoid typeover.

Action taken: The form was modified to accommodate the change.

9. Problem: The form used to add tasks needed instructions on what to do when finished.

Action taken: Statement added to form explaining procedure for adding more tasks or exiting without adding more tasks.

10. Problem: An explanation is needed on the three-table form showing software type, function, and difficulty an instruction that the software type and function data scrolls within the small box.

Action taken: The form fills up the entire screen and does not contain room for additional instructions. This instruction was moved to the user's manual.

#### Suggested Changes for User's Manual

1. Problem: User's manual needed to be expanded to contain full instructions and illustrations for the various forms.

Action taken: Developer rewrote the user's manual including detailed explanations of each menu choice and all sub choices. A screen capture utility was used to record

pictures of every screen to reduce for inclusion in the manual.

Specific problems include:

2. Problem: The manual should explain specifically which keys can be used for editing.

Action taken: Note made that [TAB] key, [Insert] key and the [End] key may be used for editing.

3. Problem: If the user exited before completely entering all instructions when adding a new performance test, the only way to return was to go back to the performance test menu and choose the Edit option.

Action taken: An explanation was added to the form and the user's manual on how to move backwards through the data using the [F7] function key.

4. Problem: The add/edit test performance form contains three pages. Instructions for its use were not clear.

Action taken: The instructions were clarified and expanded in user's manual and on data input form.

5. Problem: The performance test instruction data entered by the user would wrap as with a word processor.

Action taken: An explanation of this characteristic is written into the user's manual.

6. Problem: The subject requested an explanation that the up arrow may be used to reach the bottom of a menu or a list.

Action taken: Since this is only a partially true statement throughout the program, the suggestion that this instruction be included in the user's manual was not followed.

7. Problem: The manual needed to describe fully the procedure for deleting instructions in the performance test.

Action taken: A full explanation of the delete procedure was included in the user's manual indicating that the user was to press [Alt] and choose Delete row.

8. Problem: A thorough explanation of the two basic types of tests: (1) by course and competency, and (2) by software type should be described in the manual.

Action taken: A full explanation describing the types of tests was added to the user's manual.

9. Problem: Some explanation to the user about the lack of a way to "save" data in the traditional sense should be included.

Action taken: Reference to the concept of saving data automatically as one exits from the system was explained in the user's manual.

10. Problem: The use of a screen which has three data entry boxes was unclear. Each box contains choices of (1) software type (2) function and (3) difficulty. User encountered difficulty understanding how to move from box to box.

Action taken: Because of the limitation of lines in the form itself, the instructions for using this screen form were described in detail in the user's manual.

11. Problem: As the screen with three data boxes for software type, function, and difficulty appears, the user must wait until the program has displayed the data and placed it in the box before pressing a key.

Action taken: It was not possible to correct this problem as it was the slowness of the database management system retrieving the data to be displayed. The user should not be bothered by this after using the program a few times.

12. Problem: Some of the popup menus provided an explanation describing the action to be taken in the bottom right of the screen and others do not.

Action taken: The popup menus were all modified to provide a description of the action.

13. Problem: User did not understand that the reports, lesson plans, and tests could be saved in a file that could be retrieved in a word processor.

Action taken: Instructions concerning this concept were explained in the user's manual.

14. Problem: User was concerned that it might not be understood that one must follow separate types of entry procedures when using the course and competency picking form.

Action taken: This instruction was highlighted on the form and was added as a full explanation in the expanded user's manual.

#### Suggested Changes for Reports

1. Problem: The performance test needed to be printed in sorted order. This sorting should be controlled by the program and invisible to the user.

Action taken: This problem was corrected by modifying the programming code.

2. Problem: When saving a report to a file, users needed a message instructing them not to type a drive letter.

Action taken: The on-screen instructions for the user were modified to include a statement to type a valid file name and not to include a drive letter followed by a colon.

3. Problem: The report which printed all multiple choice questions by software type did not "page up" on the last page. In a page printer (like a laser printer) this means that the final page of the report remained in the printer; on a line printer the printing stopped after the last character.

Action taken: The report was modified within the database so that it contained an automatic form feed.

4. Problem: Same type of error occurred on reports (1) Business Computer Tasks by Course and Competency and (2) List of Tasks by software type, function, and difficulty.

Action taken: Both reports were modified to correct this problem.

5. Problem: User wanted to include the smart number in the lesson plan which shows the software type, function, and difficulty.

Action taken: Developer modified the lesson plan report to include the smart number.

#### Programming Errors

1. Problem: User was instructed to delete a performance test. It did not work.

Action taken: Developer checked the program again, tried the instruction and it did work.

2. Problem: The user made the choice to create a matching test by data communications, vocabulary, and beginning and intermediate difficulty levels. The user was then asked to answer "yes" if desiring a randomly generated test or "no" if user wants to select the questions. The user chose "yes" but meant "no." The subject chose Exit from the menu, but kept getting an error message stating that an invalid number of questions had been chosen. The program would not let the user exit.

Action taken: The programming code which controlled this action was modified and tested so that this condition did not occur again.

3. Problem: Some of the multiple choice questions appeared to be in the database twice.

Action taken: Developer checked the questions and removed several duplicate questions.

4. Problem: The choice to print a listing of competencies and tasks from the [Print Reports] menu in the [Task, Competency, and Course] main menu did not print.

Action taken: Programming code which had been omitted from the application was inserted in the proper place.

#### Suggestions for New Features

1. The user wanted to be able to verify that new questions had been added.

2. This user wanted to be able to pick difficulty level with course and competency.

3. The user suggested that a "smart number" be included in lesson plans.

#### Objective 3

The third objective was: Revise the user's manual into a more comprehensive document following interaction with the initial user.

Following the one-to-one evaluation it was apparent that the user's manual would need to be expanded to include

a full explanation of how the user would interact with each menu choice and the screens which followed. The developer was aware that words alone would make it difficult to understand how to use the many forms and screens since the program contains 40 individual forms, 68 menu choices, and 30 reports.

Fortunately, Microrim's R:Base 3.1C provided a "snapshot" utility that would capture the output on the screen which could be saved to a file. The file could then be retrieved in a word processor and printed. The majority of the screens used 75-80 characters so they could not be retrieved directly into the text of the manual because of the need to provide adequate space for margins.

Each screen snapshot was printed on a laser printer and reduced on a copier to 67 percent so it would fit within a six-inch line.

The developer accessed each menu choice documenting the procedure for each screen which followed. The written instructions were then typed using a word processor. The written copy was placed in a book format using PageMaker 4.0 for the PC. The page composition software made it possible to control the space where the screen snapshots were to be placed and make adjustments easily. The user's manual was then printed using a Times Roman font in 12 point type.

The revised user's manual expanded from 12 pages to 63 pages providing a comprehensive description of each menu choice. The revisions were made during September, 1992.

#### Objective 4

The fourth objective was: Revise LessonBank following an analysis of the one-to-one evaluation.

The developer worked with a volunteer representative of the target population in August, 1992. Following a review of the comments made during the structured assignments, the developer made changes to the program as described in Objective 2 on the previous pages. The revisions were made during August and September, 1992.

#### Objective 5

The fifth objective was: Conduct a field test with a small number of individual users to check effectiveness of changes made following the revision process and identify remaining problems.

Phase two involved a group of individual evaluations from among a group of nine individuals who were graduate students in an instructional methods in cooperative marketing and office education programs at East Carolina University, Greenville, North Carolina. The professor and students were interested in examining a computer-based instructional management system and agreed to each load the program, complete the 17-page structured assignment, and

return the results to the developer along with comments. The evaluations of the first seven members who returned their work according to the deadline set by the developer are reported in the following paragraphs. The suggestions and comments were grouped according to the same categories in objective two:

1. Suggested changes for data input (screen forms).
2. Suggested changes for user's manual.
3. Suggested changes for output (reports).
4. Processing errors.

#### Objective 6

The sixth objective was: Make further revisions to the program and user's manual following responses from individual evaluations in phase two field test.

##### Suggested Changes for Data Input (Screen Forms)

1. Problem: The form for entering the teacher's name and date on lesson plan told the user to press [Enter] when pressing [Alt] was correct.

Action taken: Either procedure worked correctly. Instructions were modified on the screen form.

2. Problem: Subject expressed concern about using [Shift/F8] to move from area to area while at other times the user had to use the Alternate key.

Action taken: These two limitations were controlled by the program R:Base. The Alternate key was primarily used

when finishing an operation, and the [Shift/F8] was used to move to a separate table. Changing or controlling the use of the function keys by the developer is not possible. This is a limitation of the R:Base program itself.

3. Problem: The word "inserted" was misspelled in the instruction paragraph of "Editing performance test."

Action taken: Spelling was corrected.

4. Problem: User did not know how to proceed to change competency number.

Action taken: Instructions on screen form indicated that user should press [Shift/F8] to enter a new number. This instruction could be easily overlooked so it was moved to a new location on the form.

5. Problem: Too many instructions on the add competency data screen confused the user.

Action taken: Instructions revised and reworded to clear up screen clutter.

6. Problem: Form used to edit the name of a course needed "Press Alt when finished."

Action taken: Instruction was added to the form.

7. Problem: Instructions for exiting from heading form were not clear.

Action taken: This problem was corrected. The form had been modified earlier, but as the form was displayed on

the screen it was retrieved too low and the instructions did not appear.

8. Problem: The user had difficulty exiting from difficulty level before printing the test. User did not want to choose a difficulty level.

Action taken: The program code was checked and it was determined that it was possible to exit without selecting a difficulty level. Further explanation was added to user's manual.

9. Problem: The user needed a way to change the test number in performance tests.

Action taken: The program code for this module was rewritten to make numbering tests automatic.

#### Suggested Changes for User's Manual

1. Problem: User reported difficulty in moving around in print report options in tasks, competency, and course data.

Action taken: Full explanations in the use of the form were expanded in the user's manual.

#### Suggested Changes for Output (Reports)

1. Problem: One user suggested that the top margins for the answer key to the performance test and quiz be increased.

Action taken: This margin was increased in the program to approximately 1 inch.

### Processing Errors

1. Problem: The menu choice to print multiple choice questions by software type crashed. The user was returned to a C prompt. This problem was encountered by all seven evaluators.

Action taken: The error was corrected in the program code. An earlier modification had erroneously created the error.

2. Problem: Competency numbers should be displayed in sorted order after adding new ones.

Action taken: Error corrected in the program code for delete and view competency action. No correction on the single form supporting two tables was able to be made.

3. Problem: Five users reported difficulty with an instruction to delete the performance test DPB1. They indicated it did not exist even though subjects were to have created it.

Action taken: Developer checked the procedure for entering the test and deleting it. Modifications in the code were made to correct the problem.

4. Problem: Subject thought the computer got hung up when writing a lesson plan to a file. Lesson plan report seemed to lock up.

Action taken: Code controlling this program was revised and redesigned to speed up processing.

5. Problem: The program needed a warning message when packing the database because users got impatient and did not realize what was going on.

Action taken: A message was added to the screen.

#### Miscellaneous Errors

1. Problem: User was concerned that a key did not print with request to print multiple copies (student copies) of a test generated earlier.

Action taken: The key was printed when the test was first generated. The user would not need multiple keys.

2. Problem: One of the questions had the answer spelled incorrectly.

Action taken: The word was corrected.

3. Problem: When adding matching questions, the subject reported using [Shift/Tab] key combination and moving from question to task number, skipping the answer block.

Action taken: The instruction to edit had been placed between definition and answer. The instruction was moved to a new location on the form to prevent the misunderstanding.

4. Problem: Subject did not understand that word processing was a software type. The user kept thinking of it as a course and tried to complete structured exercise in

the course/competency area. The user encountered the same problem when asked to do an activity involving DOS and fundamentals. The user tried to find it as a "course." The user encountered the same problem with data communications and could not decide whether it was a course or a software type.

Action taken: This particular volunteer was a marketing teacher from North Carolina rather than a business education teacher from Virginia. She was not familiar with the structure of business education courses in Virginia and did not have the opportunity to study the Virginia Business Education curriculum guide.

5. Problem: One user reported an inconsistency with popup menu which uses the "Continue" choice to move to the print menu and at other times "Stop here" moves to the print menu.

Action taken: Comments for popup menus were added to a prompt line at the bottom of the screen.

6. Problem: A user reported pauses during the printing of lesson plan data. A second subject thought the computer had "hung up" while printing the report.

Action taken: The report being generated involved a slow process of "look ups." Code was rewritten and data reorganized to speed up this process.

### Suggestions for New Features

1. One user would like to see test before printing.
2. When choosing "Print to Screen" by mistake, user would like to be given a second chance to print without having to redo everything.
3. The delete function should allow more than one delete at a time.
4. A counter was needed to look at as user selects questions.
5. There should be a way to exit if the user gets into something by mistake.
6. There should be an easier way to put in task numbers with performance tests if you forget.

### Objective 7

The seventh objective was: Conduct final field test with a group of Virginia business education teachers who would individually appraise LessonBank to make certain all menu choices perform correctly.

Fifteen Virginia business education teachers returned the form (Appendix E) in December, 1992, volunteering to evaluate the program for the developer. A revised program and user's manual, structured assignments, check sheet, and evaluation form were sent by priority mail with a stamped priority mail return envelope to the 15 teachers on January 4, 1993. The deadline for returning the materials was

February 8, 1993. Five teachers returned the materials on time; two experienced problems with disks and called the developer. Replacement disks were sent to each. Two of the teachers from the same school did the exercise together and returned the materials a week late because of the difficulty with the disks. Two teachers returned the materials indicating they were too busy at the time to participate. Five teacher's did not respond at all. One of the teachers with disk problems was asked to evaluate a revised version in two weeks. The revised materials (program and manual) were sent to this volunteer on February 16, 1993. They were eventually returned with an explanation that not enough disk space existed on the school's computer to accommodate the program.

Each subject was asked to complete a series of structured exercises measuring every aspect of the program, record any comments about the program, forms, and reports, and timing themselves on each part. Table 4 on page 115 shows the capability of the six volunteers to complete the structured exercises assigned in Appendix D and the length of time it took to complete each part.

The subjects in phase three averaged six hours and 25 minutes evaluating all the components of LessonBank. The large investment of time was required because the purpose of the structured assignment was to make a comprehensive

evaluation and test each component of the entire system. For example, in Part 7, Creating Multiple Choice Tests, the users generated 11 tests in 75 minutes. Under normal conditions a user would not be using each menu choice during a single session. A typical user would use the program to complete one or two activities at a time. For example, to create a lesson plan, generate one test, or print a report should easily take less than 30 minutes.

Table 4

Program completion time and successful completion rate of structured assignment -- Phase 3

Part	Structured Assignment Description	Percent Completed	Min. to Complete Mean	Complete Median
1	Add/Update Task Data	100	22	19
2	Add/Update Competency Data	100	16	11
3	Add/Update Course Data	100	7	6
4	Assign Tasks to Competencies	100	25	23
5	Print Reports	100	25	20
6	Creating Lesson Plans	50	25	20
7	Creating Multiple Choice Tests (11 tests)	83*	75	43
8	Add/Update Multiple Choice Test Questions	83	22	21
9	Creating Matching Tests (10 tests)	83	65	45
10	Add/Update Matching Test Questions	83	17	13
11	Create performance tests	83	38	25
12	Utilities	83	47	40

\*One volunteer experienced a full hard disk and was unable to complete the exercises from Part 7 forward.

Appendix G contains a list of problems, comments and suggestions reported by the volunteers. Subjects 1-5 were

individual instructors and subject 6 consisted of two teachers from the same school who worked together to complete the exercises.

The following pages describe the structured assignment (Appendix D), the desired outcomes, the comments made by the users of phase three, and any action taken.

### Task Banking

The exercises in Parts 1 through 6 were designed to test the [Task, Competency, and Course Data] menu shown below:

Add/update task data  
Add/update competency data  
Add/update course data  
Assign tasks to competencies within courses  
Print reports  
Exit to main menu

The exercises in Part 1 came from the menu choice [Add/update task data] and were designed to test the following menu:

Add task data  
Edit task data  
Delete task data  
Look at task data  
Exit to previous menu

Change the description of a task  
Change the function of a task  
Change the difficulty of a task

Users were asked to add three tasks and were supplied with the software type, function, difficulty, and task description. In addition, they had to change the difficulty of a task in the system, delete a task, and view tasks that had been added or that did exist. All six users were able to complete the exercises. The time required to complete the exercise ranged from 10-45 minutes with 22 minutes as the mean and 19 as the median.

The system was developed so that a form was displayed allowing the user to move the cursor beside the choice of software type, press the [S] key to select one choice, and press the [ALT] key when finished. This activity was repeated to make a selection from both function and difficulty categories.

The comments "repetitive tasks" and "too many steps" from two users is easily explained and understandable. After choosing any one of the seven menu choices (four from the first level and three from the edit level) the user sees the three forms (software type, function, difficulty) in a repeating fashion.

The developers utilized a method where pressing the [S] key displayed a check mark beside the selection. The "check mark method" of selecting the required categories was established for two reasons: (1) data integrity would be preserved by not allowing users to keyboard actual names for

software type, function, difficulty and (2) for ease of use, i.e., users were not required to refer to a secondary source (a manual) to look up either a code for software type, function, and difficulty.

One user remarked that browsing (or editing and deleting) could be sped up by entering the task number itself, if known, and advancing immediately to the desired task. The developers acknowledged that this concept is true. However, this also would require reference to a secondary source for the information and the concept developed was that the program would allow the user to query the database without relying on secondary sources. It would actually be much easier to ask the user to enter the actual task number to edit or delete and this procedure could be changed.

One user commented about getting acquainted with the various function keys and their uses. The program uses the [F8] key or [Enter] to move down a line and the [F7] key to move up one line. The database management system itself establishes these keystrokes. This version of R:Base does not allow one to use the up arrow and down arrow keys to move through the tables of data.

One teacher commented about the order of the tasks. Each task has a unique number that serves as a key to the table and is associated with a software type, function, and

difficulty, and does not lend itself to any type of meaningful ascending or descending order arrangement. Therefore, the developers allowed the users to browse the entire list of tasks in an unsorted fashion.

Another reason for the three selections of software type, function, and difficulty throughout the program is to narrow the listing of 1,100 tasks to a smaller group which match the criteria selected. From this grouping it is possible to select a task to edit, delete, or view.

One user indicated the instructions were easy to follow and another user desired a clearer explanation of the project itself and the desired outcomes.

#### Competency Banking

One component of the system is a collection of a structured set of 140 computer software competencies related to business education courses. These competencies were extracted from the Business Education Suggested Course Competencies and Performance Objectives published by the Virginia Department of Education in 1989.

The exercises in Part 2 came from the menu choice [Add/update competency data] and were designed to test the menus on the next page:

```
graph TD; A["Add competency data  
Edit competency data  
Delete competency data  
Print competency data by course  
View competency data  
Exit to previous menu"] --- B["Change the description of a competency  
Change the number associated with a competency  
Exit to previous menu"]
```

Add competency data  
Edit competency data  
Delete competency data  
Print competency data by course  
View competency data  
Exit to previous menu

Change the description of a competency  
Change the number associated with a competency  
Exit to previous menu

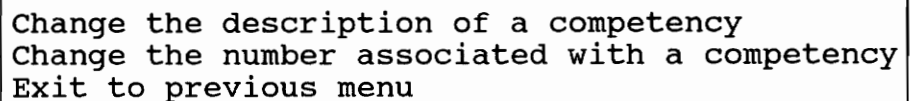
Users were asked to add a competency to a course, edit the description, change the number, print the competency information for a specific course, view the competency data and delete an existing competency. All six users were able to successfully complete the required exercises. The time required to complete the exercise ranged from 8-40 minutes with 16 minutes as the mean and 11 minutes as the median.

The system was developed so that a form was displayed allowing the user to move the cursor beside the choice of specific course, press the [S] key to select one, and press the [ALT] key when finished. Selecting the appropriate course either assigned a competency to that course or displayed the existing competencies for the course.

In this section two users indicated no problems, one user overlooked an instruction on the screen and had to repeat the exercise. One user commented "too much going back and forth," which probably refers to the exercise assignment where users added a competency, changed the

wording of that same competency, changed the number of that competency, viewed the competencies to determine if changes had been recorded correctly, deleted the same competency and then viewed the competency data again to verify that it had been deleted.

One user commented that "in editing, option to enter competency number would be helpful." From the [Edit competency data] menu choice users are allowed to change either the description or the number of an existing competency as shown below:



Change the description of a competency
Change the number associated with a competency
Exit to previous menu

The reason for separating these two editing options is that the description is contained only in the competency table and the competency number must be updated in two tables. These changes must be carefully controlled to protect the integrity of the data.

The view competency screen failed to indicate that the [F8] and [F7] keys could be used to scroll through the data. An explanation was added to the screen form. One user commented on "poor directions" in step 1 of the exercise which did not directly relate to the improvement of the program.

### Course Banking

The exercises in Part 3 came from the menu choice [Add/update course data] and were designed to test the following menus:

Add new courses Edit course information Delete a course Print a course listing Exit to previous menu
--

Users were asked to add a course, edit the name of the course, print a list of courses and delete the course added in the first exercise. The time required to complete the exercise ranged from 2-15 minutes with 7 minutes as the mean and 6 minutes as the median.

Three users had no comments or problems with this assignment. One commented on a slight change in the editing procedure which allowed the user to simply move to the course name on the screen and make required changes. At one end of the spectrum a user commented "total confusion" but did not explain what was meant and on the other end one user commented "instructions were easy to follow."

### Assigning Tasks to Competencies

The exercises in Part 4 came from the menu choice [Assign tasks to competencies within courses] and were designed to test the menu on the following page:

Assign tasks to competencies within courses Unassign tasks Exit to previous menu
--

Users were asked to assign tasks to a competency for a specific course, print the list of competencies and tasks for the named course, delete the task assignment made and print the listing again to verify that it had been removed or unassigned. All users were able to complete the exercise successfully. The time required to complete the exercise ranged from 10-45 minutes with 25 minutes as the mean and 23 minutes as the median.

The "assign" and "unassign" tasks portion of this exercise displayed the list of courses allowing the user to move the cursor beside the name of a specific course, press the [S] key to select one, and press the [ALT] key when finished. Next a form displaying the competencies appeared on the screen. The user was to press the [S] key to select a competency to assign tasks and press [ALT] when finished. Since the competency is associated with one software type, users select a software type to narrow the list of tasks to be displayed. A list of all possible tasks by software type is displayed. The user was prompted to use the "check mark" (press [S] key) to select as many tasks as desired to be assigned to the competency chosen. Unassign follows the same procedure.

The assign and unassign activities in this part elicited only one comment. The user commented that "deleting is not unassigning (using space bar)" which referred to the display on the screen which stated "You are deleting tasks for: . . ." and the standard directions "Use [S] to select as many tasks as you desire; [space bar] to unmark". The screen display was changed to read "You are unassigning tasks for:. . ." No change was made to other instructions as the user's manual instructs users to place a mark beside each task to unassign and press [Alt] when finished.

The second part of this exercise required users to access the [Print reports] menu. Problems expressed by users in this phase were encountered by users in both phase one and two of this project. However, the developers thought that if the users would pay closer attention to the instructions in the user's manual the problem would correct itself. Since misunderstanding continued, a revision to the process of (a) selecting software type, function, difficulty and (b) selecting course and competency was obviously required. These changes have been made and tested fully by the developers and a student volunteer.

The problems occurred because the developers had designed a form combining three forms into one for selecting software type, function, difficulty instead of using the

standard procedure of displaying separate forms for each choice and another form combining the two course and competency forms into one. The combination form displayed only three lines of data from all three tables and was crowded (because of the 24 line limit). All explanations for its use had to be confined to the user's manual. Users prefer following on screen directions rather than consulting a secondary source if at all possible.

The problem was compounded because of a popup menu which asked for a response to "continue, stop here, redo, or exit" following each choice. Users were unfamiliar with this routine and found it very confusing.

The second form that had caused difficulties was a dual table form (course/competency) which required that users keep the cursor directly on the course name before pressing [Shift/F8] (a key combination established by the database management system) to access the competency table. This instruction was on the form but could be easily missed or violated. Users were instructed to mark as many competencies as desired and press [Enter]. If they failed to press the [Enter] key the last competency would not be selected.

Comments from users in phase three such as "could not get to function; kept highlighting stop instead of continue; did not realize the list in function was longer; very

confusing" made reference to the problems described above (See Appendix G for a complete listing of comments).

### Printing Reports

The exercises in Part 5 came from the menu choice [Print reports] and were designed to test the following menu:

List of tasks by software type, function, or difficulty
List of competencies by course
List of competencies and tasks by course
Exit to previous menu

Users were asked to print various reports based on variables of software type/function/difficulty and course/competency. All users were successful in completing the exercise. The time required to complete the exercise ranged from 10-50 minutes with 25 minutes as the mean and 20 minutes as the median.

Users in Part 5 were beginning to understand the multi-table forms described in the paragraphs above since they had experienced problems in Part 4 when printing reports. However, it was the opinion of this developer that the procedure should be simplified to follow the earlier pattern of separate forms and eliminate the opportunity to redo the selections which did require several additional steps and keystrokes. One user commented "too many steps to print, I could key in commands faster." The developer acknowledged

that entering data manually is an option which was considered but decided against because it would require reference to a secondary source for the information and the program was established so users could query the database without relying on secondary sources.

A listing of comments for Part 5, Print Reports, can be found in Appendix G.

#### Creating Lesson Plans

The exercises in Part 6 came from the main menu choice [Lesson Plans] and were designed to test the following menu:

Create a new lesson plan Reprint previous lesson plan Exit to main menu
---

Users were asked to select a course, select the competencies on which to base the lesson plan, print a copy to the printer, reprint a copy to a file, exit from LessonBank, retrieve the lesson plan in a word processor, and print a copy from the word processor. Four users were able to print the lesson plan. However, only one user could retrieve a copy in a word processor. The time required to complete the exercise ranged from 10-50 minutes with 26 minutes as the mean and 20 minutes as the median.

One user reported no problem with the exercise and indicated the instructions were easy to follow. However,

other users experienced difficulties with the group of instructions.

The steps for this exercise asked the user to select a course (any course) and a competency (any competency) to develop a lesson plan. The developer realized that this open ended instruction was a mistake. Apparently course and competencies were chosen for which there were no tasks assigned. One user sent a print screen of her failure to create the desired lesson plan. After further investigation, it was discovered that the competency she had chosen did not have any tasks assigned to it. Tasks have now been assigned to all competencies.

Three users could not retrieve the document in a word processor. The developer attempted this exercise again and found that when using Word Perfect 5.1 and retrieving the document with [Shift/F10] a message came on the screen indicating "document conversion in progress" and then appeared to "hang up." After waiting a short period of time and pressing Enter, the document did appear on the screen.

The exercise was rewritten and modified versions of the program were sent to all six users on February 23, 1993. Each was asked to load the revised program and complete the activity with specific instructions and return the results to the developer. The first user to return the new assignment had been one of the subjects who had not been

able to retrieve the report in the word processor. After becoming more explicit with instructions, she was able to print the report on a word processor without any problem.

### Multiple Choice Question Banking

The exercises in Part 7 came from the main menu choice [Multiple Choice Tests] and the submenu shown below. The exercises in this part were to test the [Create a multiple choice test] selection.

Create a multiple choice test  
Add new multiple choice questions  
Print questions from testbank by software type  
Edit or delete existing multiple choice questions  
Exit to main menu

Users were asked to print 10 multiple choice tests in the five configurations which follow.

By course and competency  
By software type  
By software type, function  
By software type, difficulty  
By software type, function, difficulty

One set of tests was to be generated at random by the program and the other set of test questions was to be chosen by the user. All but one user successfully completed this exercise. The time required to complete the exercise ranged from 25-180 minutes with 75 minutes as the mean and 43 minutes as the median.

The comments in this exercise ranged from no problems to extremely frustrating. The multiple choice tests use the troublesome multiple table forms described in Part 4 above. Two teachers had difficulty with the database during this assignment and had to start the software again. The developers think that this problem occurred because of difficulties with the multiple table forms. Criticism concerning use of those forms continued. The program was modified and single table forms have been incorporated in the application.

A suggestion was received from three teachers that the program should dynamically indicate the number of questions selected. The developers were not able to accommodate this request through additional programming, but did modify the form with the prompt "Press [Alt] to verify the number of questions selected." When the user presses the [Alt] key a popup menu is displayed at the top of the screen:

Continue--Prepare test now Redo--Add or subtract questions Stop--Cancel and return to previous menu Question count = 8
---

The user may then accept the number shown by pressing C to Continue and prepare the test, reject the number by pressing R to add more questions or to remove some chosen, or select S to exit from the [Create multiple choice] menu option.

The exercises in Part 8 came from the main menu choice [Multiple Choice Tests] and the submenu shown below. The exercises in this part were to test the [Add new multiple choice questions], [Print questions from testbank by software type], [Edit or delete existing multiple choice questions] selections.

Create a multiple choice test Add new multiple choice questions Print questions from testbank by software type Edit or delete existing multiple choice questions Exit to main menu
--

Users were asked to add five multiple choice questions, delete one of them, edit the answer to one of the questions added, and print all the multiple choice questions by a specific software type. Five of the six users were able to complete this activity. The time required to complete the exercise ranged from 10-35 minutes with 22 minutes as the mean and 21 minutes as the median.

Users had a variety of comments with this activity. Only user No. 2 seemed to have extreme difficulty with this set of exercises. Fortunately, this user did print a directory listing and returned it with her exercises. The printout indicated there were no bytes free on her computer. The developer believes this explains the problems this particular teacher had with completing any of the activities from this point forward. A warning notice was added to the

user's manual cautioning users about attempting to work with a full disk.

The exercises were returned with each packet and in all cases except the one noted above, users were able to add, edit and delete the questions and print all the questions for a software type satisfactorily. Two comments focusing on editing and deleting prompted the developer to create edit/delete forms for multiple choice and matching questions which clarified and expanded instructions to both edit and delete. The original routine, which used the add question form, was not adequate for editing and deleting.

One user commented "very unforgiving of mistakes" and followed the comment with "cursor movement to task number is unexpected and at this point causes many problems." The problem can be explained by understanding how this form works: To add questions, users must know the task number that the question measures. Users key the stem, press [Enter]; key answer a, press [Enter]; key answer b, press [Enter]; key answer c, press [Enter]; key answer d, press [Enter]; key the correct answer, and the program immediately moves to the area for task number. The correct answer column is only one character in length and when it is filled the user does not press [Enter]. Pressing [Enter] at this point brings a popup menu (Add row, Discard Row, Add Row and exit) which causes confusion to the user. To assist the

user a comment was added to the form beside the task number space "Press [Alt] if you missed this!" This may assist users who encounter this problem.

#### Matching Question Banking

The exercises in Part 9 came from the main menu choice [Matching Tests] and the submenu shown below. The exercises in this part were to test the [Create a matching test] selection.

Create a matching test  
Add new matching questions  
Print questions from testbank by software type  
Edit or delete existing matching questions  
Exit to main menu

Users were asked to print 10 matching tests in the five configurations which follow.

By course and competency  
By software type  
By software type, function  
By software type, difficulty  
By software type, function, difficulty

One set of tests was to be generated at random by the program and the other set of test questions were to be chosen by the user. Five of the six subjects completed the exercises successfully. One subject experienced difficulty with the program due to a full hard disk on the computer. The time required to complete the exercises ranged from 7-

180 minutes with 65 minutes as the mean and 45 minutes as the median.

Generating matching tests requires several manipulations of the data and the creation of temporary tables to number the questions consecutively, sort the answers alphabetically, and, in the case of random selection, create a table of random numbers which can be matched with questions meeting the criteria chosen. The programming code is fairly complicated and because temporary tables must first be deleted and then rebuilt it appears that the user has to wait a long time for this entire process to take place. Those using slower model computers will notice that this activity takes considerable time.

One user noticed an error in the messages appearing on the screen. While data is being processed messages frequently appear such as "Please wait, R:Base is removing old data" or "R:Base is busy working for you. . ." Another message states "Please wait, R:Base is randomly picking the number of questions you requested." This latter message erroneously appeared after the user had selected the questions. The error was corrected in the programming code.

The exercises in Part 10 came from the main menu choice [Matching Tests] and the submenu shown below. The exercises in this part were to test the [Add new matching questions],

[Print questions from testbank by software type], [Edit or delete existing matching questions] selections.

Create a matching test  
Add new matching questions  
Print questions from testbank by software type  
Edit or delete existing matching questions  
Exit to main menu

Users were asked to add six matching questions, delete one of them, edit the answer to one of the questions added, and print all the matching questions by a specific software type. Five of the six subjects completed the exercises successfully. One subject experienced difficulty with the program due to a full hard disk on the computer. The time required to complete the exercise ranged from 5-35 minutes with 18 minutes as the mean and 13 minutes as the median.

#### Performance Test Banking

The exercises in Part 11 came from the main menu choice [Performance Tests] and the submenu shown below. The exercises in this part were designed to test the following menu choices:

Add performance test instructions, assign tasks,  
create quiz  
Edit instructions, task assignments, quiz questions  
Delete performance test  
Print list of performance tests  
Print performance test  
Print quiz based on performance test  
Print tasks measured by performance test  
Exit to main menu

Users were asked to enter a new performance test, enter tasks measured by the performance test, enter a quiz, edit the new performance test, delete a test, print a performance test, quiz, answer key, and a report of tasks measured by a specific test. Five of the six subjects completed the exercises successfully. One subject experienced difficulty with the program due to a full hard disk on the computer. The time required to complete the exercise ranged from 15-90 minutes with 38 minutes as the mean and 25 minutes as the median.

The developer asked the teachers to delete a performance test and then erroneously requested they print the test and quizzes. The majority substituted another test for this. One user commented, "How to retrieve and print a deleted file?" One substantive comment came from a user who stated that the red block on the quiz form was hard on the eyes. This color was changed.

### Utilities

The exercises in Part 12 came from the main menu choice [Utilities] and the submenu shown below. The exercises in this part were designed to test the menu choices shown on the next page.

Backup database (*.RBF) only to floppy disks Restore database (*.RBF) files from floppy disks Show directory for any drive Check the disk situation on any drive Pack this database. Caution: Make a backup before packing DOS backup ALL <u>LessonBank</u> files to floppy disks Format a floppy disk Exit to main menu
---

Users were asked to complete activities to measure all seven menu choices. Five of the six users were successful with all exercises but two. The following problems were noted and corrected: (1) Users were unable to make a backup to drive B. (2) The restore routine needed an explanation that two megabytes of free disk space was required because the original database was renamed before restoring the backup copy was completed. This statement was added to the user's manual. Minor changes were made in the order of items on the [Utilities] menu. The time required to complete the exercise ranged from 10-120 minutes with 47 minutes as the mean and 40 minutes as the median.

#### Overall Evaluation of LessonBank

The users of LessonBank in phase three completed the program evaluation by rating 19 categories as shown in Table 5. They completed a questionnaire as shown in Appendix F and marked the various categories as excellent, good, fair, poor, or very poor. The forms were tallied using the following weights: excellent, 10 points; good, 7; fair, 4;

poor, 2; and very poor, 0. A weighted average was calculated and is reported in Table 5 on page 136.

The weighted average on all categories was 6.57 for a rating of good. The developer used the following weighted table for evaluation purposes:

<u>Range</u>	<u>Rating</u>
8.00-10.00	Excellent
5.00-7.00	Good
3.00-4.00	Fair
1.00-2.00	Poor
0.00-0.99	Very poor

Table 5

Overall evaluation of LessonBank

<u>Category</u>	<u>Weighted Average</u>	<u>Rating</u>
<u>Program Format</u>		
Consistent use of terminology	8.50	Excellent
Clear and concise instructions	5.00	Good
Well organized and sequential	7.67	Good
<u>Output format</u>		
Screen and printer or other peripherals used	8.00	Excellent
Report generation is logical and readable	8.50	Excellent
Screen displays are consistent and easy to use	5.00	Good
<u>Input and operation</u>		
Procedure for entering data is consistent	7.00	Good
Prompts are plentiful and clear	5.33	Good
System commands are logical	4.00	Fair
Error trapping is sufficient	6.00	Good
Error messages are understandable	6.17	Good
Restart and recovery is easy and allowed	4.67	Fair
Sufficient data field length	9.00	Excellent
Number of records is adequate	8.00	Excellent
Speed to access records is adequate	5.50	Good
Maintenance of file is acceptable	7.50	Good
<u>Installation and Backup</u>		
Ease of installation	7.67	Good
Overall ease of operation	5.50	Good
Ease of backup	5.83	Good
Overall average:	6.57	Good

Of the three categories under program format the program received one excellent and two good rankings. The three categories ranked under output format received two excellents and one good ranking. Ten categories were included under input and operation. These categories included two excellent ratings, six good ratings and two fair. All three categories listed under installation and backup received good rankings.

The developer believes the fair ratings were based on the confusing multi-table forms referred to earlier. These forms have subsequently been changed and the procedure simplified for input and operation.

At the time the developers created the multi-table forms in question, there was a doubt as to whether this procedure would work smoothly without being able to train a user. Since LessonBank had several components it was simple enough to use two approaches to selecting (a) software type, function and difficulty and (b) course and competency. Even though the developers personally preferred the multi-table approach, the users preferred a simpler procedure.

#### Summary

The formative evaluation process uncovered a number of problems and errors. These problems were all corrected. The suggestions from users were incorporated in the program and user's manual where feasible and strengthened the

program. Users should understand that all computer programs have room for improvement. This is verified by software companies as frequent upgrades and revisions are developed. The formative evaluation process does not insure a perfect program, but is a successful vehicle for removing many problems. The author believes that LessonBank has been stabilized through the formative evaluation process.

## CHAPTER 6

### SUMMARY AND RECOMMENDATIONS

Chapter 6 contains background information, a summary and discussion of the development process, recommendations for developers, recommendations for teachers, supervisors, and administrators, potential uses of the system, and suggestions for getting teachers to use the system.

#### Background

The purposes of this project were to (a) evaluate and revise a computer-based instructional management system developed to organize business computer competencies, and (b) develop and revise documentation for using the system.

The instructional management system consists of a database and a database application employing relational database architecture. The resulting system can be used by Virginia business teachers in implementing their curricula.

The prototype system was developed initially to organize a taxonomy of tasks identified to measure computer competencies. The computer competencies were extracted from the Business Education Suggested Course Competencies and Performance Objectives, published by the Virginia Department of Education in 1989. The taxonomy resulted in the publication of the Business Computer Software Curriculum Series in 1990. This latter publication forms the core of the instructional management system. The instructional

management system was ultimately expanded to include multiple choice and matching test questions organized to measure the competencies. A module of performance tests was incorporated into the system as well.

The resulting program was titled LessonBank: The Instructional Management System (Virginia Department of Education, 1993). LessonBank is a menu-driven program designed to be used as a tool to assist teachers with the management function of teaching. The system incorporates a number of components. The components are (a) course banking, (b) competency banking, (c) task banking, (d) test question banking, (e) performance test banking, (f) test generation, (g) lesson plan generation, (h), reporting, and (i) database maintenance.

A bank is defined as a system that allows for data collection that is both structured and formatted. Structured data are data organized into spreadsheet or database tables (Kroenke & Dolan, 1990). Each bank in the program has the capability of having data added, revised, deleted, selected, sorted, and printed.

LessonBank is organized around relational database architecture which stores not only data, but also relationships among data. With the relational model, the user need only specify which records to process (for example

which course and competency). The DBMS navigates through the database manipulating the data for the user.

Relational database systems require more computer resources, and as a result they were much slower than systems based on earlier database models (Kroenke, 1992). Early versions of relational products had a slow response time that was unacceptable to users. As faster computers were developed and prices dropped, relational database products became more attractive. The relational model has only recently gained true popularity with microcomputer users.

#### Project Summary and Discussion

During phase one of the evaluation process, the developer used a one-to-one evaluation with one individual user to obtain an initial reaction to all components of the system. The developer met with the user for an orientation session in which the purpose of the instructional management program and the one-to-one testing was explained. A user's manual was developed for evaluation during phase one. The developer and user met, loaded the program in the microcomputer, and the developer explained how the program was to operate. The developer then observed as the subject used the program. The developer interacted with the user following each major choice from the main menu and made notes concerning every suggestion for improvement. No

coaching was allowed. The user was given an assignment for each menu choice designed to test each component of the program. The developer observed by recording places where errors were made, questions were asked, and instructions were not clear. Notes were made of all comments and suggestions as well as alternative explanations made by the developer. The purpose of this first stage of formative evaluation was to identify and eliminate the most obvious errors and obtain initial reactions to the program (Dick & Carey, 1985). Revisions were made to the program based on the suggestions of the initial evaluator and the user's manual was expanded from 12 to 63 pages.

Phase two of the formative evaluation process contained seven subjects who were as nearly representative of the target population as possible. The subjects were members of a graduate class in instructional methods in cooperative marketing and office education programs at East Carolina University. Subjects loaded the program and completed the structured assignments on their own. The subjects were given an assignment for each menu choice designed to test each component of the program. The purpose of phase two evaluation was to determine the effectiveness of changes made following the one-to-one evaluation and to identify any remaining problems. A secondary purpose was to determine if subjects could use the program without interaction with the

developer. The subjects comments and structured assignments were analyzed to discover problems. Revisions were made to the program following this cycle.

Phase three, the field trial, involved distribution of the revised LessonBank program to 15 Virginia business teachers. The developer contacted a vocational business education specialist in the Virginia Department of Education who provided the names and addresses of 25 local business education supervisors and vocational administrators. The supervisors were asked to provide the names of two teachers who might be interested in participating in the study. Fifteen supervisors responded to the request. Thirty teachers were contacted and from that number 15 volunteered to participate in the project. The developer prepared 15 copies of the program, user's manual, structured assignment sheet, check sheet, and evaluation form. Seven teachers field tested the entire program and returned the materials.

Following the responses from the field test, the developer acknowledged a difficulty existed with two rather cumbersome screen forms which had been receiving negative responses since phase one. The use of the forms was not intuitive and even with expanded explanation in the user's manual the complaints persisted. These forms were subsequently simplified to follow an exact procedure used in another part of the program. A student of the developer

once again tested each menu choice to make certain that the program worked completely as intended. No additional problems were encountered.

The project summary and discussion for developers is organized around the seven objectives of the project.

1. Developed written documentation for using LessonBank in the form of a user's manual. Users of microcomputer systems are generally not information systems professionals, and they prefer user-friendly products. The developers of LessonBank therefore created a menu-driven program which did not require extensive training time. The system has instructions on each screen and a set of menus to guide the user through the processes of adding, editing, deleting, and printing the data.

A 12-page manual describing the capability of LessonBank, its six main menu choices, system requirements, size and type of computer required, how to install and start the program, and an overview of the keys to be used was developed. The manual was used as part of the evaluation discussed in objective 2.

System users require documentation for system installation, entering and editing data, correcting errors, running the system, and interpreting the output. Documentation is one of the five major components of a computer information system. The other four components of

the system model are hardware, programs, data, and people (Kroenke & Dolan, 1990). Procedures (documentation) are written instructions for users of a system and must be developed and tested during the program development stage.

2. Conducted a one-to-one evaluation with an individual to obtain an initial reaction to all components of LessonBank and the user's manual. Phase one of the program evaluation used an experienced high school teacher from the target population. This user loaded the program and completed structured assignments. The entire process took 10-12 hours as each menu choice had to be tested. The developer observed and made note of any difficulties encountered. Errors and problems were recorded and later grouped into suggested changes for data input (screen forms), user's manual, data output (reports), processing errors, and suggested new features. The user encountered some difficulties with interpretation of the instructions contained in the structured assignment.

The one-to-one or clinical evaluation used in this project identified problems in the program. Direct interaction between the developer and a user helped to identify and remove the most obvious errors in the program and to obtain an initial reaction to the components of the computer information (instructional management) system.

3. Revised the user's manual into a more comprehensive document following interaction with the initial user. The menu and screen interface worked for the first subject; however, she expressed a desire for a more extensive user's manual to help with program and application interaction. With the aid of a snapshot utility provided by the database management system, the developer was able to record the screen output of all menus, forms, and reports for inclusion in a detailed user's manual. The output was printed on a laser printer and reduced on the copying machine. The manual grew from 12 pages to 63 pages, providing a comprehensive description of each menu choice.

A detailed user's manual with full instructions and illustrations for each component of LessonBank is required to use the program successfully without assistance from the developer. Documentation should be evaluated by concerned personnel and should be comprehensive enough to provide complete instructions on using and maintaining the system.

4. Revised LessonBank following an analysis of the one-to-one evaluation. The developer made several modifications to the screen forms clarifying instructions and improving the interface between the user and the database. Programming problems were corrected and changes to reports and menus were made as suggested by the first subject to test the system.

The formative evaluation model used in this project was effective in identifying problems with the program. The developer analyzed the recommendations of the user and incorporated these suggestions where possible. A user in this interactive process is able to describe difficulties with the sequence of the program and the interface of the data input screens as well as typographical errors and inconsistent or incorrect instructions.

Formative evaluation is an appropriate method to use in the revision process of a computer based instructional management system. This conclusion is supported in studies by Keatley (1987), Sherron (1984), and George (1992) who identified formative evaluation as an appropriate method for discovering and amending problems in instructional programs.

5. Conducted a field test with a small number of individual users to check effectiveness of changes made following the revision process and identified remaining problems. This field test was conducted with graduate students in an instructional methods class at East Carolina University. These users made additional suggestions for improving the screen forms and reports. They were the first to use the expanded user's manual. They installed the program and used it without assistance from the developer to complete structured exercises. The users wrote comments and suggestions directly on the structured assignment form and

in the manual. This field test resulted in improved explanations on the use of a multi-table form which caused problems for users in both phase one and two. Minor changes were suggested for instructions on preparing reports. Additional processing errors were corrected and notes were made of suggestions for new features. The developer determined from feedback that some of the instructions on the structured assignments were not clear.

Evaluation with a second group of users continued to identify suggestions for program improvement and programming problems. This phase of development was successful in indicating that users were able to use the system without interaction with the developer. The individual evaluation process is an important step in improvement of the instructional management program. Input from several individual users will help focus on the main problems with the system.

6. Made further revisions to the program and user's manual following responses from individual evaluations in the field test. The revision process was continued as suggested changes were incorporated in the program and manual. The developer made programming changes during this phase to speed up the processing of data within the program. Some users thought the program had locked up during the generation of some of the reports. Processing speed was

affected favorably when appropriate keys for indexing were defined in various tables. A warning message was added to the program as well asking users to be patient as the data was being processed.

The instructional management system was further improved following feedback from individual users in phase two. Subjects reactions were analyzed and changes incorporated into the program where feasible. The developer received several suggestions for improvement to screen instructions.

Comments received from subjects indicated that the users were encountering some difficulties with the instructions themselves in the twelve different parts of the structured assignments. An evaluator should make certain that all instructions are clear to each subject before making an assignment.

7. Conducted final field test with a group of Virginia business education teachers to individually appraise LessonBank and make certain all menu choices perform correctly. Fifteen Virginia business education teachers volunteered to evaluate the program. The computer program, user's manual, structured assignments, and evaluation form were sent to the teachers. Seven subjects (two teachers worked together) completed all exercises and recorded comments, problems, and suggestions on the evaluation form.

Returned assignments and forms were analyzed by the developer to determine if the exercises had been completed correctly. In all but one of the 12 parts contained in the exercises, subjects were able to complete the activities. The only major difficulty experienced by the subjects was creating a lesson plan. Because the developer had not structured the lesson plan assignment the same as the others, the exercise was rewritten and mailed to the seven subjects for further input. Four of the seven users returned the new assignment successfully completed. One user sent a correct screen print of the document from her word processor and printed copy of the document containing garbage which indicated that an incorrect printer had been selected. Users continued to encounter minor difficulties with the structured assignments.

The final stage of the formative evaluation model indicated that the subjects could use the program in the environment for which it was intended. LessonBank was designed to assist business educators plan instruction on the sizable number of computer operations required for occupations in business.

Subjects in the final phase continued to express some of the same complaints users in phase two had revealed. The developer had attempted to correct these problems by explaining in greater detail in the user's manual how to use

two complex screen forms rather than changing the forms themselves. The developer should continue to work to acquire an alternative method of program data input design when users experience problems with an existing interface.

#### Recommendations for Developers

The following recommendations are made for developers who might use the formative evaluation procedure to evaluate and revise a comprehensive computer program such as LessonBank.

1. Develop a user's manual as the program is being written to document the procedures for using the system. The development of documentation should be considered as important as program development. Problems and difficulties could be identified early in the process.

2. Because users in each phase appeared to struggle with some of the directions provided with the assignments, completion of a preliminary evaluation of the assignments themselves by one or two persons to clarify instructions would be desirable.

3. Describe fully the purpose of any procedures assigned in the comprehensive structured assignments.

4. Include a check off method with each activity to indicate success or failure for each part.

5. Consider showing actual screen displays in the structured assignments or reference to a specific page in the user's manual.

6. Use a computer that offers multitasking capability so the development and revision process can be expedited.

7. Use a modular approach to software design so that program changes can be incorporated easily.

#### Instructional and Research Recommendations

The following recommendations are made for teachers, supervisors, and policymakers who have responsibility for improvement and management of instruction.

1. Teachers should strengthen the program by adding questions to the matching and multiple choice modules as well as the performance test bank. Geisert & Futrell (1990) note that five to ten tests items should exist for each competency.

2. Teachers should use the instructional system in the setting for which it was intended for a period of time and a summative evaluation should be conducted by an external evaluator to certify the program utility (Worthen & Saunders, 1987).

3. Policymakers should consider the life cycle of any system. Once the project is concluded and teachers begin to use the system, consideration should be given to who will maintain and update it as needs change.

## Potential Uses

1. LessonBank can be used by teachers to create lesson plans based on a chosen competency. The pre-assignment of tasks to existing competencies presents the teacher with a report which indicates the tasks needed to be completed in order to achieve the competency. This report can be used as a checklist for planning purposes, presented to a parent interested in the activities that are taking place in the classroom, or delivered to the supervisor or principal who may have a need to know what is taking place in the classroom.

2. The system allows teachers to evaluate the pre-assignment of tasks to competencies and make any changes desired. LessonBank is an interactive system which can be easily changed to meet the needs of a particular school or department.

3. The system can be used by teachers to add, modify, or delete competencies as the demands in course content change.

4. Users are able to use the system to add, modify, or delete course information.

5. LessonBank can be used to add, modify, or delete task information as needs change.

6. LessonBank can be used by teachers to generate a listing of tasks to be accomplished based on a particular

software type. The teacher may chose one or more of the nine functions on which to develop a listing of tasks. The task listing may be used in consultation with students, parents, or supervisors.

7. The system can be used by teachers to generate a listing of tasks for a particular software type and a difficulty level as well. For example, a listing of all beginning word processing tasks may be easily generated. Perhaps the teacher needs a listing of advanced database tasks. These reports can be created from existing data by making appropriate choices on the menus and forms provided. The resulting information may be used as the basis for conducting an in-service workshop for teachers or other persons.

8. The test banking components of the system allow the teachers the opportunity to create parallel versions of tests. The test generating capability of a CMI system allows instructors to create nearly unlimited testing and retesting opportunities for students.

#### Suggestions for Getting Teachers to Use the System

1. Teachers could be provided with inservice training programs to introduce them to the system.

2. Teachers could be provided with training opportunities by satellite, microwave, or other distance learning technologies.

3. Workshops and short courses could be offered for teachers during the summer.

4. Exercises or lessons could be distributed with a copy of the program to demonstrate its capabilities.

## REFERENCES

- Baker, E. L. (1974). Formative evaluation of instruction. In W. J. Popham (Ed.), Evaluation in education (pp. 533-573). Berkeley, CA: McCutchan Publishing Corporation.
- Baker, F. B. (1978). Computer managed instruction: Theory and practice. Englewood Cliffs, New Jersey: Educational Technology Publications.
- Bank, A., & Craig, E. (1987). Some lessons for educators from management information systems literature. In A. Bank & R. C. Williams (Eds.), Information systems and school improvement: Inventing the future (pp. 22-38). New York, NY: Teacher's College Press.
- Bank, A., & Williams, R. C. (1987). The coming of instructional information systems. In A. Bank & R. C. Williams (Eds.), Information systems and school improvement: Inventing the future (pp. 3-10). New York, NY: Teacher's College Press.
- Coburn, P., Kelman, P., Roberts, N., Snyder, T. F., Watt, D. H. & Weiner, C. (1985). Practical guide to computers in education (second edition). Reading, Massachusetts: Addison-Wesley Publishing Co.
- Crist-Whitzel, J. L., Terry, P. D., Edelstein, R., Rowan, B. (1986). Patterns of implementing a district computerized instructional management system. San Francisco, CA: Far West Laboratory for Educational Research and Development. (ERIC Document Reproduction Service No. ED 277 367)
- Dick, W. (1977). Formative evaluation. In L. J. Briggs (Ed.), Instructional design: Principles and applications. Englewood Cliffs, NJ: Educational Technology publications.
- Dick, W. & Carey, L. (1985). The systematic design of instruction. Glenview, Illinois: Scott, Foresman and Company.
- Duby, A. (1987). Self-formative evaluation of instructional materials. Educational Technology, 27(2), 48-50.

- Gagne, R. M. & Briggs, L. J. (1974). Principles of instructional design. New York, New York: Holt, Rinehart, and Winston, Inc.
- Geisert, P. G. & Futrell, M. K. (1990). Teachers, computers, and curriculum: Microcomputers in the classroom. Boston: Allyn and Bacon.
- George, C. A. (1992). The application of formative evaluation in the development of a database program for school district administrators. (Doctoral dissertation, University of Pittsburgh, 1992). Dissertation Abstracts International, 53(05), 1340-A.
- Golas, K. C. (1983). The formative evaluation of computer-assisted instruction. Educational Technology, 23(1), 26-28.
- Gutherie, H. (1987). Computer managed learning--A monograph. TAFE National Centre for Research and Development. Australia, Melbourne: Nelson Wadsworth.
- Jorczak, R. L. & Roberts, F. C. (1987). Test design for computer managed instruction systems. Proceedings for the Association for the Development of Computer-based Instructional Systems. (pp. 135-139)
- Keatley, M. (1987). Development of a computer-assisted instructional program to teach word processing terminology. (Doctoral Dissertation, Virginia Polytechnic Institute & State University, 1987). Dissertation Abstracts International, 48(5), 1096-A.
- Kohnken, J. D. (1987). Data base management systems for microcomputers: potential administrative applications in schools. (Doctoral Dissertation, Hofstra University, 1987). Dissertation Abstracts International, 49(4), 681-A.
- Kroenke, D. M. (1992). Database Processing (4th ed.). New York, NY: Macmillan Publishing Co.
- Kroenke, D. M. & Dolan, K. A. (1990). History of information systems. Business Computer Systems. New York, NY: Mitchell McGraw-Hill.
- Kroenke, D. M. & Nilson, D. E. (1986). Database Processing for Microcomputers. Chicago: Science Research Associates, Inc.

- Lancaster, David. (1985). Management and planning issues in the use of microcomputers in schools. Bangkok, Unesco.
- Lundgren, T. D. "Computerized testing: An innovative teaching tool", Instructional Strategies: An applied research series Delta Pi Epsilon, 7, (Winter, 1991).
- Marcom, J., Jr., & Bellew, P. A. (1985, April 17). Slow response. The Wall Street Journal, pp. 1, 23.
- Martin, C. D. (1991). New findings from qualitative data using hypermedia: Microcomputers, control and equity. Computers in Education, 16(3), 219-227.
- McKinnon, K. W. (1986). Planning and using a computerized instructional management system. AASA Convention.
- Nevo, D. (1986). Conceptualization of educational evaluation. In E. R. House (Ed.) New directions in educational evaluation. London: Falmer Press.
- Office of Technology Assessment (1988). Power on! New tools for teaching and learning. OTA-SET-379. Washington, DC: U.S. Government Printing Office.
- Patterson, A. C. & Bloch, B. (1987). Formative evaluation: A process required in computer-assisted instruction. Educational Technology, 27(11), 26-30.
- Paulson, C. E. (1985). Present and future use of computers in high school programs of vocational agriculture in the United States as perceived by teachers of vocational agriculture. (Doctoral Dissertation, Texas A&M University, 1985). Dissertation Abstracts International, 46(10), 2895-A.
- Popham, W. J. (1988). Educational evaluation. Englewood Cliffs, NJ: Prentice Hall.
- Rae, J. (1990). Getting to grips with database design: A step by step approach. Computers in Education, 14(6), 481-488.
- Robleyer, M.D. (1983). Toward more effective microcomputer courseware through application of systematic instructional design methods. AEDS Journal, 17, 23-32.

- Roth, G. & Tesolowski, D. (1986). Performing classroom management functions with competency-based instruction. Microcomputer applications for vocational teachers: A competency-based approach. Idaho State Univ., Pocatello.; Illinois State Board of Education, Springfield, Dept. of Adult, Vocational and Technical Education. (ERIC Document Reproduction Service No. ED 281 025)
- Scriven, M. (1973). Frameworks for planning evaluation studies. In B. R. Worthen & J. R. Sanders (Eds.), Educational evaluation: theory and practice (pp. 62-105). Worthington, OH: C.A. Jones Publishing Company
- Scrogan, L. (1988). The OTA report: New technologies are making a difference. Classroom Computer Learning (October).
- Sherron, J. A. E. (1984). An empirically validated model program for teaching alphabetic keyboarding skills via microcomputer. (Doctoral dissertation, Virginia Polytechnic Institute & State University, 1984). Dissertation Abstracts International, 45(7), 1967-A.
- Status of the American Public School Teacher, 1990-1991. Washington, DC: National Education Association Research Division.
- Stufflebeam, D. L. (1983). CIPP Model for program evaluation. In G. F. Madaus, M. S. Scriven & D. L. Stufflebeam (Eds.), Evaluation models: viewpoints on educational and human services evaluation (pp. 117-125). Boston: Kluwer Academic Publishers.
- Stufflebeam, D.L. (1974). Alternative approaches to education evaluation: A self-study guide for educators. In W. J. Popham (Ed.) Evaluation in education: current applications (pp. 97-103). Berkeley, CA: McCutchan Publishing Corporation.
- Tuckman, B. W. (1985). Evaluating instructional programs. Boston: Allyn and Bacon.
- Tyler, R. W., Gagne, R. M., & Scriven, M. (1967). Perspectives of Curriculum Evaluation. Chicago: Rand McNally & Company.
- Tyre, T. (1989). CMI seen as possible solution to quality of education issue. T.H.E. Journal, (January).

Worthen, B. R. & Sanders, J. R. (1987). Educational evaluation: alternative approaches and practical guidelines. New York: Longman.

## APPENDICES

**APPENDIX A**  
**ELEMENTS OF**  
**LESSONBANK: THE INSTRUCTIONAL MANAGEMENT SYSTEM**

## LessonBank: The Instructional Management System elements

- I. Course and Competency Banking
- II. Task Banking
- III. Question Banking
  - Multiple-choice
  - Matching
- IV. Performance Test Banking
  - Quizzes
- V. Test Generation
  - By variables course and competency
  - By variables software type, function, and difficulty or any combination
    - Random selection
    - User selection
  - Print multiple copies
- VI. Report Generation
  - Task listing
    - By course and competency
    - By software type, function, and difficulty or any combination
  - Competency listing by course
  - Competency and task listing by course
  - Lesson plans by course and competency
  - Test bank questions
  - Output to:
    - Screen
    - Printer
    - File (DOS text)
- VII. Other features
  - Task assignment to competencies
  - Backup features
  - On-line editing of all items

APPENDIX B  
MENU CHOICES FROM LESSONBANK

## LessonBank: The Instructional Management System

### I. Task, Competency, and Course Data

- A. Maintain task data
  - 1. Add task data
  - 2. Edit task data
    - a. Change the description of a task
    - b. Change the function of a task
    - c. Change the difficulty of a task
    - d. Exit to previous menu
  - 3. Delete task data
  - 4. Look at task information
  - 5. Exit to previous menu
- B. Maintain competency data
  - 1. Add competency data
  - 2. Edit competency data
    - a. Change the description of a competency
    - b. Change the number associated with a competency
    - c. Exit to previous menu
  - 3. Delete competency data
  - 4. Print competency data by course
  - 5. View competency data
  - 6. Exit to previous menu
- C. Maintain course data
  - 1. Add new courses
  - 2. Edit course information
  - 3. Delete a course
  - 4. Print course listing
  - 5. Exit to previous menu
- D. Assign tasks to competencies within courses
  - 1. Assign tasks to competencies within courses
  - 2. Unassign tasks
  - 3. Exit to previous menu
- E. Print reports
  - 1. List of tasks by software type, function, or difficulty
  - 2. List of competencies by course
  - 3. List of competencies and tasks by course
  - 4. Exit to previous menu
- F. Exit to main menu

### II. Lesson Plan

- A. Create a new lesson plan
- B. Reprint previous lesson plan
- C. Exit to main menu

### III. Multiple Choice Tests

- A. Create a multiple choice test
  - 1. By course and competency
  - 2. By software type, function, and difficulty or combination
  - 3. Print student copies of previous test
  - 4. Exit to previous menu
- B. Add new multiple choice questions
- C. Print multiple choice questions from test bank by software type
- D. Edit or delete existing multiple choice questions
- E. Exit

### IV. Matching Tests

- A. Create a matching test
  - 1. By course and competency
  - 2. By software type, function, and difficulty or combination
  - 3. Reprint multiple student copies of previous test
  - 4. Exit to previous menu
- B. Add new matching questions
- C. Print matching questions from test bank by software type
- D. Edit or delete existing matching questions
- E. Exit

### V. Performance Tests

- A. Add performance test instructions, assign tasks, create quiz
- B. Edit instructions, task assignments, quiz questions
- C. Delete instructions, task assignments, quiz questions
- D. Print a list of performance tests
- E. Print performance test
  - 1. Select performance test to print
  - 2. Print student copies of performance test
  - 3. Exit to main menu
- F. Print quiz based on performance test
  - 1. Select performance quiz to print
  - 2. Print answer key
  - 3. Print student copies of performance quiz
  - 4. Exit to main menu
- G. Print task data measuring performance test
- H. Exit to main menu

VI. Utilities

- A. Backup database (spans floppies) to any drive
- B. Restore database from any drive
- C. Show directory for any drive
- D. Check the disk situation on any drive
- E. Pack this database. Caution: Make a backup before packing
- F. Backup all files to floppies
- G. Format a floppy disk
- H. Exit to main menu

VII. Exit

APPENDIX C  
LETTERS TO SUPERVISORS AND TEACHERS

November 25, 1992

1 ~  
2 ~  
3 ~  
4 ~  
5 ~

Dear 6 ~ :

As a part of a research project at Virginia Polytechnic Institute under the direction of Dr. Jeffrey R. Stewart, a database program entitled LessonBank: The Instructional Management System, is in the final stages of development.

The program is designed to provide an automated method of organizing courses, competencies, and tasks. As you are undoubtedly aware, a listing of tasks needed to achieve a competency was developed by a research team and resulted in the *Business Computer Software Curriculum Series* published by the Virginia Department of Education in 1990. This publication was organized around computer competencies extracted from the *Business Education Suggested Course Competencies and Performance Objectives* published by the Virginia Department of Education in 1989.

Anne Rowe has provided me with your name. I am asking you to provide me with the names of two teachers who would be willing to spend a few hours using the program, completing some "assignments" and providing evaluation feedback.

LessonBank is a menu-driven database program for managing instruction. Instructors are able to use the system for planning course content organized around competencies that require instruction in the use of computer software. The automated system will contain information about individual courses and competencies related to software use. The system centers around 1,100 tasks organized by software type, function, and level of difficulty. In addition, the database contains matching and multiple choice test questions which can be used to generate tests based on specific competencies and software types.

1 ~

Page 2

November 25, 1992

I would appreciate your providing me with the names and addresses of at least two business education teachers who you feel would be willing to participate in this study. Please complete the attached form and FAX it to me at 919-398-8225 or drop it in the mail by December 3 1992. I will contact each teacher personally seeking their acceptance as an evaluator.

Sincerely yours,

Andrea E. Eason

Enclosure

December 10, 1992

1 ~

Dear 2 ~ :

As a part of a research project at Virginia Polytechnic Institute under the direction of Dr. Jeffrey R. Stewart, a database program entitled LessonBank: The Instructional Management System, is in the final stages of development.

The program is designed to provide an automated method of organizing courses, competencies, and tasks. As you are undoubtedly aware, a listing of tasks needed to achieve a competency was developed by a research team and resulted in the *Business Computer Software Curriculum Series* published by the Virginia Department of Education in 1990. This publication was organized around computer competencies extracted from the *Business Education Suggested Course Competencies and Performance Objectives* published by the Virginia Department of Education in 1989.

LessonBank is a menu-driven database program for managing instruction. Instructors are able to use the system for planning course content organized around competencies that require instruction in the use of computer software. The automated system will contain information about individual courses and competencies related to software use. The system centers around 1,100 tasks organized by software type, function, and level of difficulty. In addition, the database contains matching and multiple choice test questions which can be used to generate tests based on specific competencies and software types.

2~

Page 2

December 10, 1992

Your name has been provided to me as a teacher who may be interested in testing and evaluating the program in its final stages of development. If you agree to participate, please return the enclosed form to me before leaving for the Christmas holidays.

During the first week in January, 1993, you will receive a copy of the program, an instruction manual, and several "assignments" to complete. In addition, you will be asked to complete a questionnaire concerning your impressions and difficulties with the program. This information will need to be returned to me by the first of February.

I hope that I can count on your input as LessonBank is developed for final distribution to business education teachers in the State of Virginia.

Sincerely yours,

Andrea E. Eason

Enclosure

**APPENDIX D**  
**STRUCTURED APPLICATION ASSIGNMENT**

### *LessonBank: The Instructional Management System*

The exercises in each part below test each menu choice in the system.

#### Directions:

1. Read through pages 1-16 in the instruction manual before you begin. Refer to the instruction manual as necessary when completing the assignments for each part.
2. Load *LessonBank* as instructed.
3. Complete each assignment. The data in italics is to be keyed in by you as instructed.
4. Attach any printed reports to the instructions for that Part.
5. Your job is to evaluate the program and the instruction manual, not the contents of the database (tasks, competencies, questions, etc.) At the end of each Part, please record any comments you have about the program, on-screen forms, reports, etc. You may write on the back of the page if necessary.

The following is Main Menu of *LessonBank*

Task, Competency and Course Data  
Lesson Plan  
Multiple Choice tests  
Matching tests  
Performance Tests  
Utilities

Exercises Part I through Part V may be completed by accessing the first choice on the Main Menu: Task, Competency, and Course Data.

Exercise Part VI is completed by choosing Lesson Plan from the Main Menu.

Exercises in Parts VII and VIII are completed by choosing Multiple Choice tests from the Main Menu.

Exercises in Parts IX and X are completed by choosing Matching tests from the Main Menu.

Exercise Part XI is completed by choosing Performance tests from the Main Menu.

Exercise Part XII is completed by selecting Utilities from the Main Menu.

To introduce a new user to the listing of the tasks and competencies defined for use with the existing system, we will begin our exercise by printing and reviewing various reports.

Lesson Bank is a menu driven system that will guide the user through these assignments. Each of the instructions listed below are separate activities and require the user to begin with the "Print Reports" menu choice in the Tasks, Competency, and Course Data menu.

## Part I. ADD/UPDATE TASK DATA

You have looked over the list of tasks and have decided to determine whether *LessonBank* will allow you to add and/or update several tasks.

All tasks require they be identified by software type, function, and difficulty. This requires advance planning by the teacher. For this exercise, these decisions have been made for you.

Study the menu choices for Add/Update Task data and select the appropriate ones to complete the following activities. Please read each instruction below carefully before beginning the exercises.

1. Add the following two tasks for  
software type: DOS and Fundamentals  
function: vocabulary  
difficulty: beginning

*Identify motherboard*  
*Identify operating system*

2. Change the difficulty of "Identify motherboard" to intermediate.
3. Add the following task for  
software type: database;  
function: vocabulary;  
difficulty: advanced.

*Identify variable*

4. Delete the task "Identify operating system" from DOS and Fundamentals, vocabulary, beginning.
5. Look at task data for DOS and Fundamentals, vocabulary, beginning. (This option would allow the user to make sure the data had been entered correctly and did appear when viewed.)

## Part II. ADD/UPDATE COMPETENCY DATA

Study the menu choices for Add/Update Competency data and select the appropriate ones to complete the following activities:

1. Add the following competency for the course Accounting Computer Applications:  
  
*16*  
*Select a business application and describe how it would be automated.*
2. Edit that same competency by inserting the word common as follows: Select a common business application and describe how it would be automated.
3. Change the competency number from 16 to 34.
4. Print the competency data for Information/Word Processing.
5. View the competency data for Accounting Computer Applications. Was the new competency added? (Y or N)
6. Delete competency 34 for Accounting Computer Applications.
7. View the competency data for Accounting Computer Applications. Was the new competency deleted? (Y or N)

### Part III. ADD/UPDATE COURSE DATA

1. Add the Shorthand course, 6211.
2. Edit the name of the Shorthand course to Shorthand I.
3. Print list of courses.
4. Delete Shorthand I course.

### Part IV. ASSIGN TASKS TO COMPETENCIES WITHIN COURSES

1. Assign the following task to Information/Word Processing, competency No. 9.  
software type: database;  
function, vocabulary;  
difficulty, advanced;  
Identify variable
2. Print a listing of competency No. 9 and tasks for Information/Word Processing.
3. Delete the task assignment (unassign) made in item #1 above.
4. Repeat instruction No. 2 to verify the tasks were "unassigned."

## Part V. PRINT REPORTS

The following reports should be sent to a printer.

1. Print a list of all tasks for the software type word processing.
2. Print a list of tasks for word processing and the functions access software/data and data/text entry.
3. Print a list of tasks for word processing formatting functions.
4. Print a list of tasks ONLY for beginning word processing formatting functions.
5. Print a list of competencies for Information/Word Processing.
6. Print a listing of competencies and tasks for Accounting Computer Applications, competencies 22 and 23.
7. Exit to previous menu when finished printing reports

## Part VI. CREATING LESSON PLANS

1. Choose create a lesson plan.
2. Select a course (any course).
3. Select the competency or competencies on which you wish to base your lesson plan.
4. Type the appropriate name and date information and press the [ALT] key to continue.
5. Print a copy to the printer.
6. Reprint a copy to a file.
7. Exit from *LessonBank* when finished and load your favorite word processor.
8. Retrieve the lesson plan file you just saved. Change the left and right margins in the word processor to 1/2 inch. You may now alter this file in any way you wish. (Add other objectives, delete some data, etc.). When you finish "polishing it up," print a copy.

## Part VII. CREATING MULTIPLE CHOICE TESTS

Print the following exercises to the printer:

1. Create a multiple choice test by course and competency for Information/Word Processing, competency 17. Let the computer generate a 10 question test randomly. Use the following heading for the test:

*Information/Word Processing*  
*Spreadsheet Test I*  
*Mrs. Smith's First Period*

2. Create a multiple choice test by course and competency for Information/Word Processing, competency 17. You select 10 questions for the test. Use the following heading for test:

*Information/Word Processing*  
*Spreadsheet Quiz I*

3. Create a multiple choice test for software type word processing. Let the computer generate a 20 question test randomly. Use the following heading for the test:

*Business Supervision and Management*  
*Word Processing Test I*  
*Current Date*

4. Create a multiple choice test for software type word processing. Pick 20 questions. Use the following heading for the test:

*Information/Word Processing*  
*Word Processing Test II*  
*Current Date*

4. Create a multiple choice test for DOS and Fundamentals, Spreadsheets, Functions 1-5. Let the computer generate a 20 question test randomly. Use the following heading for the test:

*DOS and Fundamentals, Spreadsheets*  
*Random 20*  
*Current Date*

5. Repeat No. 4 only you select the questions this time for a total of 10. Use same heading only change line 2 to read "*Teacher Selection 10*".
6. Create a multiple choice test for database, beginning difficulty level. Let computer randomly generate a 10 question test. Use the following heading:  
  
*Database*  
*Random 10*  
*Current Date*
7. Repeat No. 6 only you select the questions this time for a total of 10. Use same heading only change line 2 to read "*Teacher Selection 10*".
8. Create a randomly generated 10 question multiple choice test for data communications, vocabulary, intermediate and beginning difficulty levels. Use heading as in No. 6 only replace line 1 with Data Communications.
9. Repeat No. 8 only you select 6 questions. Use heading as follows:  
  
*Data Communications*  
*Teacher Selection 6*  
*Current Date*
10. Print 5 copies of the test generated in instruction 9.

## Part VIII. ADD, UPDATE, PRINT MULTIPLE CHOICE TEST QUESTIONS

**Note:** Before adding questions to the question bank, the teacher must determine which task the question measures. Questions cannot be added without a task number. (This has been figured in advance for you for these exercises by referring to a listing of tasks by software type.)

1. Add the following multiple choice questions:

*Documents prepared on word processing systems can be*

- a. stored on disk
- b. edited
- c. printed multiple times
- d. all of the above

*Task No. 1287 Answer: d*

*Which of the following is a feature associated with a spreadsheet?*

- a. document editing
- b. bold text
- c. automatic recalculation
- d. centering lines on a page

*Task No. 1013 Answer: c*

*Which of the following personal computer software packages is used to perform financial calculations?*

- a. graphics programs
- b. spreadsheets
- c. electronic mail
- d. word processing

*Task No. 1006 Answer: b*

*The intersection of a column and row on a spreadsheet is called a*

- a. window
- b. cell
- c. formula
- d. label

*Task No. 1000 Answer: b*

*Which of the following terms is associated with database management systems?*

- a. automatic recalculation
- b. select and sort
- c. cut and paste
- d. cells

*Task No. 1747 Answer: d (note: this will be changed later)*

2. Print all the multiple choice questions for spreadsheets.
3. Delete the 4th question above.
4. Edit the last question above and change the answer to b.

## Part IX. CREATING MATCHING TESTS

Print the following exercises to the printer:

1. Create a matching test by course and competency for Information/Word Processing, competency 17. Let the computer generate a 10 question test randomly. Use the following heading for test:

*Information/Word Processing*  
*Spreadsheet Test I*  
*Mrs. Smith's First Period*

2. Create a matching test for competency 17 in Information/Word Processing. Pick 10 questions. Use the following heading for test:

*Information/Word Processing*  
*Spreadsheet Quiz I*

3. Create a matching test for word processing. Let the computer generate a 10 question test randomly. Use the following heading for the test:

*Business Supervision and Management*  
*Word Processing Test I*  
*Current Date*

4. Create a matching test for word processing. Pick 10 questions. Use the following heading for the test:

*Information/Word Processing*  
*Word Processing Test II*  
*Current Date*

4. Create a matching test for DOS and Fundamentals and Spreadsheets, Functions 1-5. Let the computer generate a 10 question test randomly. Use the following heading for the test:

*DOS and Fundamentals, Spreadsheets*  
*Random 10*  
*Current Date*

5. Repeat No. 4 only you select the questions this time. Use same heading only change line 2 to read "*Teacher Selection 10*".

6. Create a matching test for database, beginning difficulty level. Let computer randomly generate a 10 question test. Use the following heading:

*Database*  
*Random 10*  
*Current Date*

7. Repeat No. 6 only you select the questions this time for a total of 10. Use same heading only change line 2 to read "*Teacher Selection 10*".
8. Create a randomly generated 10 question matching test for data communications, vocabulary, intermediate and beginning difficulty levels. Use heading as in No. 6 only replace line 1 with *Data Communications*.

9. Repeat No. 8 only you select 6 questions. Use heading as follows:

*Data Communications*  
*Teacher Selection 6*  
*Current Date*

10. Print 5 copies of the test generated in instruction 9.

## Part X. ADD, UPDATE, PRINT MATCHING TEST QUESTIONS

Note: Before adding questions to the question bank, the teacher must determine which task the question measures. Questions cannot be added without a task number. This has been determined in advance for you by referring to a listing of tasks by software type.

1. Add the following matching questions:

*Something that uniquely identifies a record.*

Answer: *key field* Task No. 1736

*Rearrange records in a database into some order.*

Answer: *sort* Task No. 1748

*Print data in a certain format.*

Answer: *report* Task No. 1746

*Display the contents of a DOS ASCII file*

Answer: *type* Task No. 1215

*A group of characters that represent a single piece of data.*

Answer: *field* Task No. 1728 (*this is wrong purposely*)

*All the information about students in a school.*

Answer: *file* Task No. 1731

2. Print all the matching questions for software type database.
3. Delete question No. 4 above.
4. Edit question No. 5 above and change the task number to 1729.

## Part XI. PERFORMANCE TESTS

The performance tests consist of a series of numbered instructions. The teacher would need to determine which tasks the test measures. A short "quiz" of questions to be answered about the assignment can be developed and recorded as well. The performance test is developed around several tasks that need to be measured. These tasks should be determined before a user begins to enter a new test.

Print a list of existing performance tests:

1. You are interested in determining how many performance tests currently exist. Make the appropriate choice from the Performance Test menu to print a list of all performance tests.

Enter a new performance test:

2. Please enter the following performance test for Desktop Publishing, Intermediate Level, No. 1

*Title: Desktop Publishing--Intermediate Test #1*

1. *Create a one-page newsletter for FBLA.*
2. *Collect two text files on disk. One will be a story about FBLA. The second text document should be a calendar of events planned for the semester or year. The two files should be about 1,500 words long.*
3. *Load your desktop publishing software program.*
4. *Open a template for a 3 column newsletter or create a new document with 3 columns. If you open a template, save it under a new name that includes the issue number, such as VOL1-04.*
5. *Create a banner using the letters FBLA with ruled lines and shading of some type. You may be original with this.*
6. *Below the banner place the identification of the newsletter as Volume 1, No. 4 October/November 1992*
7. *Use only one or two different typefaces in the newsletter.*
8. *All columns should bottom out to the same point.*

9. *Use 10 pt. Times with auto leading for body copy. Use 8 pt. Times for calendar text. Use 14 pt. Times Bold for Volume/Issue ID. Use 24 pt. Times Bold for main article title.*
10. *If the final text is too long to fit within the one page, change the leading to shrink the text.*
11. *Use full justification and hyphenation on the body text.*
12. *Save the document and print on a laser printer.*

The following tasks are measured by the performance test:

*1571, 1572, 1573, 1585, 1587, 1589, 1591, 1594, 1595, 1596, 1598, 1602, 1603, 1615, 1617, 1618, 1620, 1623, 1628, 1634, 1650, 1651, 1652, 1653, 1667, 1690, 1693, 1712, 1713*

The following quiz should be entered:

1. *How many columns are in the newsletter?*  
*Answer: three*
2. *What is the leading for the body type?*  
*Answer: automatic*
3. *What is the point size of type for the body text?*  
*Answer: 10 pt.*
4. *What is the volume number of this publication?*  
*Answer: volume 1, No. 4*
5. *What is the orientation for the newsletter--wide or tall?*  
*Answer: tall*

Editing a Performance test:

3. Edit performance test for Desktop Publishing, Intermediate Level, No. 1.
  - a. Insert the following instruction after No. 9:  
*Place the calendar in the center column with ruled lines to set it off.*

- c. Add task number 1709 to the list of those that measure this performance test. There are no new questions to add to the quiz.

Delete a Performance Test:

4. Delete performance test SSI1.

Print a performance test:

5. Make the appropriate menu choice from the Performance Test menu to print the intermediate performance test #1 for Spreadsheets.
6. Print 3 copies of SSI1 for student use.

Print a performance quiz, multiple quizzes, an answer key, and a report of tasks measured by

7. Print the quiz for SSI1.
8. Print the answer key for SSI1.
9. Print 3 copies of performance quiz for the students.
10. Print the tasks measured by performance test for SSI1.

## Part XII. UTILITIES

This part of *LessonBank* offers the user several DOS commands by selecting from the Utilities menu. You will need a disk for formatting and at least three high density disks for backing up the entire directory where the database is store.

1. Format a disk in drive A or B
2. Backup the database to disks in drive A or B.
3. Show a directory for drive C.
4. Check the disk situation on Drive C.
5. Pack the database.
6. Copy the database files to drive A:
7. Restore the database from drive A or B using the backup created in No. 2 above.

APPENDIX E  
VOLUNTEER SOLICITATION FORM

Yes, I am willing to evaluate LessonBank: The Instructional Management System

Name\_\_\_\_\_

Home Address\_\_\_\_\_

City\_\_\_\_\_ State\_\_\_\_\_ Zip\_\_\_\_\_

Home Phone(\_\_\_\_\_)\_\_\_\_\_

School\_\_\_\_\_

Address\_\_\_\_\_

City\_\_\_\_\_ State\_\_\_\_\_ Zip\_\_\_\_\_

Office Phone(\_\_\_\_\_)\_\_\_\_\_

Concerning computer use, I would rate myself as a:

\_\_\_\_Novice \_\_\_\_Casual User \_\_\_\_Heavy User \_\_\_\_Expert

Courses I teach where computers are used:

\_\_\_\_\_  
\_\_\_\_\_

Courses I teach where computers are not used:

\_\_\_\_\_  
\_\_\_\_\_

My computer education includes (check all that apply):

\_\_\_\_self-taught \_\_\_\_workshops \_\_\_\_0-3 sem. hrs.

\_\_\_\_4-6 sem. hrs. \_\_\_\_7-9 sem. hrs. \_\_\_\_10+ sem. hrs.

Other\_\_\_\_\_

Signed:\_\_\_\_\_Date:\_\_\_\_\_

APPENDIX F  
SUMMARY EVALUATION FORM

### Overall Evaluation of LessonBank

Very poor =1  
 Below average = 2  
 Average = 3  
 Above Average = 4  
 Very Good = 5

Program format	1	2	3	4	5
Consistent use of terminology					
Clear and concise instructions					
Well organized and sequential					
Output format					
Screen and printer or other peripherals used					
Report generation is logical and readable					
Screen displays are consistent and easy to use					
Input and operation					
Procedure for entering data is consistent					
Prompts are plentiful and clear					
System commands are logical					
Error trapping is sufficient					
Error messages are understandable					
Restart and recovery is easy and allowed					
Sufficient data field length					
Number of records is adequate (capacity)					
Speed to access records is adequate					
Maintenance of file is acceptable					
Installation and Backup					
Ease of installation					

Overall Ease of operation					
Ease of backup					

Comments:

APPENDIX G  
DATA FROM CHECK SHEET AND EVALUATION FORM OF SIX SUBJECTS

Compilation of data from check sheet and evaluation form of six subjects

Assignment Description		Time to Complete	Problems/Comments/Suggestions
Part I Add/Update Task Data	1	10 min.	Repetitive tasks Trying to get acquainted with function keys and look up steps; too many steps Documentation not clear to enter original difficulty when going through menus. A way to move screen by screen would speed up browsing and/or to enter task # if known and advance immediately to desired task. Expected task to be in alphabetical order as beginning were. Note that tasks will be in order by task number would be helpful. Instructions were easy to follow! Needed more overall explanation about the project--what the outcomes were to be
	2	10 min.	
	3	25 min.	
	4	12 min.	
	5	30 min.	
	6	45 min.	

Assignment Description		Time to Complete	Problems/Comments/Suggestions
Part II Add/Update Competency Data	1	12 min.	Too much going back and forth
	2	8 min.	None
	3	10 min.	In editing, option to enter competency number would be helpful; view screen does not indicate F8 & F7 to move up and down.
	4	10 min.	Box where # and competency are inserted is awkward to use when prompted as in directions--no goes in first box. Select...goes in second box. Poor directions--enter 16 (rather than No. 16) would be an improvement.
	5	40 min.	I overlooked SHIFT/F8 notation and I had to back track.
	6	15 min.	No problems. Exercise went well.
Part III Add/Update Course Data	1	2 min.	No comment
	2	7 min.	No comment
	3	5 min.	Change in editing procedure from above was a little confusing.
	4	10 min.	Total confusion!
	5	15 min.	Instructions were easy to follow!
	6	5 min.	No problems

Assignment Description		Time to Complete	Problems/Comments/Suggestions
Part IV Assign Tasks to Competencie s within Courses	1	15 min.	I kept highlighting Stop instead of Continue Took 3 attempts; could not get to function and difficulty; each time wanted to print had to repeat steps. Deleting is not unassigning (using space bar) Very confusing! For each printing, I had to go through the menu selection twice; I am not sure why. "Function" confused us. We felt we should be in a different menu. We did not realize that by pressing F8 there was more to the list.
	2	45 min.	
	3	15 min.	
	4	10 min.	
	5	35 min.	
	6	30 min.	
Part V Print Reports (6 reports)	1	20 min.	No comment Too many steps to print; I could key in commands faster. Option 3 performs differently from other 2 and makes for confusion. Why not use Shift +F8 in option 1 & 2 to move to next box? After Alt in option 3, must choose continue to print; however in option 1 and 2 it is "stop here." Better! It's starting to make some sense. No problems; instructions were easy to follow! We omitted two items because we were unsure on how to do them. After completing Exercises 8 & 9 we could do them. We needed to be told to "Stop here."
	2	20 min.	
	3	20 min.	
	4	10 min.	
	5	30 min.	
	6	50 min.	

Assignment Description		Time to Complete	Problems/Comments/Suggestions
Part VI Creating Lesson Plans	1	15 min.	Couldn't get to print through WordPerfect. Copy included.
	2	50 min.	5 attempts. Could not get printout; all steps were followed carefully; see print screens; could not retrieve into WordPerfect or DW4
	3	20 min.	Unable to perform this function. Competency chosen did not come up in lesson plan.
	4	10 min.	Lesson plan did not print. Could not retrieve lesson plan in WordPerfect.
	5	20 min.	Easy instructions to follow!
	6	40 min.	Could not do step 8 (retrieve lesson plan in word processor)
Part VII Creating Multiple Choice Tests (11 tests)	1	25 min.	On "Teacher Selection," had trouble keeping up with how many selections I had made. As I stopped to read each question, I would lose track of what selection number I was on.
	2	130 min.	Screen should indicate # of questions selected. Questions would not print. Each time I desired to work with questions I had to exit back to C> and start software again. Very cumbersome
	3	45 min.	Shift + F8 to move to next box to be consistent; continue to enter printing mode; When selecting questions, screen showing number selected redo question at bottom of screen, answers at top is distracting.

Assignment Description		Time to Complete	Problems/Comments/Suggestions
Part VII Creating Multiple Choice Tests (11 tests)	4	40 min.	Extremely frustrating at first. Critical on screen prompts (enter means down, keep arrow on choice with Shift/F8 to move to competencies) should be much more prominent.
	5	180 min.	PROBLEMS! Difficulty keeping track of number of questions selected. When completing 4B (assignment) the computer locked; I had to do a warm boot; lost all data; had to reload original program.
	6	30 min.	No problems
Part VIII Add/Update Multiple Choice Test Questions	1	20 min.	No comment
	2	22 min.	Cannot add questions; will not accept task numbers provided. Tried 3 questions; task number not valid
	3	25 min.	Easy to manipulate; To edit I had to enter delete and answer no. Is This the way it is supposed to work?
	4	20 min.	Very unforgiving of mistakes. Cursor movement to task # is unexpected and at this point causes many problems.
	5	35 min.	My confusion--STOP HERE to print or EXIT to print.
	6	10 min.	Here's where we discovered more items in the software type, function, and difficulty

Assignment Description		Time to Complete	Problems/Comments/Suggestions
Part IX Creating Matching Tests (10 tests)	1	60 min.	Waited a lot for software to make question selections.
	2	7 min.	No questions in the database. Could not complete this section.
	3	45 min.	Each instance where I made selection of test questions, message came up on screen that program was randomly selecting questions.
	4	30 min.	Prompt at bottom of screen and answer at top--confusing.
	5	180 min.	PROBLEMS! Computer locked when attempting to print; I had to reload the original program!
	6	13 min.	In step 3 (create a test for word processing) we had to know to mark all the questions in order to generate the test.
Part X Add/Update Matching Test Questions	1	15 min.	No comment
	2	35 min.	Could not add/update questions; tried 3 times to add matching questions.
	3	5 min.	No problems--edit worked properly in this section.
	4	10 min.	No comment.
	5	30 min.	No problems.
	6	10 min.	No problems.

Assignment Description		Time to Complete	Problems/Comments/Suggestions
Part XI Create performance tests	1	40 min.	No spreadsheet No. SSI1 to delete Did not attempt Red block on quiz very hard on eyes; instruction #4 says to delete SSI1; then instructions 5-10 used it. Since I had deleted it I substituted another test. How to create quiz? How to retrieve and print a deleted file? Explanation needed to clear for tens. SSI1 was deleted in step 4; yet, the directions call for you to work with the file in steps 7-10. I used DPI1 to complete the exercises. We did realize how we needed to enter all the task numbers. Also, we deleted SSI1 in Step 4 and could not make copies.
	2	0 min.	
	3	20 min.	
	4	15 min.	
	5	90 min.	
	6	25 min.	
Part XII Utilities	1	40 min.	In backup, when I selected b:, computer would try to read a: Would not back up on high density disks. When finished, restoring came to blank screen, no program, no prompt, could not even reset. Had to turn computer off and back on. could not make backup--no test on assignment sheet. The problems that I had are listed on the assignment sheet. We had a box of defective diskettes and were unable to do this section.
	2	45 min.	
	3	20 min.	
	4	10 min.	
	5	120 min.	
	6	0 min.	

## VITA

Andrea Emmot Eason graduated from Independence High School, Independence, Kansas, in 1960. She earned an Associate of Science degree from Independence Community College, Independence, Kansas, 1962. She attended Pittsburg State University, Pittsburg, Kansas, and received the B. S. degree in business education in 1966. She was granted the M. S. degree in business education from Virginia Polytechnic Institute and State University, Blacksburg, Virginia, in 1968.

Andrea's first teaching position was with the Department of Business at Richard Bland College, Petersburg, Virginia, 1967-1969. In 1969, she was employed by Chowan College in the Department of Business and currently serves as an Associate Professor of Business.

In 1992, Andrea was appointed to the position of Director of Academic Computing at Chowan. In addition, she continues her primary teaching responsibilities in the computer information systems area.

During her residency in the doctoral program at Virginia Polytechnic Institute and State University, Andrea worked with the National Center for Research in Vocation Education, and taught a class within the Division of Vocational-Technical education. Andrea completed the doctoral program in May, 1993.

THE FORMATIVE EVALUATION AND REVISION OF AN  
INSTRUCTIONAL MANAGEMENT SYSTEM  
FOR BUSINESS COMPUTER COMPETENCIES

by

Andrea Emmot Eason

(ABSTRACT)

The purpose of this project was to (1) evaluate and revise a computer-based instructional management system developed to organize business computer competencies, and (2) develop and revise documentation for using the system.

The instructional management system consists of a database and various applications employing relational database architecture. The resulting system will be used by Virginia business teachers in implementing their curricula.

The prototype system was developed initially to organize a taxonomy of tasks identified to measure computer competencies. The computer competencies were extracted from the Business Education Suggested Course Competencies and Performance Objectives, published by the Virginia Department of Education in 1989. The taxonomy resulted in the publication of the Business Computer Software Curriculum Series in 1990. This latter publication forms the core of the instructional management system. The 1990 curriculum guide was ultimately expanded to include multiple choice and matching test questions organized to measure the tasks. A

module of performance tests was incorporated into the system as well.

The resulting program was titled LessonBank: The Instructional Management System. LessonBank is a menu-driven program designed to be used as a tool to assist teachers with the management function of teaching. The system incorporates a number of components. The components are (a) course banking, (b) competency banking, (c) task banking, (d) test question banking, (e) performance test banking, (f) test generation, (g) lesson plan generation, (h), reporting, and (i) database maintenance.

A three phase formative evaluation procedure was used to improve the program and the user's manual. Data were collected from the evaluators as the subjects in each phase tested all components of the entire system. The developer identified the needed revisions and made necessary corrections. Following the third phase, it was determined that the prototype program was ready for distribution to business teachers in the state of Virginia.

Suggestions for use of the computer-based instructional management system are given. Recommendations are made for developers. In addition, recommendations are made for teachers, supervisors, and administrators who have responsibility for improvement and management of instruction.