Using a problem analysis model to enhance student learning in computer programming

Hsiao-Shen Wang
Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/rtd

Part of the Communication Technology and New Media Commons, Curriculum and Instruction Commons, Educational Psychology Commons, Instructional Media Design Commons, and the Science and Mathematics Education Commons

Recommended Citation
Wang, Hsiao-Shen, "Using a problem analysis model to enhance student learning in computer programming" (1996). Retrospective Theses and Dissertations. 11498.
https://lib.dr.iastate.edu/rtd/11498

This Dissertation is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.
Using a problem analysis model to enhance student learning in computer programming

by

Hsiao-Shen Wang

A dissertation submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Department: Curriculum and Instruction

Major: Education (Curriculum and Instructional Technology)

Major Professors: Constance P. Hargrave and Ann D. Thompson

Iowa State University

Ames, Iowa

1996
This is to certify that doctoral dissertation of

Hsiao-Shen Wang

has met the dissertation requirements of Iowa State University

Signature was redacted for privacy.

Co-Major Professor

Signature was redacted for privacy.

Co-Major Professor

Signature was redacted for privacy.

For the Major Department

Signature was redacted for privacy.

For the Graduate College
# TABLE OF CONTENTS

**LIST OF TABLES**

**LIST OF FIGURES**

**ABSTRACT**

**CHAPTER I. INTRODUCTION**
- Background of the Study
- Statement of the Problem
- Purpose of the Study
- Research Hypotheses
- Limitations of the Study
- Definition of Terms
- Summary

**CHAPTER II. REVIEW OF LITERATURE**
- Constructivism
- The Difficulties of Novice Programmers
- Using Conceptual Models of Computer Operations in Computer Programming Instruction
- Problem-solving oriented instruction in Computer Programming

**CHAPTER III. METHODOLOGY**
- Overview of the Research Design
- Sample
- Treatments
  - Control group
  - Problem analysis group
- Database Simulation
  - Description of the Database Simulation
  - Contents of the Database Simulation
  - The role of the Database Simulation
- Holistic Instructional Approach
  - Description
  - Four-step procedure of the holistic instructional approach
  - The role of the holistic instructional approach
- Instrumentation
  - The Computing Survey
  - Database assignments
  - The Database Examination
- Variables of the Study
- Procedure
Data Analysis 51
The Computing Survey 52
Database assignment and Database Examination 52
A Pilot Study 57

CHAPTER IV. RESULTS AND FINDINGS 60
Analysis of Research Components 60
The Computing Survey 60
Results of Hypothesis Testing 63
Summary 74

CHAPTER V. SUMMARY, DISCUSSION AND RECOMMENDATIONS 76
Summary 76
Discussion 80
Recommendations 87
Conclusion 88

APPENDIX A. COGNITIVE INFORMATION PROCESSING AND THE ACT* THEORIES 90
APPENDIX B. WORKSHOP SCHEDULES 98
APPENDIX C. HUMAN SUBJECTS APPROVAL FORM AND CONSENT FORM 101
APPENDIX D. LESSON PLAN AND DATABASE ASSIGNMENTS 104
APPENDIX E. THE COMPUTING SURVEY 112
APPENDIX F. THE DATABASE EXAMINATION 120
APPENDIX G. DATABASE UNIT OUTLINE 124
APPENDIX H. DATABASE REFERENCE MATERIALS 127
REFERENCES 139
ACKNOWLEDGMENTS 143
LIST OF TABLES

Table 3.1. Demographic information of the control and experimental group participants 37
Table 3.2. Pearson Product Moment Correlation coefficients and t-tests for paired evaluations of the control group participants’ scores on database assignments 1 - 5 54
Table 3.3. Pearson Product Moment Correlation coefficients and t-tests for paired evaluations of the problem analysis group participants’ scores on database assignments 1 - 5 55
Table 3.4. Means and standard deviations of participants’ scores on database assignments by group 56
Table 3.5. Pearson Product Moment Correlation coefficients and t-tests for paired evaluations of the control and problem analysis participants’ scores on Database Examination 58
Table 3.6. Means and standard deviations of participants’ scores on Database Examination by group 59
Table 4.1. T-tests on basic computing knowledge by group 61
Table 4.2. Factor loading and internal consistencies of the attitude section of the Computing Survey 62
Table 4.3. Means, standard deviations, and t values of the attitude factors for the control and problem analysis group 63
Table 4.4. Analysis of covariance (ANCOVA) of database assignment 1 by treatment groups (Covariate: BCC) 65
Table 4.5. Analysis of covariance (ANCOVA) of database assignment 2 by treatment groups (Covariate: BCC) 65
Table 4.6. Analysis of covariance (ANCOVA) of database assignment 3 by treatment groups (Covariate: BCC) 66
Table 4.7. Analysis of covariance (ANCOVA) of database assignment 1 by treatment groups (covariate: Attitudes) 66
Table 4.8. Analysis of covariance (ANCOVA) of database assignment 2 by treatment groups (covariate: Attitudes) 67
Table 4.9. Analysis of covariance (ANCOVA) of database assignment 3 by treatment groups (covariate: Attitudes) 67
Table 4.10. Analysis of covariance (ANCOVA) of database assignment 4 by treatment groups (covariate: BCC) 69
Table 4.11. Analysis of covariance (ANCOVA) of database assignment 5 by treatment groups (covariate: BCC) 69
Table 4.12. Analysis of covariance (ANCOVA) of database assignment 4 by treatment groups (covariate: Attitudes) 70
Table 4.13. Analysis of covariance (ANCOVA) of database assignment 5 by treatment groups (covariate: Attitudes) 70
Table 4.14. Analysis of covariance (ANCOVA) of Database Examination (general syntax) by treatment groups (covariate: BCC) 72
Table 4.15. Analysis of covariance (ANCOVA) of Database Examination (general syntax) by treatment groups (covariate: Attitudes) 72
Table 4.16. Analysis of covariance (ANCOVA) of Database Examination (programming problems) by treatment groups (covariates: BCC, Attitudes) 73
Table 4.17. Analysis of covariance (ANCOVA) of Database Examination (programming problems) by treatment groups (covariate: BCC) 73
Table 4.18. Analysis of covariance (ANCOVA) of Database Examination (programming problems) by treatment groups (covariate: Attitudes) 74
LIST OF FIGURES

Figure 3.1. The Database Simulation 41
Figure 3.2. Clean your table Simulation 42
Figure 3.3. Sort your data Simulation 44
Figure 3.4. Find your data Simulation 44
Research on computer programming suggests that novice programmers possess inert knowledge when trying to solve programming problems. Moreover, research on teaching and learning computer programming indicates that offering appropriate conceptual models of computer programming concepts to novice programmers enhances their mental models and reduces their misconceptions in computer programming. The purpose of this study was to examine the effectiveness of a problem analysis learning model of computer programming to help novice programmers overcome their inert knowledge and learn a programming language.

The problem analysis learning model combines a conceptual model and a holistic instructional approach for computer programming instruction. The conceptual component of the problem analysis model includes several computer simulations of database concepts. The purpose of the conceptual component of the problem analysis model is to offer students an opportunity to manipulate data in the computer simulations before formal instruction in order to help them construct their own knowledge of basic database concepts. The purpose of the holistic component of the problem analysis learning model is to help students integrate their programming knowledge to solve database problems. The holistic approach includes a four-step process that consists of problem introduction, problem diagnosis, learning activities, and database assignments.

This study involved 100 inservice teachers enrolled in a basic computer programming workshop at The Institute for Secondary Schools Teachers in Taiwan (ISST). The teachers were randomly assigned to one of the two workshops conducted in this study (43 teachers in
the control group and 53 teachers in the experimental group). Each workshop consisted of 36 hours of instruction over five days. Three subjects were taught in each workshop: basic computer concepts (BCC), Chinese word processing, and database activities.

The results indicated that the problem analysis learning model helped the participants develop more complete mental models of basic database concepts. Moreover, the participants in the problem analysis learning model developed better database programming skills than those in the traditional computer programming workshop.

The results of this study may provide a useful conceptual framework for the design of a computer programming course for teachers. Due to the results of this study, computer practitioners/teachers may re-organize their instructional methods used with existing content materials to enhance student learning in computer programming. Future researchers may use the problem analysis model as a foundation to explore learning in other subjects. Moreover, researchers could examine parts of this model in greater detail to identify specific items or procedures that contribute to student learning. Finally, this research study also provides a workshop structure to help inservice teachers increase their computing proficiency, and it may assist institutions in organizing their training programs.
CHAPTER I. INTRODUCTION

Teaching and learning computer programming is a complex task for both teachers and students. Several instructional models have been used to help students learn computer programming. Two instructional models that have been successful in improving novice programmers' learning in computer programming are conceptual models and problem-solving oriented instructional models. The purpose of this study was to examine the effectiveness of a combination of conceptual and problem-solving oriented instructional models to improve student learning in computer programming. In this chapter, an overview of the study is provided; this chapter is divided into five sections: 1) background of the study, 2) statement of the problem, 3) purpose of the study, 4) research hypotheses, and 5) definition of terms.

Background of the Study

Research studies have shown that the skill of effectively programming a computer requires the ability to integrate several types of knowledge such as syntactic knowledge, semantic knowledge, and problem solving skills (Linn & Dalbey, 1985; Pea, 1986; Sleeman et al., 1986; Soloway, 1985). The task of computer programming requires the integration and implementation of complex cognitive processes. Thus, the design of computer programming instruction needs to be based on learning theories that suggest how a person's cognitive processes are developed and operate (Anderson, 1980, 1982, 1983; Bitter & Lu, 1988; Cavaiani, 1989; Chesson, 1992; Webb, 1984). Based on developments in learning theory, research on teaching and learning computer programming suggests that effective instruction in computer programming should be aimed at improving students' mental model of computer
concepts, overcoming students' inert knowledge, and developing students' metacognitive learning of computer programming (Bayman & Mayer, 1983, 1988; Du Boulay, 1986; Galloway, 1990; Linn & Clancy, 1992; Joni & Soloway, 1986; Mayer, 1987; Pea, 1986; Perkins et al., 1986; Segal et al., 1992; Sleeman et al., 1986; Soloway et al., 1983; Volet and Lund, 1994). Commonly cited in the research literature about effective computer programming instruction, these three factors related to three broader issues of student learning: conceptual models of computer programming instruction, fragile knowledge, and problem-solving instruction. A description of these broader issues is presented in the following sections.

**Conceptual models of computer programming instruction**

A common finding among research studies is that it is difficult to fully develop students' programming knowledge through traditional computer programming instruction (Cope & Walsh, 1990; Dalbey & Linn, 1986; Linn & Dalbey, 1985; Nutter & Hassall, 1985; Oliver, 1993). Research on teaching and learning computer programming indicates that novice programmers have several difficulties conceptualizing computer programming concepts such as looping, variables of iteration, and manipulating values in computer memory (Bayman & Mayer, 1983, 1988; Du Boulay, 1986; Galloway, 1990; Joni & Soloway, 1986; Mayer, 1987; Pea, 1986; Perkins et al., 1986; Segal et al., 1992; Sleeman et al., 1986; Soloway et al., 1983).

The difficulties novice programmers encounter may be due to a lack of understanding about computers and programming language statements (Bayman & Mayer, 1983). Bayman
& Mayer (1983) have suggested that when teaching computer programming, students first need to be introduced to a conceptual model of the operations of computer language statements to enhance their mental models of computing concepts. Research on computer programming instruction suggests that providing students with conceptual models of the phenomena being studied will enhance their mental models and may reduce their computer programming misconceptions (Bayman & Mayer, 1983, 1988; Mayer, 1987). According to Bayman and Mayer (1988), appropriate mental models in computer programming include a knowledge of computer operations, the order of execution of computer language statements, computer memory features, and computer looping procedures.

Several empirical studies indicate that various conceptual models have been successful in enhancing novice programmers’ mental models of the operations of computer language statements, design skills, memory features, and looping concepts and in improving their achievement in computer programming (Bayman & Mayer, 1988; Dalbey & Linn, 1986; Hooper & Thomas, 1990; Shih & Allesi, 1993; Upah & Thomas, 1993). Although the use of computer programming instruction based on conceptual models was successful in helping students improve their computer programming, this approach was not fully successful in helping students use or integrate their programming skills to solve complex programming problems. That is, novice programmers were not fully able to use their programming knowledge to solve programming problems.
Fragile Knowledge

In a clinical study designed to examine students' ability to integrate their programming knowledge to solve programming problems, Perkins et al. (1985) found that novice programmers have fragile knowledge. Perkins et al. referred to knowledge that was incomplete or unstable as fragile knowledge; they cited four types of such knowledge: partial knowledge, inert knowledge, misplaced knowledge, and conglomerated knowledge. Partial knowledge is "... knowledge [that] the student has not retained or even never learned." Inert knowledge "... refers to situations where the student fails to retrieve command knowledge but in fact possesses it." Misplaced knowledge "... designates circumstances where a student imports command structures appropriate to some contexts into a line of code where they do not belong." Conglomerated knowledge "... signifies situations where a student produces code that jams together several disparate elements in a syntactically or semantically anomalous way in an attempt to provide the computer with the information it needs" (p. 5).

This lack of a complete knowledge base among students may be due to the extensive focus given to syntactic and semantic knowledge during computer programming instruction (Linn & Clancy, 1992). Through a series of case studies, Linn and Clancy (1992) reported that novice programmers often organize their programming knowledge in terms of language syntax and engage in random trial and error tasks to design a program to solve problems. In contrast, expert programmers "... often attempt to organize their knowledge of programming in large conceptual structures rather than syntax [and] have complex knowledge structures involving algorithms and associated information" (p. 122). Thus, Perkins et al. (1985) and Linn and Clancy (1992) suggested that an explicit planning and design strategy, such as
introducing ways to solve a problem, in computer programming instruction may help students integrate their programming knowledge and skills to solve programming problems. Moreover, they reasoned that helping students think about their learning, in general, and learn to learn computer programming, specifically, may be useful in computer programming instruction.

**Problem-solving oriented instruction**

Incorporating explicit planning and design strategies as aids in learning, Volet and Lund (1994) used metacognitive instruction to help students integrate their programming knowledge in introductory computer programming courses. In their study, metacognitive instruction referred to an instructional package designed to encourage students to develop a metacognitive strategy relevant to computer programming via interactive teaching. A five-step metacognitive strategy was introduced to guide students' program planning processes: (1) problem definition; (2) algorithm development; (3) flowchart or pseudocode implementation; (4) coding; and (5) execution of the code, debugging errors and program improvement.

In the study, the researchers compared metacognitive instruction with traditional instruction in an introductory computer programming course for college students. The course assessment was based on two assignments, one mid-semester test, a major project, and a final examination. Achievement data consisted of retention rates, performance marks in the introductory course, and enrollment and overall performance in the subsequent more advanced computing course. Performance in the introductory course was based on the overall course mark and the scores on the final examination. The final examination consisted
of three parts. Both part one and two of the examination assessed knowledge of general syntax, programming concepts, and functions and procedures. Part three of the examination involved a practical exercise requiring students to apply their computing knowledge to solve a moderately complex programming problem. The results indicated that Volet and Lund’s metacognitive instruction in introductory computer programming was more effective than traditional instruction on student retention rates and programming achievement.

According to Pressley and McCormick (1995), metacognition refers to “... knowledge about and awareness of one’s thinking” (p.2). Thus, although Volet and Lund (1994) labeled their approach as metacognitive instruction, this researcher considered it to be more of a problem-solving instructional approach. That is, because Volet and Lund did not explicitly direct or encourage the students to think about their thinking of how to solve programming problems, their approach was more of a problem-solving instructional model than a metacognitive approach.

The Volet and Lund study illustrated the potential of problem-solving oriented instructional approaches to help students integrate their programming knowledge and skills and improve their learning. When computer programming instruction emphasized problem-solving, novice programmers gave more attention to design-oriented tasks. Moreover, they tended to process, implement, and revise their programming strategies using the programming language as a tool. Furthermore, when problem-solving oriented instruction is directed toward a real-world situation, the likelihood that the learning experience will be viewed as more authentic increases and can result in more successful transfer from the school setting and the workplace (Spiro et al., 1992).
Using a constructivist-based learning environment in real-world situations, Black et al. (1994) developed two graphic computer simulations (Parkside & Guestwear) to help high and middle school students learn how business organizations and financial systems work. The purpose of the simulations was to anchor learning in an authentic environment to help students learn about hotel and clothing manufacturer management. The researchers employed six principles of constructivist design in the simulations to simulate managing a hotel and a clothing manufacturer. The six principles were: generating as much knowledge as possible in real situations, anchoring activities in authentic situations, using cognitive apprenticeship models, situating knowledge in multiple contexts, creating environments to learn in multiple perspectives, and collaborating learning with knowledge construction. The results of this study indicated that students learned from the simulations and they improved their higher-level thinking skills as indicated by their written answers on the essay exam.

When learning from simulations such as Parkside, students construct their own knowledge through active participation in an authentic environment. They not only build strong mental models of the domain content, but also anchor their knowledge to decide when and where to use it. Thus, integrating constructivist principles into simulation design enables the use of problem-solving oriented approaches to learning.

In conclusion, it can be stated that learning computer programming is “...conceptual in nature,” involving an understanding of the language syntax, concepts, and principles of programming, and “...procedural in nature,” involving how to apply problem solving skills to solve a specific problem (Oliver, 1993, p. 299). The difficulties novice programmers encounter may be due to a lack of understanding about computers and programming language
statements (Bayman & Mayer, 1983). Research on computer programming instruction suggests that providing students with conceptual models of the phenomena being studied will enhance their mental models and may reduce their computer programming misconceptions (Bayman & Mayer, 1983, 1988; Mayer, 1987). Several empirical studies indicate that various conceptual models have been successful in enhancing novice programmers' mental models of computer concepts but not in strengthening their fragile programming knowledge (Bayman & Mayer, 1988; Dalbey & Linn, 1986; Hooper & Thomas, 1990; Upah & Thomas, 1993). To help students develop their fragile programming knowledge and integrate problem-solving skills, problem-solving oriented instruction enhanced student learning in computer programming (Volet & Lund, 1994). Programming knowledge represents hierarchical levels of a complex task. Instruction in computer programming should build the learners' mental models of domain content and focus on design strategies of problem solving using programming skills. Thus, the potential of combining conceptual models and problem-solving oriented instruction needs be explored to improve computer programming instruction.

Statement of the Problem

Empirical research studies indicated that learning computer programming is a complex process which requiring strong mental models of domain content and the integration of programming knowledge (Anderson, 1980, 1982, 1983; Bitter & Lu, 1988; Cavaiani, 1989; Chesson, 1992; Webb, 1984). Numerous studies have been conducted that examine the effectiveness of using conceptual models to improve student learning in computer programming (Bayman & Mayer, 1988; Dalbey & Linn, 1986; Hooper & Thomas, 1990; Shih
& Allesi, 1993; Upah & Thomas, 1993). The results suggest conceptual models are effective in enhancing student's mental models of basic computer concepts but have a little effect in improving integrated programming knowledge to solve programming problems. To enhance students' ability to integrate programming knowledge and problem-solving skills, research on the effectiveness of using problem-solving oriented instruction in computer programming has also been conducted (Volet & Lund, 1994). The results of which indicate that problem-solving oriented instruction enhanced students' ability to integrate programming knowledge and problem-solving skills. Instruction in computer programming needs to build the learners' mental models of computer domain content and focus on the design strategy of problem solving. Although the individual potential of each has been studied, little research exists that investigates a combination of using conceptual models and problem-solving oriented instruction to facilitate student learning in computer programming. Thus, studies that examine the combined effectiveness of conceptual models and problem-solving oriented instruction in computer programming needs to be undertaken.

Purpose of the Study

The purpose of this study was to examine the effectiveness of a combination of conceptual models and problem-solving oriented instruction in computer programming to enhance students' mental models of database concepts and enable them to integrate their knowledge to solve database programming problems. The problem analysis learning model combines a conceptual model and a holistic instructional approach in computer programming instruction.
The conceptual component of the problem analysis model includes several computer simulations of database concepts. The purpose of the conceptual component is to offer students an opportunity to manipulate data in computer simulations before formal instruction in order to help them construct their own knowledge of basic database concepts. The holistic component of the problem analysis model helps students integrate their programming of problem-solving knowledge to solve real-world problems with a complete view of their plan. This component includes a process that defines the problem, breaks it into sub-problems and acknowledges their relationships. Then, the students learn the computer language to solve the sub-problems, and finally solve the whole problem.

**Research Hypotheses**

The following research hypotheses were developed in the study:

*Hypothesis 1:* Students taught computer programming with the problem analysis model will develop more complete mental models of basic database concepts than students taught with traditional computer programming instruction as measured by their achievement on database assignments.

*Hypothesis 2.* Students taught computer programming with the problem analysis model will develop more complete mental models of managing multiple database files than students taught with traditional computer programming instruction as measured by their achievement on database assignments.
Hypothesis 3. There will be no significant achievement difference between participants in the problem analysis model and traditional instructional model on the general syntax section of the Database Examination.

Hypothesis 4. Students taught computer programming with the problem analysis model will perform better on the programming problems section of the Database Examination than students taught with traditional computer programming instruction.

Limitations of the Study

Several limitations existed in this study. They were:

1. The study was limited to the dBase language, and the subjects were inservice teachers in Taiwan.

2. The study was limited to selected laboratory and classroom activities.

3. The database materials covered in this study were related to dBase language programming with practical applications.

Definition of Terms

BASIC language - A computer language. The word BASIC is an acronym for Beginners All-purpose Symbolic Instruction Code.

dBase language - A computer language. The word dBase literally means database. dBase language is useful for creating and managing database files.

Declarative knowledge - Knowledge that consists of facts, concepts, or principles.

Procedural knowledge - Knowledge that consists of how to complete a task.
Mental model - Refers to an internal representation in one’s mind to understand the outside environment. That is, a person’s understanding of the environment.

Conceptual model - Instruction or a system that provides an appropriate representation of the state and relationship of the content domain. Typically, used to develop mental models.

Situated learning - Learning that takes place in a specific, well-defined context used to facilitate the development of usable knowledge.

Problem analysis model - A learning model comprised of a conceptual mode to enhance student mental models of basic database concepts and a holistic instructional approach to guide student learning about database problems.

Holistic instructional approach - A four-step procedure for learning about programming problems. The four steps are: problem introduction, problem diagnosis, learning activities, and learning assignments.

Summary

Research on computer programming suggests that novice programmers possess inert knowledge when trying to solve programming problems. Moreover, research on teaching and learning computer programming indicates that offering appropriate conceptual models of computer programming concepts to novice programmers enhances their mental models and reduces their misconceptions in computer programming. A significant amount of research has identified that through traditional computer programming instruction it is difficult to fully develop students’ programming knowledge (Cope & Walsh, 1990; Dalbey & Linn, 1986; Linn & Dalbey, 1985; Nutter & Hassall, 1985; Oliver, 1993). The purpose of this study was to
examine the effectiveness of a problem analysis learning model of computer programming in helping novice programmers integrate syntax and semantic knowledge and use a programming language to solve problems.

The problem analysis learning model combines a conceptual model and a holistic instructional approach for computer programming instruction. The conceptual component of the problem analysis model includes several computer simulations of database concepts. The purpose of the conceptual component of the problem analysis model is to offer students an opportunity to manipulate data in the computer simulations before formal instruction in order to help them construct their own knowledge of basic database concepts. The purpose of the holistic component of the problem analysis learning model is to help students integrate their programming knowledge to solving database problems. The holistic approach includes a four-step process that consists of problem introduction, problem diagnosis, learning activities, and database assignments.
CHAPTER II. REVIEW OF LITERATURE

Schwartz et al. (1989) indicated that computer programming is "... precision-intensive," requiring special care with details; "... problem-solving intensive," resolving programming problems; and "... design-intensive," constructing a whole process to complete certain jobs (p. 264). Moreover, empirical studies have shown that the ability to program a computer is the result of many cognitive processes (Bitter & Lu, 1988; Cavaiani, 1989; Chesson, 1992; Webb, 1984). To enhance student learning in computer programming, instructional strategies based on theories of student learning and cognitive psychology need to be developed (Anderson, 1983, 1987, 1988; Dayman & Mayer, 1983, 1988; Mayer, 1987).

To effectively teach computer programming, it is necessary to understand how learning occurs. Historically, approaches to computer programming instruction were based on learning theories such as Cognitive Information Processing (CIP) and ACT* theory. In the CIP model, the mind is viewed as a central information processor to process information between input from the environment and output from the individual. The ACT* theory describes forms of knowledge and the relationship of knowledge to structures of the mind. In both the CIP and ACT* models, learning is exhibited through the actions or behaviors of the learner. Moreover, these models suggest that the way to teach computer programming is to separately teach language syntax, semantics, and problem solving. That is, simplify and regularize the content and transfer it to students (A detailed description CIP and ACT theories appear in Appendix A).
Cognitive psychology provides several models of learning that suggest knowledge acquisition is an on-going, internal building process. Constructivism is one such model that provides a foundation for developing new approaches to computer programming instruction; it is discussed in this section.

**Constructivism**

An area of cognitive psychology that has important implications for computer programming instruction is constructivism. Although significantly different from CIP and the ACT theory, constructivism, situated cognition, and metacognition provide insight into how knowledge is anchored in context situations. From the constructivist view, learning is “... a constructive process in which the learner is building an internal representation of knowledge, a personal interpretation of experience” (Bendar et al., 1992, p. 21). That is, each individual constructs his or her own knowledge through interactions between existing knowledge and personal experience with the outside world during the learning process. Constructivists claim that meaning or knowledge does not exist in the world independently of learners, rather it is imposed on the world by learners (Duffy & Jonassen, 1992). Each individual has his or her own experiences from which to view the real world. “Each experience with an idea - and the environment of which that idea is a part - become part of the meaning of that idea” (Duffy & Jonassen, 1992, p. 4). Brown et al. (1989) indicated that “... knowledge is not separable from the actions that give rise to it nor from the culture in which those actions occur” (Pressely & McCormick, 1995, p. 80). That is, the acquisition of knowledge cannot be separated from content and actions. “It is not possible to isolate units of information or make
a priori assumptions of how the information will be used. Facts are not simply facts to be remembered in isolation" (Bendar et al., 1992, p. 23).

Spiro et al. (1992) also indicated that context is an integral part of meaning. They also suggest that several contexts of the same content should be presented to help learners construct multiple understanding of the content. If learners focus only on the critical features of a concept, they will have a limited understanding. Furthermore, when learners focus entirely on the critical features of a concept, they are not working with the concept in a complex environment, and they are not experiencing the complex interrelationships in that environment to determine how and when the concept is used.

In skill acquisition, Cunningham (1992) emphasized that skills cannot be considered independently of the problems to which they are applied. Learning a particular sub-skill means using it effectively in solving problems. Therefore, the goal of instruction is not to ensure that individuals know particular facts, but rather, the goal of instruction is to provide opportunities or learning environments to let individuals construct multiple interpretations of those facts.

Perkins (1992) indicated that an active learner is not just an active processor of information, but more importantly, the learner manipulates, interprets, extrapolates and assesses the information. When learning in context, learners can actively interact with their environment to develop usable knowledge. That is, when learning in a context environment, learners may construct their knowledge in a meaningful situation. Thus, situated knowledge is not “... residing in the mind but spread across the mind and the environment” (Pressely & McCormick, 1995, p. 81). That is, knowledge is not static in one's brain; through the
interaction of mind and environment, much of one's knowing is situated. "Constructivists focus on the process of knowledge construction and the development of reflexive awareness of that process" (Bednar et al., 1992, p.24). Therefore, the goal of instruction is to provide experiences from which the learner can build meaningful perceptions and to help the learner develop strategies and processes for building multiple perceptions. (Spiro et al., 1992; Cognition and Technology Group, 1992). Thus,

Instruction should provide contexts and assistance that will aid the individual in making sense of the environment as it is encountered. A [learning] plan is one part of that sense making, but plans must be constructed, tested, and revised as a function of the particular encounters in the environment. (Duffy & Jonassen, 1922, p. 5)

Learning computer programming may be regarded as the acquisition of different type of knowledge (syntax, semantic, and problem-solving skills). However, it is important to combine syntactic, semantic, and problem-solving knowledge during the learning of computer programming. Constructivism suggests new ways of teaching and learning computer programming. When applying constructivist ideas in computer programming instruction, it is necessary to understand the difficulties of novice programmers in order to design instruction for learners to improve their learning. The difficulties of novice programmers are discussed next.

**The Difficulties of Novice Programmers**

Teaching and learning computer programming is a difficult task for both teachers and students. How to effectively and efficiently improve student learning in computer programming has been the subject of many research studies over the past two decades.
Empirical studies have identified several programming problems of novices, such as basic computer concept problems, program writing style problems, and language-independent conceptual problems.

After studying bugs (i.e. errors in computer programs) and misunderstandings in the minds of the novice programmers, Soloway et al. (1983) indicated that looping strategies and the uses of variables are two key factors that affect novice learning of programming. Sleeman et al. (1986) also reported that, when learning Pascal language, high school students had two types of errors: "(1) those due to lack of attention or knowledge; and (2) those caused by the interaction of the student's knowledge of the formally defined domain with his or her common sense knowledge" (p. 21).

By studying the difficulties encountered when learning to program computers, Du Boulay (1986) identified five general areas of problems for novice programmers. The five problem areas were: (1) basic orientation (finding out what programming is for); (2) understanding the general properties of the machine; (3) formal language notation; (4) standard structures; and (5) mastering the pragmatics of programming. Du Boulay also noted that commonly there are three types of mistakes made by novice programmers: (1) misapplication of an analogy; (2) over-generalizations of language statements; and (3) inexpert handling of complexity in general, and interactions in particular. Furthermore, Joni & Soloway (1986) indicated that a poorly constructed working code is often produced by novice programmers. That is, novice programmers tend not to focus on the principle of program readability.
Pea (1986) demonstrated that there are language-independent conceptual bugs in novice programming. These include: a) parallelism bugs, b) intentionality bugs, and c) egocentrism bugs. Pea described parallelism bugs as the conception in the mind of the novice that “. . . different lines in a program can be active or somehow known by the computer at the same time” (p. 27). Intentionality bugs occur when “. . . the student attributes goal directness or farsightedness to the program and “goes beyond the information given” in the lines of programming code” (p. 29). Egocentrism bugs take place when “. . . students assume that there is more of their meaning for what they want to accomplish in the program than is actually present in the code they have written” (p. 30).

In addition to language-independent conceptual bugs, several empirical studies have identified computer operation and language structure misconceptions that novice programmers possess. Putnam et al. (1986) found that high school students in beginning BASIC programming courses had eight areas of misconceptions related to BASIC syntax: assignment statements, print statements, read statements, variables, loop construction, if statements, other flow control difficulties, and tracing and debugging. These misconceptions were due mainly to a lack of understanding of the meaning of basic language statements.

Galloway (1990) also found that, in a computer literacy course conducted for preservice teachers, the participants had fundamental computing misconceptions and had particular difficulty with the concepts and characteristics of data and files in a computer. Furthermore, in studying students’ ability to learn ALGOL 68 computer language, Segal et al. (1992) indicated that students had major difficulties using the semicolon which is the sequencing operator of the programming language. Segal et al. argued that this difficulty was
due to the fact that students did not immediately understand a specific rule of syntax. They claimed that "... the learner's developing understanding of the nature of the terms used in defining a syntax rule may play a crucial role in understanding that rule" (p. 151).

In summary, the difficulties encountered by novice programmers included language-independent conceptual bugs, language-dependent learning bugs, and basic computing misconceptions. These difficulties may be due to a lack of correct understanding about computers and programming language statements (Bayman & Mayer, 1983). Thus, Bayman & Mayer (1983) suggested that students need to be introduced to a conceptual model showing the variable locations on the computer and verbal and visual descriptions of the variables transactions for each statement.

Using Conceptual Models of Computer Operations in Computer Programming Instruction

Using a conceptual model showing the variable locations on the computer and detail descriptions of variable transactions for each computer language statement, may enhance novice programmers' mental models of computer operations. A mental model refers to "... a knowledge construct in the mind that represents a person's conceptual understanding of the domain" (Shih, 1991, p. 19). Pressley and McCormick (1995) indicated:

A development in mental model theory is the possibility that mental models are not entirely mental but rather often involve interactions between a partially complete internal representation and an environment that fills in the gaps in the mental model. (p. 79)

Therefore, by providing appropriate conceptual models to fills in the gaps of mental models, more complete mental models can be developed (Gentner & Stevens, 1983; Payne,
Research on computer programming instruction suggests that providing students with conceptual models of the phenomena being studied will enhance their mental models and may reduce their misconceptions in computer programming (Bayman & Mayer, 1983, 1988; Mayer, 1987). According to Bayman and Mayer (1988), the appropriate mental models in computer programming include a knowledge of computer operations, order of execution of computer language statements, computer memory features, and computer looping procedures.

Bayman and Mayer (1988) examined the effect of using English transactions as conceptual models to teach BASIC computer programming. In their study, Bayman and Mayer described each BASIC statement as a list of transactions that explain how the statement will be executed in the computer.

For example, the statement ‘70 LET A=B+1’ includes the following transactions:
1. Copy the number in memory space B to the work space.
2. Add 1 to the number in the work space.
3. Erase the number in memory space A.
4. Put the number from the work space into memory space A.
5. Move the program pointer to the next statement in the program. (p. 292)

Based on these transactions, the researchers added additional information to the standard manual, entitled *BASIC in Six Hours* (1980). A total of five different instructional manuals were developed. Each manual was more detailed and included pictorial information of BASIC language concepts. Each of the five manuals corresponded to one of the five treatments groups: standard (control group), summary-transaction, transaction, diagram and transaction-diagram.
The standard manual was a self-instructional, mastery manual that consisted of two lessons: statement execution in the immediate mode and introductory program preparation and simple programs. The other four manuals had progressively more detailed explanations of the BASIC language statements. The summary transaction manual summarized the major operations carried out in the computer during statement execution. The transaction manual listed the operations carried out by the computer during statement execution. The diagram manual presented a pictorial representation of the state of the computer before and after statement execution, and the transaction-diagram manual included the same content as both the transaction and the diagram manuals.

A total of 95 college student novice programmers learned BASIC by studying one of the five manuals. After passing two mastery tests of the content in their manual, the subjects took four posttests: programming, fact-retention, procedure-specification, and diagram-specification. Procedure- and diagram-specification tests were used to assess conceptual knowledge. The results indicated that the students in the four conceptual treatment groups had fewer misconceptions of computer operation and language statements and performed better on programming skill items than the students who used the standard manual (control group). There was no difference among the groups in their syntactic knowledge of BASIC. The researchers concluded that the added description model was successful in enhancing students' mental models of programming in BASIC.

Examining the development of students design skills in computer programming, Dalbey and Linn (1986) used a conceptual-based computer learning environment, called "Spider World", to assist student learning. In the Spider World learning environment,
programming language features were minimized, easily understandable templates were provided, and the use of procedural skills were encouraged. In Spider World, the users command a hypothetical robot to create colored patterns on the computer screen to develop and refine their computer programming templates.

The purpose of the study was to use Spider World as a tool to help students practice planning programs and test their programs without spending time learning detailed language features. Dalbey and Linn implemented their Spider World study in a 12-week introductory computer programming course. In the experimental group, Spider World was taught for the first three weeks of the course, then during the remaining nine weeks students received traditional BASIC instruction. The control group received 12 weeks of traditional BASIC instruction.

After the first three weeks of the study, students were given a test designed to measure their ability to use program templates in BASIC. The results of the test indicated that students in the Spider World group performed better than those in the control group. The researchers reported that instruction incorporating Spider World was preferable to traditional BASIC instruction alone for learning design templates.

At the end of the 12-week study, comprehension, reformulation and design skills were assessed. Comprehension was defined as a student’s understanding of the overall meaning of programming language features. Reformulation was defined as the application of skills to new problems in a known domain, and design was defined as the application of skills to a new situation in a new domain. The final assessment of these three skills showed that the two treatment groups had the same level of programming knowledge when judged by these
standards. However, all of the students' scores were much lower than the researchers had expected. The researchers concluded that, although the Spider World model was useful for acquiring knowledge of design templates, additional instructional factors were needed to expand student learning. The researchers suggested that additional factors may include suitable learning materials, teacher training, and instructional methods.

To investigate student learning of Pascal programming, Hooper and Thomas (1990) proposed a manipulative model of computer memory called MEMOPS. The purpose of MEMOPS was to help students develop their conceptual understanding of basic memory functions. The study consisted of two groups of college students who were learning Pascal programming. In the experimental group, MEMOPS was used before formal instruction to help the students construct their knowledge about the functions of computer memory such as the destructive property of memory, the use of additional memory to preserve information, and the syntax rules associated with programming statements. In the control group, a computer-based geometric puzzle was used before formal instruction to solve geometric problems. After completing the learning activities, the students in the both groups attended the traditional lectures and completed the same assignments. At the end of the study, the researchers found that students in the experimental group had fewer errors and constructed more complex algorithms in Pascal programming than the students in the control group. That is, students constructed more complete mental models of computer memory through the manipulation of the computer memory simulation than students in the traditional learning environment.
In another study about student learning in Pascal, Upah and Thomas (1993) investigated the effect of two computer simulations (LOOPS & LOOPOPS) on learning programming loops (WHILE-DO and REPEAT-UNTIL). The LOOPS simulation was designed to offer students an opportunity to manipulate specific computer codes in order to understand looping concepts before formal instruction. The LOOPOPS simulation was designed to allow students to practice and test their understanding of looping concepts with Pascal programming statements. The goal of this practice was to enhance the students’ ability to integrate abstract looping concepts with concrete computer codes. There were two groups in the study: a control group and an experimental group. The control group treatment consisted of a computer-based tutorial (prior to formal instruction), formal instruction, and paper-and-pencil exercises. The experimental treatment consisted of the LOOPS simulation (prior to formal instruction), formal instruction, and the LOOPOPS simulation (after formal instruction).

Both groups completed two post-test problems and one final exam. One of the post-test problems required the tracing of nested loops while the other required subjects to translate looping codes from REPEAT and WHILE to FOR commands. The final exam included three looping programming problems.

The researchers found that LOOPS and LOOPOPS simulations improved students’ ability to solve new looping problems when compared to the tutorial approach used with the control group. The results indicated that the experimental group did better than the control group in code translation. There was no difference in code tracing. Upah and Thomas
concluded that "... the use of the dynamic models permits students to attain a deeper understanding of new and difficult concepts" (p. 409).

In a clinical study about programming knowledge, Perkins et al. (1985) employed a scaffolded interview to investigate the problems encountered by novice programmers when learning BASIC language. In the study, twenty high school students enrolled in the second semester of a year-long BASIC course were interviewed. A sequence of eight programming problems (ranging from easy to difficult) on the FOR-NEXT loop command was given to each student. The investigators interviewed the students and asked questions to track their thinking. When the students found it difficult to answer the questions, the investigators provided graduated levels of assistance. First, they offered general prompts to provoke the student’s strategic thinking. Next, hints, leading questions, and bits of information were proposed. Finally, exact solutions were presented to help the students solve immediate dilemmas.

Working one-on-one with the students, data collected during the sessions included notes taken by the investigator, code written by each student and transcribed by the investigator, and an audiotape of conversations. A qualitative analysis was conducted to interpret the data. The researchers reported that student knowledge does not simply mean to know or not to know something but to possess “fragile knowledge”. Perkins et al. referred to knowledge that was incomplete or unstable as fragile knowledge. They interpreted the students’ difficulties in solving the FOR-NEXT loop problems as being one (or more) of four types of fragile knowledge of BASIC language commands. The four types of fragile knowledge were:
Partial knowledge is the straightforward case of an impasse due to knowledge the student has not retained or even never learned, as revealed by clinical probes failing to reveal signs of the knowledge. Inert knowledge refers to situations where the student fails to retrieve command knowledge but in fact possesses it, as revealed by a clinical probe. Misplaced knowledge designates circumstances where a student imports commands structures appropriate to some contexts into a line of code where they do not belong. Conglomerated knowledge signifies situations where a student produces code that jams together several disparate elements in a syntactically or semantically anomalous way in an attempt to provide the computer with the information it needs. (Perkins et al., 1985, p. 5)

Perkins reported that the four types of fragile knowledge effected the students’ ability to solve programming problems. The result of the study indicated that students may not organize their existing programming knowledge in a meaning way or a complete view.

Through case studies of programming problems of students, Linn and Clancy (1992) found that novice programmers often organize their programming knowledge in terms of language syntax and engage in random trial and error tasks to design a program. In contrast, expert programmers “... often attempt to organize their knowledge of programming in large conceptual structures rather than syntax [and] have complex knowledge structures involving algorithms and associated information” (p. 122).

In summary, several empirical studies indicate that various conceptual models have been successful in enhancing novice programmers’ mental models of the operations of computer language statements, design skill, memory features, and looping concepts and in improving their achievement in computer programming (Bayman & Mayer, 1988; Dalbey & Linn, 1986; Hooper & Thomas, 1990; Shih & Alesi, 1993; Upah & Thomas, 1993). The use of computer programming instruction based on conceptual models was successful in helping students improve their computer programming. However, the results of several conceptual
models (Bayman & Mayer, 1988; Dalbey & Linn, 1986; Hooper & Thomas, 1990; Upah & Thomas, 1993) indicated that using conceptual models alone did not help students overcome fragile knowledge or fully develop their programming skills to solve complex programming problems. After conceptual-based instruction novice programmers had fragile programming knowledge and organized their programming knowledge according to language syntax not as conceptual structures. Thus, to enhance students' ability to organize their programming knowledge, Perkins et al. (1985) and Linn and Clancy (1992) suggested that “... direct teaching or indirect encouragement of strategic self-prompting and other tactics should help students to learn to program better and increase the likelihood of transfer from programming as well” (p. 30).

**Problem-solving oriented instruction in Computer Programming**

Focusing on design strategies, Volet and Lund (1994) investigated the effect of using metacognitive instruction in introductory computer programming courses. In Volet and Lund’s study, metacognitive instruction referred to instructional approaches aimed at inducing students to develop a metacognitive strategy relevant for computer programming. Volet and Lund introduced a five-step problem-solving strategy to guide students' program planning processes. The five-step strategy consisted of:

1) problem definition - analysis of the problem for its suitability for computer solution, identification of data input, output, and specific conditions;
2) algorithm development - a step by step procedure for solving the problem in plain English;
3) conversion of the algorithm into the rigid formalism of a flowchart (pictorial scheme) or pseudocode representation;
4) coding - using the formula algorithm as a guide to writing the program in a specific programming language; and
5) execution of the code, debugging errors and program improvement. 
(Volet & Lund, 1994, p. 304-305)

In each of the two treatment groups, twenty-eight subjects were matched by computing background, gender and age. The computer course was a 13 week computer science college course for first year students. The programming language taught in the course was Turbo BASIC. Each week of the study, three one-hour lectures were given to the whole class and one two-hour laboratory session was conducted. The intervention for the students in the experimental group was conducted during tutorial time (1 hour per week) by the regular tutor. The five-step metacognitive strategy was introduced in the first tutorial session, and students were encouraged to use that strategy in all of their programming exercises throughout the semester. The interactive teaching approach used during tutorial time involved a large amount of tutor-group verbal interactions that consisted of modeling and coaching of problem solutions for weekly exercises using the five-step planning strategy.

During their weekly tutorial time, students in the control group were required to work on their weekly exercises while the tutors acted as consultants, with occasional group explanations. These students had been introduced to algorithm development in the lectures but were not taught any strategies to facilitate their program planning. The computer programming content covered in the tutorials and the time the tutors spent with the students were the same for both the experimental and control groups.

The course assessment was based on two assignments, one mid-semester test, a major project, and a final examination. Data related to achievement consisted of retention rates, performance marks in the introductory course, and enrollment and overall performance in the
subsequent more advanced computing course. Performance in the introductory course was based on overall course marks and the scores on the final examination. The final examination consisted of three parts. Parts one and two of the examination assessed student knowledge of general syntax, programming concepts, and functions and procedures. Part three of the examination involved a practical exercise requiring students to apply their computing knowledge to solve a moderately complex programming problem.

According to Pressley and McCormick (1995), metacognition refers to “...knowledge about and awareness of one’s thinking” (p.2). Thus, although Volet and Lund (1994) labeled their approach as metacognitive instruction, this researcher considered it to be more of a problem-solving instructional approach. That is, because Volet and Lund did not explicitly direct or encourage the students to think about their thinking about solving their programming problems, their approach was more of a problem-solving instructional model than a metacognitive instructional model.

The results of the study indicated that problem-solving oriented instruction in introductory computer programming significantly effected the students’ retention rate, and short-term and long-term achievement. The researchers concluded that their instructional approach was a better explanatory construct for students’ computing performance than traditional personal variables such as background knowledge, program major, gender, or age. Moreover, the problem-solving oriented instructional method involving tutor-group interactions was well suited for improving process-oriented instruction.

The process used by Volet and Lund (1994) carried out the suggestions from the Soloway et al. (1983, 1985, 1988) studies. Soloway et al. (1983) suggested that computer
programming instruction should focus less on the syntax and semantics of a programming language. "More emphasis should be placed on teaching the plans and strategies relevant to programming than is done now" (Soloway et al., 1983, p. 52). Researchers have found that when computer programming instruction focuses on explicit planning strategies, novice programmers tend to pay more attention to design-oriented tasks. They process, implement, and revise their strategy with the help of the programming language tools. Moreover, if the task is oriented toward a real-world situation, the learning experience is likely to be more authentic and transfer easily between the school setting and the workplace.

Black et al. (1994) proposed two graphic computer simulations (Parkside and Guestwear) using a constructivist design to help middle and high school students understand how business organizations and financial systems work. In their simulations, Black et al. employed six principles of constructivist design to simulate managing a hotel and a clothing manufacturer. The six principles were:

1. Set the stage but have the students generate the knowledge for themselves as much as possible (Jacoby, 1978; Black, Carroll and McGuigan, 1987)
2. Anchor the knowledge in authentic situations and activities (Cognition and Technology Group at Vanderbilt, 1990)
3. Use the cognitive apprenticeship methods of modeling, scaffolding, fading and coaching to convey how to construct knowledge in authentic situations and activities (Collins, Brown and Newman, 1990)
4. Situate knowledge in multiple contexts to prepare for appropriate transfer to new contexts (Gick and Holyoak, 1983)
5. Create cognitive flexibility by ensuring that all knowledge is seen from multiple perspectives (Spiro, Feltovich, Jacobson and Coulson, 1991)
6. Have the students collaborate in knowledge construction (Johnson and Johnson, 1975) (cited in Black et al., 1994, p. 64)
For example, as users begin the Parkside simulation, they can sit in their simulated office or wander around the simulated hotel (principle 1). Then, various problems arise through interacting with simulated people, reading memos, answering phone calls, etc. The learners have to gather information from a variety of information sources provided by the simulation to figure out how to deal with the problems. This information includes occupancy rates, income, customer satisfaction reports, newspaper articles, advice from staff, etc. All graphic displays and the underlying functional relationships of the simulation provide authentic situations and activities that anchor the material (principle 2); and the learners are given authentic business situations in which to make decisions. During the learners’ use of the Parkside simulation, the instructor serves as a facilitator to provide help and advice to the students (principle 3). The students work in groups to collaborate with each other in the simulated world (principle 4) and to make decisions through their discussion (principle 5). To enhance the user’s ability to apply their skills in a variety of situations, the simulation provides several contexts in which the user can develop and test their knowledge and skill (principle 6).

Black et al. (1994) used the Parkside simulation to help students learn hotel management. They employed a pre- and post-test design to examine the effectiveness of using the Parkside simulation. Sixteen subjects were involved in the study. Twelve subjects completed all of the activities of the study. The final test in the study covered several new business cases. The students were required to make a series of decisions about the business, acquire a new vocabulary for business and economics, and explain their reasoning for the decisions they made in essay questions. The results of this study indicated that students
learned from the simulations and they improved their higher-level thinking skills as indicated by their written answers on the essay exam. (Black et al., 1994).

When learning from simulations such as Parkside, students construct their own knowledge through active participation in an authentic environment (Black et al., 1994). Not only do they build strong mental models of the domain content, but they also anchor their knowledge about when and where to use it. Thus, integrating constructivist principles of learning into the design of simulation enables the use of a problem-solving oriented instructional approach.

The skill of computer programming requires the ability to integrate syntax, semantic, and problem solving knowledge. Historically, the task of teaching computer programming was a process of first helping students learn language syntax, then semantics, and finally problem-solving knowledge. Effective computer programming instruction must also help students combine and integrate syntactic, semantic and problem solving knowledge. That is, anchoring student learning of computer programming in a context environment is needed. Because computer programming is a unique instructional environment, understanding the role and implications of constructivism (especially situated cognition) may provide insight into how computer programming instruction can be best designed to bring about high levels of student learning.

Finally, learning computer programming is "... conceptual in nature," involving an understanding of the language syntax, concepts, and principles of programming, and "... procedural in nature," involving how to apply problem solving skills to solve a specific problem (Oliver, 1993, p. 299). Programming knowledge presents hierarchical levels of a
complex task. Instruction in computer programming should build the learners' mental models of domain content and focus the design strategy on problem solving. Thus, the potential of combining conceptual models, anchoring learners' learning in their previous experiences and real world situations, and problem-solving oriented instruction needs to be explored to improve computer programming instruction.
CHAPTER III. METHODOLOGY

In this chapter, a description of the methods and procedures used to investigate the research hypotheses of this study are presented. This chapter is organized into eleven sections. The eleven sections include the overview of the experimental research design, sample, treatments, Database Simulation, holistic instructional approach, instrumentation, variables of the study, procedures, hypotheses, data analysis, and a summary of the pilot study.

Overview of the Research Design

This study involved 100 inservice teachers enrolled in a basic computer programming workshop at ISST. Two workshops were conducted to accommodate the teachers. Each workshop consisted of 36 hours instruction over five days. Three subjects were covered in each workshop: BCC, Chinese word processing, and a database unit. The instructional schedules for the two workshops are shown in Appendix B.

Prior to conducting the study, a copy of the research methodology was submitted to the Human Subjects Review Committee at Iowa State University to ensure that the rights and welfare of the human subjects were adequately protected, confidentiality of data was maintained, and informed consent of participation was obtained from each inservice teacher. The Human Subjects Approval Form is shown in Appendix C.

The participants in the first workshop served as the control group and were taught using a traditional approach to computer programming instruction. The participants in the second workshop served as the experimental group and were taught using a problem analysis
model of computer programming instruction. Five assignments and a posttest were given to both the control group and the problem analysis group. The resulting achievement of the two groups were compared to examine the effective of the problem analysis model of instruction on student learning of computer programming.

Sample

The population for this study was inservice secondary school teachers in Taiwan. The sample selected from this population came from 15 counties and 6 cities in the Province of Taiwan. Through their county or city educational bureau, the subjects voluntarily registered for an introductory computer programming workshop and were randomly selected to participate in the study. The sample consisted of two or three inservice teachers from each county or city in the Province of Taiwan.

The control group was comprised of teachers who attended the first workshop. Of the 53 inservice teachers initially enrolled in the first workshop, 47 participants completed all of the learning activities. Participants in the second workshop comprised the experimental group. Of the 56 inservice teachers initially enrolled in the second workshop, 53 participants completed all the of learning activities.

The results of the demographic section of the Computing Survey indicated that out of the 47 inservice teachers in the control group, 13 were males (27.7%) and 34 were females (72.3%). Of the 53 inservice teachers in the problem analysis group, 21 were males (39.6%) and 32 were females (60.4%). The demographic data are shown in Tables 3.1. As illustrated
Table 3.1. Demographic information of the control and experimental group participants

<table>
<thead>
<tr>
<th>Variables</th>
<th>Level value</th>
<th>Control N</th>
<th>Control Percent</th>
<th>Experimental N</th>
<th>Experimental Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gender</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gender</td>
<td>Male</td>
<td>13</td>
<td>27.7</td>
<td>21</td>
<td>39.6</td>
</tr>
<tr>
<td></td>
<td>Female</td>
<td>34</td>
<td>72.3</td>
<td>32</td>
<td>60.4</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>47</td>
<td>100.0</td>
<td>53</td>
<td>100.0</td>
</tr>
<tr>
<td>Age</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Age 21 - 23 years</td>
<td>3</td>
<td>6.4</td>
<td>1</td>
<td>1.9</td>
<td></td>
</tr>
<tr>
<td>Age 24 - 31 years</td>
<td>22</td>
<td>46.8</td>
<td>16</td>
<td>30.1</td>
<td></td>
</tr>
<tr>
<td>Age 32 - 39 years</td>
<td>8</td>
<td>17.0</td>
<td>10</td>
<td>18.9</td>
<td></td>
</tr>
<tr>
<td>Age 40 or older</td>
<td>14</td>
<td>29.8</td>
<td>26</td>
<td>49.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>47</td>
<td>100.0</td>
<td>53</td>
<td>100.0</td>
</tr>
<tr>
<td>Teaching experience</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Teaching experience 1-5 years</td>
<td>24</td>
<td>51.1</td>
<td>20</td>
<td>37.7</td>
<td></td>
</tr>
<tr>
<td>Teaching experience 6-10 years</td>
<td>5</td>
<td>10.6</td>
<td>5</td>
<td>9.5</td>
<td></td>
</tr>
<tr>
<td>Teaching experience 11-15 years</td>
<td>5</td>
<td>10.6</td>
<td>4</td>
<td>7.5</td>
<td></td>
</tr>
<tr>
<td>Teaching experience 16 or more</td>
<td>13</td>
<td>27.7</td>
<td>24</td>
<td>45.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>47</td>
<td>100.0</td>
<td>53</td>
<td>100.0</td>
</tr>
<tr>
<td>Educational background</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Educational background Bachelors</td>
<td>33</td>
<td>70.2</td>
<td>29</td>
<td>54.7</td>
<td></td>
</tr>
<tr>
<td>Educational background Bachelors + 15 credits</td>
<td>10</td>
<td>21.3</td>
<td>20</td>
<td>37.7</td>
<td></td>
</tr>
<tr>
<td>Educational background Master</td>
<td>1</td>
<td>2.1</td>
<td>2</td>
<td>3.8</td>
<td></td>
</tr>
<tr>
<td>Educational background Other</td>
<td>3</td>
<td>6.4</td>
<td>2</td>
<td>3.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>47</td>
<td>100.0</td>
<td>53</td>
<td>100.0</td>
</tr>
<tr>
<td>Preferred working style on a project</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Preferred working style on a project Alone</td>
<td>27</td>
<td>57.4</td>
<td>28</td>
<td>52.8</td>
<td></td>
</tr>
<tr>
<td>Preferred working style on a project With small group</td>
<td>20</td>
<td>42.6</td>
<td>25</td>
<td>47.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>47</td>
<td>100.0</td>
<td>53</td>
<td>100.0</td>
</tr>
<tr>
<td>Working style when learning mathematics concepts</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Working style when learning mathematics concepts Alone</td>
<td>19</td>
<td>40.4</td>
<td>21</td>
<td>39.6</td>
<td></td>
</tr>
<tr>
<td>Working style when learning mathematics concepts With small group</td>
<td>28</td>
<td>59.6</td>
<td>31</td>
<td>58.5</td>
<td></td>
</tr>
<tr>
<td>Working style when learning mathematics concepts No opinion</td>
<td>0</td>
<td>0.0</td>
<td>1</td>
<td>1.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>47</td>
<td>100.0</td>
<td>53</td>
<td>100.0</td>
</tr>
<tr>
<td>Working style when solving mathematics problems</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Working style when solving mathematics problems Alone</td>
<td>25</td>
<td>53.2</td>
<td>25</td>
<td>47.2</td>
<td></td>
</tr>
<tr>
<td>Working style when solving mathematics problems With small group</td>
<td>22</td>
<td>46.8</td>
<td>27</td>
<td>50.9</td>
<td></td>
</tr>
<tr>
<td>Working style when solving mathematics problems No opinion</td>
<td>0</td>
<td>0.0</td>
<td>1</td>
<td>1.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>47</td>
<td>100.0</td>
<td>53</td>
<td>100.0</td>
</tr>
<tr>
<td>Prior computer course</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prior computer course No</td>
<td>26</td>
<td>55.3</td>
<td>29</td>
<td>54.7</td>
<td></td>
</tr>
<tr>
<td>Prior computer course Yes</td>
<td>21</td>
<td>44.7</td>
<td>23</td>
<td>43.4</td>
<td></td>
</tr>
<tr>
<td>Prior computer course No opinion</td>
<td>0</td>
<td>0.0</td>
<td>1</td>
<td>1.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>47</td>
<td>100.0</td>
<td>53</td>
<td>100.0</td>
</tr>
<tr>
<td>Prior computer experience</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prior computer experience No</td>
<td>5</td>
<td>10.6</td>
<td>10</td>
<td>18.9</td>
<td></td>
</tr>
<tr>
<td>Prior computer experience Some</td>
<td>39</td>
<td>83.0</td>
<td>40</td>
<td>75.5</td>
<td></td>
</tr>
<tr>
<td>Prior computer experience Much</td>
<td>3</td>
<td>6.4</td>
<td>3</td>
<td>5.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>47</td>
<td>100.0</td>
<td>53</td>
<td>100.0</td>
</tr>
<tr>
<td>Comfort level using a mouse</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Comfort level using a mouse Very comfortable</td>
<td>15</td>
<td>31.9</td>
<td>8</td>
<td>15.1</td>
<td></td>
</tr>
<tr>
<td>Comfort level using a mouse Somewhat comfortable</td>
<td>18</td>
<td>38.3</td>
<td>33</td>
<td>62.3</td>
<td></td>
</tr>
<tr>
<td>Comfort level using a mouse Somewhat uncomfortable</td>
<td>9</td>
<td>19.1</td>
<td>4</td>
<td>7.5</td>
<td></td>
</tr>
<tr>
<td>Comfort level using a mouse Very uncomfortable</td>
<td>1</td>
<td>2.1</td>
<td>1</td>
<td>1.9</td>
<td></td>
</tr>
<tr>
<td>Comfort level using a mouse No opinion</td>
<td>4</td>
<td>8.6</td>
<td>7</td>
<td>13.2</td>
<td></td>
</tr>
<tr>
<td>Comfort level using a mouse Total</td>
<td>47</td>
<td>100.0</td>
<td>53</td>
<td>100.0</td>
<td></td>
</tr>
</tbody>
</table>
in Tables 3.1, there were nearly twice as many females as males in both the control and problem analysis groups. By age, the largest group of the participants in control group were between the ages 24-31. In the problem analysis group, the largest group of participants were 40 or older. Most of the participants had taught 1-5 years (N=24 control group, N=20 problem analysis group) or 16 or more years (N=13 control group, N=24 problem analysis group). Most of the participants had a bachelor’s degree but no additional formal education. In both groups, the majority of the participants preferred to work independently on a project; however, when learning mathematics concepts, both groups preferred to work with a small group. Ninety four percent of the participants had little or no computer experience prior to the workshop.

**Treatments**

The two treatments in the study were the control group and the problem analysis group. The control group was taught using a traditional approach of computer programming instruction. The problem analysis group was taught using a problem analysis model of computer programming instruction. Three instructors taught in each workshop. BCC and Chinese word processing were taught by one instructor who is an experienced university professor. The database unit was taught by two instructors each of whom have taught similar database units for more than three years.

**Control group**

In this study, the control group learned BCC, Chinese word processing, and database concepts via traditional computer programming instruction. The BCC and Chinese word
processing units served as a general introduction to computer hardware and software. In the
database unit, a conventional instructional approach was used to teach database concepts and
dBase language programming. That is, instruction in database concepts and dBase computer
programming was accomplished by first teaching language syntax, then semantics, and finally
problem solving. The lesson plans appear in Appendix D.

Problem analysis group

In the problem analysis group, the instructional approach used in BCC and Chinese
word processing units was the same as that used with the control group. However, in the
database unit, a problem analysis instructional approach was used to teach database concepts
and dBase language programming. The problem analysis instructional approach included a
conceptual model and a holistic instructional approach. The conceptual model included
several database simulations. The holistic approach included a four-step teaching procedure.
That is, instruction about database concepts and dBase computer programming was
accomplished by first providing the subjects with a conceptual model (via Database
Simulation), then a holistic model of database concepts. Finally, the syntax and semantics of
dBase language programming were taught as tools to solve programming problems.

Database Simulation

Description of the Database Simulation

Over more than three years of personal observations, the researcher has noted that
when students learn to use or design a database, they often have difficulties understanding the
concepts of data, a field, and a record, and they often lack experience in organizing and using a database. Moreover, students have difficulty understanding the importance of categorizing data (which enables data to be used in a specific problem) and the inter-relationships within a relational database. Given these learning difficulties, the Database Simulation was designed to provide students the opportunity to manipulate data and to acquire knowledge and experience in data categorization, data processing, data sorting, and basic concepts of a relational database.

After the first version of the Database Simulation was developed by the researcher, it was field tested with two computer scientists over a two-month period. The content and the interface of the Database Simulation were modified based on the suggestions of the two computer scientists. During the developmental stage of the Database Simulation, several instructional technology graduate students provided suggestions from a user's point of view to enhance the user-friendly capabilities of the simulation.

Contents of the Database Simulation

Providing a situated environment in which to anchor students' experiences is an important factor in helping students construct their own knowledge base. The Database Simulation included three sections: (1) Clean Your Table, (2) Sort Your Data, and (3) Find Your Answers (see Figure 3.1).
Clean Your Table

In the first section, Clean Your Table, an office-like environment comprised of a table of disorganized cards is presented (Figure 3.2). The student’s task is to clean the table by categorizing the data written on the cards into three frames. The students must use their previous experiences to classify and categorize the data. It was intended that this Database Simulation activity would induce the students to construct their knowledge about data categorization.

Thinking about moving the cards on the table prompts students to develop rules to categorize and arrange the data. After the data are moved to frames, the students are asked to answer questions about the data. The purpose of the questions is to cause the students to reflect on the effectiveness and efficiency of their data categorization. Because the data in the frames are hidden, the students must go through each data card to answer the questions. This
exercise is intended to give the students a sense of raw data and organized data, and the importance of data categorization. The goal of this Database Simulation activity is for the students to become familiar with and think about concepts such as data categorization, the difference between raw data and organized data, basic ideas of data comparison, and the benefits of data summation.

**Sort Your Data**

After students experience the concept of raw data, organized data, and data categorization, the second section of the Database Simulation (Sort Your Data) offers two tables: one table contains data while the other is empty (Figure 3.3). Students can see data on
the table and are asked to sort it from the original table to the empty table. The purpose of this section of the Database Simulation is to provide the student with opportunities to understand the relationship between data and the concept of a record. For example, item 1 states: "Please sort the data on to the empty table according to last name in ascending order."

To complete this task, the student must compare all of the data in the field, titled 'last name', to make sorting decisions. Based on their decision, students then manipulate the data by using a mouse to "drag" the data on to the empty table. The data they drag and drop are manipulated into a record unit that includes "Last name", "First name", "Math score", "Science score", and "History Score". Therefore, students see and experience the relationship between a record and the individual fields that comprise a record. Through the interaction of data comparison and the sorting process, students can build an understanding of the relationship between a record, a field, and data in different fields.

Find Your Data

By manipulating data, students link their previous experiences to the basic concepts of data processing. In the third section of the Database Simulation, Find Your Data, students use a mouse to find answers to questions related to the database (Figure 3.4). For example, question 2 asks: "What is Steven's math score?" Students must follow a field name (Fname) to find the data "Steven" in a row of the database. Because the data are hidden, students must use a mouse to go through all of the rows in the Fname field until they discover the target data. The processing required by this relational database exercise is intended to help students understand the concept of a field and the meaning of a row. After finding the data "Steven",,
Figure 3.3. Sort your data Simulation

Figure 3.4. Find your data Simulation
students also must navigate through the columns of the database to search for the Math score data. This task is intended to enable students to construct the concept of two dimensions (row and column) of a relational database. Thus, through the third activity of the Database Simulation, students experience the meaning of a field, a row, and a column in a relational database which may help them construct their knowledge about a row and relate it to a relational database.

The role of the Database Simulation

The purpose of the Database Simulation used with the problem analysis group was to build students' mental models of basic database concepts such as raw data, organized data, data categorization, the relationship of a field and a record, the concept of a relational database, and the relationship between data of different fields. The Database Simulation was designed to offer a learning environment for students to anchor their previous experiences, learn about raw data, and manipulate a relational database to construct their knowledge about basic database concepts. The Database Simulation was used prior to formal instruction to build the learner's mental model of a database. In the Database Simulation, students were strongly encouraged to actively manipulate data as a method to implement their thinking. The workshop instructors acted as facilitators during simulation activities to guide students when they asked for help or to solve mechanical problems related to hardware and software.
holistic instructional approach

description

After members of the problem analysis group completed the Database Simulation, they received formal database instruction using a holistic instructional approach. The database unit included five sections: (1) Create a Database; (2) Data Processing; (3) Create a Student Database; (4) Managing Multiple Database Files; and (5) dBase Language Programming (The lesson plans used in the database unit appear in Appendix D). Each lesson in the holistic instructional approach followed a four-step procedure. The four steps were: problem introduction, problem diagnosis, learning activities, and database assignments.

four-step procedure of the holistic instructional approach

problem introduction

The first step of the holistic approach was to introduce problems or questions that would be solved in the learning activities. The purpose of this step was to offer an authentic environment to allow students to link their previous experience or knowledge to a problem situation. For example, in section 1 of the database unit (create a database), the students were given a school grade book and asked a specific question: “Who has the highest math score?” This step was intended to give students a whole view of what they were learning and help them reflect on their thinking during the next step of the holistic approach.

problem diagnosis

The second step of the holistic instructional approach was problem diagnosis. Participants were requested to design a method to solve the problems they were given in the
problem introduction step. Based on their personal experience and knowledge, the students designed their own methods to solve the problems. After designing a method to solve the problem, the students participated in whole class discussions to generalize some procedures to solve the previous problems. Students recorded their thinking processes on paper. Through whole-class discussions, they revised their ideas and generalized some procedures for future problems. For example, to answer the question “Who has the highest math score?” (problem introduction), the students suggested “using computers to create a student database to find the highest math score” and recorded their learning procedures as “what is a database?”, “how do I create a database?”, and “how do I use a database to find out the answer to the problem?” (problem diagnosis).

Learning activities

To address the procedures generated in the problem diagnosis step, the third step of the holistic instructional approach was learning activities. The purpose of the learning activities was to provide the participants with a set of experiences that would allow them to build their knowledge and ability to solve the given problem. For example, the learning activities of section 1 of the database unit (create a database) included the following:

1. Define a database.

2. Explain the difference between data and a database structure.

3. List items to be considered before a database is created.

4. Create a database (personal database) in a computer environment.

5. Create a database using dBase language.

6. Enter data to a dBase database.
Database assignments

After the students completed the learning activities, they were given an assignment related to the learning activities. In each database assignment, the participants were asked questions or given tasks designed to reveal their mental models of a database concept. For example, after participating in a set of learning activities where the participants explored the database concepts of a field, record, and basic data categorization, the assignment was given: Please respond to the following: “Please give an example of a database.” “What is the relationship between records and fields?” ”What is the benefit of data categorization?”

Student responses to the assignments were scored based on accuracy, thoroughness, and detail. The evaluators used quantitative criteria to assess the qualitative data. The purpose of the database assignments was to gather data on students' mental models of the concepts and structures of databases.

The role of the holistic instructional approach

The purpose of the holistic approach was to offer students an complete view of problem solving in a database environment. This complete view included the whole picture of the entire database unit and a complete view of each individual sub-section. Through the holistic instructional method, it was intended that students would generalize problems, divide the problems into sub-problems, and attempt to understand their relationships. After dividing the problems into sub-problems, the students were to use the dBase language as a tool to solve the problems. Thus, the focus of learning in the holistic instructional model was to use
dBase language as a tool to solve problems (as opposed to focusing on the syntax of the dBase language).

**Instrumentation**

Three instruments were used to gather data on student learning of dBase computer language during the study. The instrumentation included the Computing Survey, database assignments, and the Database Examination.

**The Computing Survey**

The Computing Survey was comprised of 52 items and included three sections: general background, basic computing concepts, and attitudes toward computers and problem solving (Appendix E). The purpose of the general background section was to gather demographic information of the workshop participants; it contained 14 items. The purpose of basic computing concepts section was to gather data about participants' knowledge of basic computer concepts; it contained 13 items. The purpose of the attitude section was to gather data about the participants' attitudes toward computers and problems solving; it contained 25 items. The participants were asked to complete the Computing Survey at the beginning of each workshop.

**Database assignments**

The purpose of the database assignments was to gather information about the students' mental models of database concepts and basic computer programming skills during the database unit. There were 20 points possible on each of the five assignments of the database
The database assignments were given to all participants in the both control and the problem analysis groups.

The Database Examination

The Database Examination included items on basic database syntax and programming problems using database concepts. The five-page examination was comprised of 10 general syntax questions and five items on programming problems in the dBase language (Appendix F). A total of 50 points were possible on the general syntax section of the Database Examination and 100 points were possible on the programming problems of dBase language section of the Database Examination. The Database Examination was given to the participants of the control and the problem analysis groups at the end of each workshop.

Variables of the Study

The variables of the study were as follows:

*Independent variables*

The independent variables of the study included the problem analysis and traditional approaches to computer programming instruction. The problem analysis approach consisted of the Database Simulations (designed to enhance the students' mental models of basic database concepts) and the holistic instructional method. The traditional instructional approach to computer programming instruction was accomplished by first teaching language syntax, then semantics, and finally problem solving.

*Dependent variables*

The dependent variables of this study were as follows:
1. The database assignments which consisted of five exercises.

2. The Database Examination (consisting of basic language syntax and computer programming problems).

*Covariate variables*

The covariates for this study were the participants' scores on the basic computer concepts and attitudes sections of the Computing Survey the participants completed at the beginning of each workshop.

*Procedure*

Each workshop included 32 fifty-minute sessions of instruction. In addition, 50 minute periods were given for the participants to complete each data collection instrument. Each workshop consisted of three sessions on BCC, four sessions on Chinese word processing, and 25 sessions on database activities (Appendix G). At the beginning of each workshop, the participants were requested to sign a consent form to indicate their voluntary participation in the study (Appendix C). Then they completed the Computing Survey (Appendix E). At the end of each section of the database unit, assignments were given to all participants (Appendix D). At the end of each workshop, the Database Examination was given to all participants to measure their learning from the database unit.

*Data Analysis*

Below is a description of the statistical analyses conducted on the data gathered in this study.
The Computing Survey

The Computing Survey included three sections: (a) teacher background information, (b) basic computing knowledge and (c) computer and problem solving attitudes. Three statistical analyses were conducted on the data gathered via the Computing Survey.

1. A summary of frequency counts was conducted to compare the demography of the control and problem analysis groups.

2. A t-test was conducted to compare scores on basic computing knowledge of the control and problem analysis groups.

3. A factor analysis was conducted on the attitude section to identify attitudinal factors of the control and problem analysis groups.

4. A t-test was conducted to compare scores on the attitude factors of the control and problem analysis groups.

Database assignment and Database Examination

To understand how the data from the database assignments and Database Examination were analyzed, it is important to know that the two database unit instructors served as the evaluators who graded the database assignments completed by the participants and Database Examination. Two meetings were held by the two evaluators before and during each workshop to review the criteria of the database assignments and the Database Examination. Each evaluator assessed each assignment and Database Examination completed by each participant in both workshops. That is, each assignment (and each Database Examination) was evaluated twice (once by each evaluator).
To measure the consistency of the evaluations, a Pearson Product Moment Correlation coefficients was calculated on the database assignment scores to understand the correlation between the two evaluators. In addition to a paired t-test was conducted to compare the difference between the subjects' scores on assignments graded by the evaluators. It was intended that if there was no significant different between the evaluations on each database assignment and the Database Examination, the mean score for each assignment and the Database Examination for each participant would be calculated and used for data analyses (For example, if participant 1 received 18 points on assignment 1 as assessed by evaluator 1, and 20 points from evaluator 2, then the mean score used for data analysis for participant 1 on assignment 1 is 19).

The results of the Pearson Product Moment Correlation coefficients and paired sample t-tests in the control group appear in Table 3.2. The Pearson Product Moment Correlation coefficients indicate that the correlations on the five assignments graded by the evaluators ranged from .78 to .88. The result of the paired t-tests indicate that there was no statistically significant difference between the scores given by the evaluators for assignments 1 - 5 of the control group.

The results of the Pearson Product Moment Correlation coefficients and paired t-tests of the problem analysis group are shown in Table 3.3. The Pearson Product Moment Correlation coefficients indicate that the correlation on the five assignments graded by the evaluators ranged from .80 to .89. The result of the paired t-tests indicate that there was no statistically significant difference between the scores given by the evaluators for assignments 1
Table 3.2. Pearson Product Moment Correlation coefficients and t-tests for paired evaluations of the control group participants' scores on database assignments 1 - 5

<table>
<thead>
<tr>
<th>Variable</th>
<th>N pairs</th>
<th>Pearson Corr.</th>
<th>Mean</th>
<th>S.D.</th>
<th>Paired Differences</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>t-value df 2-tail sig.</td>
</tr>
<tr>
<td>Assignment 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluator1</td>
<td>47</td>
<td>.78</td>
<td>11.72</td>
<td>3.95</td>
<td>-.22 46 .83</td>
</tr>
<tr>
<td>Evaluator2</td>
<td></td>
<td></td>
<td>11.81</td>
<td>3.92</td>
<td></td>
</tr>
<tr>
<td>Assignment 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluator1</td>
<td>47</td>
<td>.88</td>
<td>15.21</td>
<td>4.48</td>
<td>.20 46 .84</td>
</tr>
<tr>
<td>Evaluator2</td>
<td></td>
<td></td>
<td>15.15</td>
<td>4.41</td>
<td></td>
</tr>
<tr>
<td>Assignment 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluator1</td>
<td>47</td>
<td>.78</td>
<td>12.23</td>
<td>3.46</td>
<td>-.78 46 .44</td>
</tr>
<tr>
<td>Evaluator2</td>
<td></td>
<td></td>
<td>12.51</td>
<td>3.83</td>
<td></td>
</tr>
<tr>
<td>Assignment 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluator1</td>
<td>47</td>
<td>.79</td>
<td>14.23</td>
<td>4.28</td>
<td>-.20 46 .84</td>
</tr>
<tr>
<td>Evaluator2</td>
<td></td>
<td></td>
<td>14.15</td>
<td>4.51</td>
<td></td>
</tr>
<tr>
<td>Assignment 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluator1</td>
<td>47</td>
<td>.88</td>
<td>13.57</td>
<td>4.2</td>
<td>-.74 46 .46</td>
</tr>
<tr>
<td>Evaluator2</td>
<td></td>
<td></td>
<td>13.81</td>
<td>4.4</td>
<td></td>
</tr>
</tbody>
</table>
Table 3.3. Pearson Product Moment Correlation coefficients and t-tests for paired evaluations of the problem analysis group participants' scores on database assignments 1 - 5

<table>
<thead>
<tr>
<th>Variable</th>
<th>N pairs</th>
<th>Pearson</th>
<th>Paired Differences</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Corr. r</td>
<td>Mean</td>
</tr>
<tr>
<td>Assignment 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluator1</td>
<td>53</td>
<td>.85</td>
<td>14.11</td>
</tr>
<tr>
<td>Evaluator2</td>
<td></td>
<td></td>
<td>14.30</td>
</tr>
<tr>
<td>Assignment 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluator1</td>
<td>53</td>
<td>.90</td>
<td>15.17</td>
</tr>
<tr>
<td>Evaluator2</td>
<td></td>
<td></td>
<td>15.60</td>
</tr>
<tr>
<td>Assignment 3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluator1</td>
<td>53</td>
<td>.81</td>
<td>15.75</td>
</tr>
<tr>
<td>Evaluator2</td>
<td></td>
<td></td>
<td>16.12</td>
</tr>
<tr>
<td>Assignment 4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluator1</td>
<td>53</td>
<td>.82</td>
<td>13.47</td>
</tr>
<tr>
<td>Evaluator2</td>
<td></td>
<td></td>
<td>13.98</td>
</tr>
<tr>
<td>Assignment 5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluator1</td>
<td>53</td>
<td>.83</td>
<td>14.19</td>
</tr>
<tr>
<td>Evaluator2</td>
<td></td>
<td></td>
<td>13.83</td>
</tr>
</tbody>
</table>
Table 3.4. Means and standard deviations of participants’ scores on database assignments by group

<table>
<thead>
<tr>
<th>Variable</th>
<th>Average mean</th>
<th>S.D.</th>
<th>Possible score</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control group</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assig. 1</td>
<td>11.77</td>
<td>3.71</td>
<td>20</td>
<td>47</td>
</tr>
<tr>
<td>Assig. 2</td>
<td>15.18</td>
<td>4.31</td>
<td>20</td>
<td>47</td>
</tr>
<tr>
<td>Assig. 3</td>
<td>12.37</td>
<td>3.44</td>
<td>20</td>
<td>47</td>
</tr>
<tr>
<td>Assig. 4</td>
<td>14.19</td>
<td>4.16</td>
<td>20</td>
<td>47</td>
</tr>
<tr>
<td>Assig. 5</td>
<td>13.69</td>
<td>4.17</td>
<td>20</td>
<td>47</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem analysis group</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Assig. 1</td>
<td>14.21</td>
<td>3.35</td>
<td>20</td>
<td>53</td>
</tr>
<tr>
<td>Assig. 2</td>
<td>15.39</td>
<td>4.27</td>
<td>20</td>
<td>53</td>
</tr>
<tr>
<td>Assig. 3</td>
<td>15.95</td>
<td>2.50</td>
<td>20</td>
<td>53</td>
</tr>
<tr>
<td>Assig. 4</td>
<td>13.73</td>
<td>4.76</td>
<td>20</td>
<td>53</td>
</tr>
<tr>
<td>Assig. 5</td>
<td>14.01</td>
<td>4.34</td>
<td>20</td>
<td>53</td>
</tr>
</tbody>
</table>

- 5 of the problem analysis group.

Because there was no statistically significant difference between the database assignment scores generated for each participant by each evaluator, the final scores of each assignment were determined by calculating the average of the scores from both evaluators (Table 3.4).

The Pearson Product Moment Correlation coefficients and paired t-tests of the Database Examination were calculated to compare the scores given by the evaluators on the Database Examination. The data are shown in Table 3.5. In the control group, the Pearson Product Moment Correlation coefficients of the two evaluators were .98 for general syntax
and .97 for programming problems of Database Examination. In the control group, there was no statistically significant difference on the two sections of the Database Examination scores given by the two evaluators.

In the problem analysis group, the Pearson Product Moment Correlation coefficients for the two evaluators were .99 for general syntax and .99 for programming problems of Database Examination. In the problem analysis group, there was no statistically significant difference on the two sections of the Database Examination scores given by the two evaluators (Table 3.5).

Because there was no statistically significant difference on Database Examination scores as graded by the evaluators, the final scores of the Database Examination for both groups were determined by calculating the average of the scores from the evaluators (Table 3.6).

Finally, computing knowledge and attitudes as measured by the Computing Survey served as the covariate. A $1 \times 2$ analysis of covariance (ANCOVA) was conducted to compare the average scores on each database assignment of the two groups. A $1 \times 2$ analysis of covariance (ANCOVA) was conducted to compare the average scores on each section of the Database Examination of the two groups.

A Pilot Study

A pilot study was conducted with 50 inservice teachers enrolled in an introductory computer programming workshop at ISST. The purpose of the pilot study was to test the problem analysis model of computer programming instruction. The pilot test of the problem
Table 3.5. Pearson Product Moment Correlation coefficients and t-tests for paired evaluations of the control and problem analysis participants' scores on Database Examination

<table>
<thead>
<tr>
<th>Variable</th>
<th>Pearson Paired Differences</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Control group</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Database Examination</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>General syntax</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluator1</td>
<td>40.34</td>
<td>9.45</td>
<td>-1.97</td>
<td>46 .055</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluator2</td>
<td>40.83</td>
<td>9.28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programming problems</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluator1</td>
<td>73.79</td>
<td>25.16</td>
<td>-1.50</td>
<td>46 .141</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluator2</td>
<td>75.23</td>
<td>26.18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Problem analysis group</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Database Examination</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>General syntax</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluator1</td>
<td>39.26</td>
<td>11.55</td>
<td>.35</td>
<td>52 .723</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluator2</td>
<td>39.19</td>
<td>11.65</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programming problems</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluator1</td>
<td>78.70</td>
<td>21.66</td>
<td>1.62</td>
<td>52 .111</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluator2</td>
<td>78.06</td>
<td>21.65</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 3.6. Means and standard deviations of participants’ scores on Database Examination by group

<table>
<thead>
<tr>
<th>Variable</th>
<th>Mean</th>
<th>S.D.</th>
<th>Possible score</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Control group</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Database Examination</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>General syntax</td>
<td>40.59</td>
<td>9.33</td>
<td>50</td>
<td>47</td>
</tr>
<tr>
<td>Programming problems</td>
<td>74.51</td>
<td>25.46</td>
<td>100</td>
<td>47</td>
</tr>
<tr>
<td><strong>Problem analysis group</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Database Examination</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>General syntax</td>
<td>39.91</td>
<td>11.57</td>
<td>50</td>
<td>53</td>
</tr>
<tr>
<td>Programming problems</td>
<td>78.38</td>
<td>21.61</td>
<td>100</td>
<td>53</td>
</tr>
</tbody>
</table>

analysis model included the conceptual model (Database Simulation) and the holistic instructional approach. The Database Simulation was used to aid student learning of basic database concepts. The holistic approach to computer programming instruction was used to assist student learning in the database unit. Based on the data gathered in the pilot study, the problem analysis model was modified for use in the research study.
CHAPTER IV. RESULTS AND FINDINGS

The purpose of this chapter is to present the results of the statistical analyses applied to the data gathered from the research instruments of this study. This chapter is organized into three sections. The findings from the Computing Survey completed by the participants are presented in the first section. In the second section, each of the research hypotheses is presented and the related findings are summarized. The final section of this chapter provides a summary of the research findings.

Analysis of Research Components

The Computing Survey

At the beginning of each workshop, the subjects completed the Computing Survey. The Cronbach alpha reliability coefficient was .69 for the basic computing knowledge section of the Computing Survey and .78 for the attitude section of the Computing Survey.

Basic computing concept

The purpose of the basic computing concept section of the Computing Survey was to gather data about the participants' basic computer knowledge. The total possible score on this section was 65 points. The mean score on the basic computing concepts section was 45.43 for the control group and 39.91 for the problem analysis group. To measure the difference in basic computer concept knowledge levels between the control and problem analysis groups, a t-test was calculated on the mean scores. The results of the t-test indicated that there was no statistically significant difference between the control and experimental
group on knowledge of basic computer concept (Table 4.1). Because there was no
significant difference between the groups on this variable, basic computing knowledge was
used as a covariate to control for any individual differences in basic computing knowledge that
may have existed.

**Attitudes toward computers and problem solving**

The purpose of attitude survey was to gather data from participants on their attitudes
toward computers and problems solving. This section of the Computing Survey include 25
Likert-type items. A factor analysis was conducted to identify attitude factors in both the

Table 4.1. T-tests on basic computing knowledge by group

<table>
<thead>
<tr>
<th>Group</th>
<th>N</th>
<th>Variable</th>
<th>Mean</th>
<th>S.D.</th>
<th>t value</th>
<th>2-tail Prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control</td>
<td>47</td>
<td>BCC knowledge</td>
<td>45.43</td>
<td>13.79</td>
<td>1.8</td>
<td>.08</td>
</tr>
<tr>
<td>Problem analysis</td>
<td>54</td>
<td>BCC knowledge</td>
<td>39.91</td>
<td>19.20</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

control and problem analysis groups. The results of the factor analysis are shown in Table
4.2. Three attitude factors were identified in both the control and problem analysis groups.
The attitude factors were: attitude towards mathematics, attitude about using computers and
problem solving, and preferred working styles. A scale was created for each factor by
summing the items that loaded at .5 or more on each factor. The means for mathematics
attitude, attitude about using computers and problem solving, and preferred working style are
shown in Table 4.3. Three t-tests were conducted to test the difference in mathematics
attitude, attitude about using computers and problem solving and preferred working styles
Table 4.2. Factor loading and internal consistencies of the attitude section of the Computing Survey

**FACTOR: Mathematic attitude**

<table>
<thead>
<tr>
<th>ITEM</th>
<th>control group</th>
<th>problem analysis group</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>.66</td>
<td>.68</td>
</tr>
<tr>
<td>31</td>
<td>.85</td>
<td>.84</td>
</tr>
<tr>
<td>43</td>
<td>.59</td>
<td>.74</td>
</tr>
</tbody>
</table>

Cronbach’s alpha: .76 .62

**FACTOR: Attitude about using computers and problem solving**

<table>
<thead>
<tr>
<th>ITEM</th>
<th>control group</th>
<th>problem analysis group</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>.74</td>
<td>.75</td>
</tr>
<tr>
<td>34</td>
<td>.53</td>
<td>.78</td>
</tr>
<tr>
<td>37</td>
<td>.50</td>
<td>.52</td>
</tr>
<tr>
<td>38</td>
<td>.61</td>
<td>.78</td>
</tr>
<tr>
<td>47</td>
<td>.74</td>
<td>.86</td>
</tr>
</tbody>
</table>

Cronbach’s alpha: .75 .80

**FACTOR: Preferred working style**

<table>
<thead>
<tr>
<th>ITEM</th>
<th>control group</th>
<th>problem analysis group</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>.79</td>
<td>.85</td>
</tr>
<tr>
<td>46</td>
<td>.80</td>
<td>.74</td>
</tr>
<tr>
<td>49</td>
<td>.52</td>
<td>.86</td>
</tr>
</tbody>
</table>

Cronbach’s alpha: .67 .77
Table 4.3. Means, standard deviations, and t values of the attitude factors for the control and problem analysis group

<table>
<thead>
<tr>
<th>Group</th>
<th>Control</th>
<th>Problem analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mathematics attitude</td>
<td>2.86 ± .27</td>
<td>2.90 ± .28</td>
</tr>
<tr>
<td>Attitude about using</td>
<td>2.82 ± .26</td>
<td>2.77 ± .26</td>
</tr>
<tr>
<td>computers and problem solving</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Preferred working style</td>
<td>2.29 ± .26</td>
<td>2.34 ± .29</td>
</tr>
</tbody>
</table>

between the control and problem analysis groups. The results indicate there was no statistically significant difference between the groups based on the three attitude factors at an alpha level of .05 (Tables 4.3). Because there was no statistically significant difference between the groups on any of the three attitude factors, a single attitude score calculated for each participant by summing the factor scores. This overall attitude score (computing attitude) was used as a covariate to control for individual differences on attitude.

Results of Hypothesis Testing

Hypothesis 1: Students taught computer programming with the problem analysis model will develop more complete mental models of basic database concepts than students taught with traditional computer programming instruction as measured by their achievement on database assignments.

Three assignments (assignment 1-3) were used to investigate the effect of using the problem analysis model on the acquisition of basic database concepts. The purpose of assignment 1 was to gather data about students' mental models of basic database
characteristics. The purpose of assignment 2 was to gather data about students' mental models of data processing of basic database file. The purpose of assignment 3 was to gather data on students' mental models about their ability to create a student database. Three analysis of covariance (ANCOVA) tests were conducted to test the first hypothesis. The dependent variables were the average scores on assignments 1-3 (Table 3.4); the covariates were basic computing knowledge and computing attitude from the Computing Survey. The independent variables was the instructional approach (problem analysis model vs. traditional model). The results of the six ANCOVA tests are shown in Tables 4.4 - 4.9. When using BCC as a covariate, there was a statistically significant difference between the experimental and control groups on database assignment 1, [F(1,99) = 16.077, p<.001] and on database assignment 3, [F(1,99) = 40.784, p<.001]. When using computing attitude as a covariate, there was a statistically significant difference between the groups on assignment 1, [F(1,99) = 11.73, p<.001] and database assignment 3 [F(1,99) = 35.437, p<.001]. That is, the average scores on assignment 1 and the average scores of assignment 3 were higher for the inservice teachers who were taught using the problem analysis model than the scores of the teachers taught using the traditional model.

Hypothesis 2: Students taught computer programming with the problem analysis model will develop more complete mental models of managing multiple database files than students taught with traditional computer programming instruction as measured by their achievement on database assignments.

Two database assignments (assignments 4-5) were used to investigate the effect of using the problem analysis model on learning about managing multiple database files concepts. The purpose of assignment 4 was to gather data about students' mental models of
Table 4.4. Analysis of covariance (ANCOVA) of database assignment 1 by treatment groups (Covariate: BCC)

<table>
<thead>
<tr>
<th>Source of Variation</th>
<th>Sum of Squares</th>
<th>DF</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig. of F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Covariates</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCC knowledge</td>
<td>96.14</td>
<td>1</td>
<td>96.14</td>
<td>8.33</td>
<td>.005</td>
</tr>
<tr>
<td>Main Effects</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(problem analysis vs. traditional)</td>
<td>185.47</td>
<td>1</td>
<td>185.47</td>
<td>16.08</td>
<td>.001*</td>
</tr>
<tr>
<td>Explained</td>
<td>244.63</td>
<td>2</td>
<td>122.32</td>
<td>10.60</td>
<td>.000</td>
</tr>
<tr>
<td>Residual</td>
<td>1119.01</td>
<td>97</td>
<td>11.54</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>1363.64</td>
<td>99</td>
<td>13.77</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.5. Analysis of covariance (ANCOVA) of database assignment 2 by treatment groups (Covariate: BCC)

<table>
<thead>
<tr>
<th>Source of Variation</th>
<th>Sum of Squares</th>
<th>DF</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig. of F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Covariates</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCC knowledge</td>
<td>355.64</td>
<td>1</td>
<td>355.64</td>
<td>23.83</td>
<td>.000</td>
</tr>
<tr>
<td>Main Effects</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(problem analysis vs. traditional)</td>
<td>16.66</td>
<td>1</td>
<td>16.66</td>
<td>1.12</td>
<td>.29</td>
</tr>
<tr>
<td>Explained</td>
<td>356.68</td>
<td>2</td>
<td>178.34</td>
<td>11.94</td>
<td>.000</td>
</tr>
<tr>
<td>Residual</td>
<td>1448.41</td>
<td>97</td>
<td>14.93</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>1805.09</td>
<td>99</td>
<td>18.23</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 4.6. Analysis of covariance (ANCOVA) of database assignment 3 by treatment groups (Covariate: BCC)

<table>
<thead>
<tr>
<th>Source of Variation</th>
<th>Sum of Squares</th>
<th>DF</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig. of F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Covariates</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCC knowledge</td>
<td>41.00</td>
<td>1</td>
<td>41.00</td>
<td>4.80</td>
<td>.031</td>
</tr>
<tr>
<td>Main Effects</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(problem analysis vs. traditional)</td>
<td>348.71</td>
<td>1</td>
<td>348.71</td>
<td>40.78</td>
<td>.001*</td>
</tr>
<tr>
<td>Explained</td>
<td>360.34</td>
<td>2</td>
<td>180.17</td>
<td>21.07</td>
<td>.000</td>
</tr>
<tr>
<td>Residual</td>
<td>829.37</td>
<td>97</td>
<td>8.55</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1189.71</strong></td>
<td><strong>99</strong></td>
<td><strong>12.02</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 4.7. Analysis of covariance (ANCOVA) of database assignment 1 by treatment groups (covariate: Attitudes)

<table>
<thead>
<tr>
<th>Source of Variation</th>
<th>Sum of Squares</th>
<th>DF</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig. of F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Covariates</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attitudes</td>
<td>2.71</td>
<td>1</td>
<td>2.71</td>
<td>.22</td>
<td>.64</td>
</tr>
<tr>
<td>Main Effects</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(problem analysis vs. traditional)</td>
<td>146.63</td>
<td>1</td>
<td>146.63</td>
<td>11.73</td>
<td>.001*</td>
</tr>
<tr>
<td>Explained</td>
<td>151.21</td>
<td>2</td>
<td>75.61</td>
<td>6.05</td>
<td>.003</td>
</tr>
<tr>
<td>Residual</td>
<td>1212.43</td>
<td>97</td>
<td>12.50</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1363.64</strong></td>
<td><strong>99</strong></td>
<td><strong>13.77</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 4.8. Analysis of covariance (ANCOVA) of database assignment 2 by treatment groups (covariate: Attitudes)

<table>
<thead>
<tr>
<th>Source of Variation</th>
<th>Sum of Squares</th>
<th>DF</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig. of F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Covariates</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attitudes</td>
<td>13.39</td>
<td>1</td>
<td>13.39</td>
<td>.73</td>
<td>.40</td>
</tr>
<tr>
<td>Main Effects</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(problem analysis vs. traditional)</td>
<td>.78</td>
<td>1</td>
<td>.78</td>
<td>.04</td>
<td>.84</td>
</tr>
<tr>
<td>Explained</td>
<td>14.44</td>
<td>2</td>
<td>7.22</td>
<td>.39</td>
<td>.68</td>
</tr>
<tr>
<td>Residual</td>
<td>1790.65</td>
<td>97</td>
<td>18.46</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>1805.09</td>
<td>99</td>
<td>18.23</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.9. Analysis of covariance (ANCOVA) of database assignment 3 by treatment groups (covariate: Attitudes)

<table>
<thead>
<tr>
<th>Source of Variation</th>
<th>Sum of Squares</th>
<th>DF</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig. of F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Covariates</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attitudes</td>
<td>9.82</td>
<td>1</td>
<td>9.82</td>
<td>1.11</td>
<td>.295</td>
</tr>
<tr>
<td>Main Effects</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(problem analysis vs. traditional)</td>
<td>314.38</td>
<td>1</td>
<td>314.38</td>
<td>35.437</td>
<td>.001*</td>
</tr>
<tr>
<td>Explained</td>
<td>329.16</td>
<td>2</td>
<td>164.58</td>
<td>18.55</td>
<td>.000</td>
</tr>
<tr>
<td>Residual</td>
<td>860.55</td>
<td>97</td>
<td>8.87</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>1189.71</td>
<td>99</td>
<td>12.02</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
managing multiple database files. The purpose of assignment 5 was to gather data about students' mental models of how to use dBase language to manage multiple database files. Four ANCOVA tests were conducted to test the second hypothesis. The dependent variables were the scores on assignments 4 and 5. Knowledge of basic computing concepts and computing attitudes as measured by the Computing Survey served as the covariate. The independent variables was the instructional approach (problem analysis model vs. traditional model). The results of the ANCOVA tests are shown in Tables 4.10 - 4.13. There was no statistically significant difference between the problem analysis and control groups on database assignments 4 and 5. Therefore, students taught computer programming with the problem analysis model did not develop more complete mental models of managing multiple database files concepts than students taught with traditional computer programming instruction as measured by their achievements on database assignments.

Hypothesis 3: There will be no significant achievement difference between participants in the problem analysis model and traditional instructional model on the general syntax section of the Database Examination.

The Database Examination was used to investigate the effectiveness of the problem analysis model in computer programming instruction. Two ANCOVA tests were conducted to test the third hypothesis. The dependent variable was the general syntax section scores on the Database Examination. Knowledge of basic computing concepts and computing attitudes as measured by the Computing Survey served as the covariate. The independent variable was the instructional approach (problem analysis model vs. traditional model). The results of the two ANCOVA tests are shown in Tables 4.14 - 4.15. The results indicated that in the general
### Table 4.10. Analysis of covariance (ANCOVA) of database assignment 4 by treatment groups (covariate: BCC)

<table>
<thead>
<tr>
<th>Source of Variation</th>
<th>Sum of Squares</th>
<th>DF</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig. of F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Covariates</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCC knowledge</td>
<td>96.03</td>
<td>1</td>
<td>96.03</td>
<td>4.97</td>
<td>.03</td>
</tr>
<tr>
<td>Main Effects</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(problem analysis</td>
<td>.48</td>
<td>1</td>
<td>.48</td>
<td>.03</td>
<td>.88</td>
</tr>
<tr>
<td>vs. traditional)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Explained</td>
<td>101.42</td>
<td>2</td>
<td>50.71</td>
<td>2.62</td>
<td>.08</td>
</tr>
<tr>
<td>Residual</td>
<td>1874.53</td>
<td>97</td>
<td>19.33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>1975.95</td>
<td>99</td>
<td>19.96</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 4.11. Analysis of covariance (ANCOVA) of database assignment 5 by treatment groups (covariate: BCC)

<table>
<thead>
<tr>
<th>Source of Variation</th>
<th>Sum of Squares</th>
<th>DF</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig. of F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Covariates</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCC knowledge</td>
<td>285.12</td>
<td>1</td>
<td>285.12</td>
<td>18.50</td>
<td>.00</td>
</tr>
<tr>
<td>Main Effects</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(problem analysis</td>
<td>18.60</td>
<td>1</td>
<td>18.60</td>
<td>1.21</td>
<td>.28</td>
</tr>
<tr>
<td>vs. traditional)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Explained</td>
<td>287.63</td>
<td>2</td>
<td>143.82</td>
<td>9.33</td>
<td>.00</td>
</tr>
<tr>
<td>Residual</td>
<td>1494.91</td>
<td>97</td>
<td>15.41</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>1782.54</td>
<td>99</td>
<td>18.01</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 4.12. Analysis of covariance (ANCOVA) of database assignment 4 by treatment groups (covariate: Attitudes)

<table>
<thead>
<tr>
<th>Source of Variation</th>
<th>Sum of Squares</th>
<th>DF</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig. of F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Covariates</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attitudes</td>
<td>1.95</td>
<td>1</td>
<td>1.95</td>
<td>.10</td>
<td>.76</td>
</tr>
<tr>
<td>Main Effects</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(problem analysis vs. traditional)</td>
<td>5.64</td>
<td>1</td>
<td>5.64</td>
<td>.28</td>
<td>.60</td>
</tr>
<tr>
<td>Explained</td>
<td>7.33</td>
<td>2</td>
<td>3.67</td>
<td>.18</td>
<td>.84</td>
</tr>
<tr>
<td>Residual</td>
<td>1968.61</td>
<td>97</td>
<td>20.23</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1975.95</strong></td>
<td><strong>99</strong></td>
<td><strong>19.96</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 4.13. Analysis of covariance (ANCOVA) of database assignment 5 by treatment groups (covariate: Attitudes)

<table>
<thead>
<tr>
<th>Source of Variation</th>
<th>Sum of Squares</th>
<th>DF</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig. of F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Covariates</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attitudes</td>
<td>.04</td>
<td>1</td>
<td>.04</td>
<td>.01</td>
<td>.96</td>
</tr>
<tr>
<td>Main Effects</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(problem analysis vs. traditional)</td>
<td>2.49</td>
<td>1</td>
<td>2.49</td>
<td>.14</td>
<td>.71</td>
</tr>
<tr>
<td>Explained</td>
<td>2.56</td>
<td>2</td>
<td>1.28</td>
<td>.07</td>
<td>.93</td>
</tr>
<tr>
<td>Residual</td>
<td>1779.98</td>
<td>97</td>
<td>18.35</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1782.54</strong></td>
<td><strong>99</strong></td>
<td><strong>18.01</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
syntax section of Database Examination, there was no statistically significant difference between the diagnostic and control group.

*Hypothesis 4: Students taught computer programming with the problem analysis model will perform better on the programming problems of the dBase language section of the Database Examination than students taught with traditional computer programming instruction.*

Three ANCOVA tests were conducted to test the fourth hypothesis. The dependent variables were the scores on the programming problems of the dBase language section of the Database Examination. The independent variable was the instructional approach (problem analysis model vs. traditional model). To control for individual differences caused by attitude and basic computing concepts, scores from both of these sections of the Computing Survey were used as covariates. The results of the ANCOVA with attitude and basic computing knowledge as covariates indicated there was a statistically significant difference (p<.04) between the problem analysis and control groups on the programming problems of the Database Examination (Table 4.16).

When using basic computing knowledge alone as the covariate, there was a statistically significant difference (p<.033) between the diagnostic and control groups on the programming problems of the Database Examination (Table 4.17). When using computing attitudes alone as the covariate, there was no statistically significant difference (p<.42) between the diagnostic and control groups on the programming problems of the Database Examination (Table 4.18).
Table 4.14. Analysis of covariance (ANCOVA) of Database Examination (general syntax) by treatment groups (covariate: BCC)

<table>
<thead>
<tr>
<th>Source of Variation</th>
<th>Sum of Squares</th>
<th>DF</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig. of F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Covariates</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCC knowledge</td>
<td>4597.93</td>
<td>1</td>
<td>4597.93</td>
<td>70.03</td>
<td>.000</td>
</tr>
<tr>
<td>Main Effects</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(problem analysis vs. traditional)</td>
<td>18.84</td>
<td>1</td>
<td>18.84</td>
<td>.29</td>
<td>.593</td>
</tr>
<tr>
<td>Explained</td>
<td>4643.92</td>
<td>2</td>
<td>2321.96</td>
<td>35.36</td>
<td>.000</td>
</tr>
<tr>
<td>Residual</td>
<td>6369.01</td>
<td>97</td>
<td>65.66</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>11012.93</td>
<td>99</td>
<td>111.24</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.15. Analysis of covariance (ANCOVA) of Database Examination (general syntax) by treatment groups (covariate: Attitudes)

<table>
<thead>
<tr>
<th>Source of Variation</th>
<th>Sum of Squares</th>
<th>DF</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig. of F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Covariates</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attitudes</td>
<td>1.60</td>
<td>1</td>
<td>1.60</td>
<td>.01</td>
<td>.91</td>
</tr>
<tr>
<td>Main Effects</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(problem analysis vs. traditional)</td>
<td>46.60</td>
<td>1</td>
<td>46.60</td>
<td>.41</td>
<td>.52</td>
</tr>
<tr>
<td>Explained</td>
<td>47.58</td>
<td>2</td>
<td>23.79</td>
<td>.21</td>
<td>.81</td>
</tr>
<tr>
<td>Residual</td>
<td>10965.35</td>
<td>97</td>
<td>113.05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>11012.93</td>
<td>99</td>
<td>111.24</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 4.16. Analysis of covariance (ANCOVA) of Database Examination (programming problems) by treatment groups (covariates: BCC, Attitudes)

<table>
<thead>
<tr>
<th>Source of Variation</th>
<th>Sum of Squares</th>
<th>DF</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig. of F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Covariates</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCC + Attitudes</td>
<td>18944.15</td>
<td>1</td>
<td>18944.15</td>
<td>52.49</td>
<td>.001</td>
</tr>
<tr>
<td>Main Effects</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(problem analysis vs. traditional)</td>
<td>1647.73</td>
<td>1</td>
<td>1647.73</td>
<td>4.57</td>
<td>.04</td>
</tr>
<tr>
<td>Explained</td>
<td>19291.13</td>
<td>2</td>
<td>9645.57</td>
<td>26.72</td>
<td>.001</td>
</tr>
<tr>
<td>Residual</td>
<td>35011.86</td>
<td>97</td>
<td>360.947</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>54302.99</td>
<td>99</td>
<td>548.52</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.17. Analysis of covariance (ANCOVA) of Database Examination (programming problems) by treatment groups (covariate: BCC)

<table>
<thead>
<tr>
<th>Source of Variation</th>
<th>Sum of Squares</th>
<th>DF</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig. of F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Covariates</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCC knowledge</td>
<td>18689.70</td>
<td>1</td>
<td>18689.70</td>
<td>51.21</td>
<td>.000</td>
</tr>
<tr>
<td>Main Effects</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(problem analysis vs. traditional)</td>
<td>1704.30</td>
<td>1</td>
<td>1704.30</td>
<td>4.67</td>
<td>.033*</td>
</tr>
<tr>
<td>Explained</td>
<td>19062.15</td>
<td>2</td>
<td>9531.07</td>
<td>26.11</td>
<td>.000</td>
</tr>
<tr>
<td>Residual</td>
<td>35404.00</td>
<td>97</td>
<td>364.99</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>54466.14</td>
<td>99</td>
<td>550.16</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 4.18. Analysis of covariance (ANCOVA) of Database Examination (programming problems) by treatment groups (covariate: Attitudes)

<table>
<thead>
<tr>
<th>Source of Variation</th>
<th>Sum of Squares</th>
<th>DF</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig. of F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Covariates</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attitudes</td>
<td>20.27</td>
<td>1</td>
<td>20.27</td>
<td>.04</td>
<td>.85</td>
</tr>
<tr>
<td>Main Effects</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(problem analysis vs. traditional)</td>
<td>364.95</td>
<td>1</td>
<td>364.95</td>
<td>.66</td>
<td>.42</td>
</tr>
<tr>
<td>Explained</td>
<td>392.72</td>
<td>2</td>
<td>196.36</td>
<td>.35</td>
<td>.70</td>
</tr>
<tr>
<td>Residual</td>
<td>54073.42</td>
<td>97</td>
<td>557.46</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>54466.14</td>
<td>99</td>
<td>550.16</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Summary**

The results of the data analysis were reported in this chapter. The demographic information gathered on the Computing Survey was analyzed in the first section. A statistical analysis of the pre-experimental measures of the teachers indicated that there were no statistically significant differences between the two groups on the teachers' basic computing knowledge and their attitudes toward mathematics, using computers and problem solving, and preferred working styles. Therefore, it was assumed that the only significant variable differentiating the two groups was the treatment (problem analysis model vs. traditional model) used to teach database concepts and programming.

In the second section of this chapter, the findings relating to the four hypotheses of the study were reported. The analysis of the data showed that, during the learning of database
activities, the teachers in the problem analysis group performed significantly better than the teachers in the traditional group when learning basic database concepts (database assignment 1) and creating a student database (database assignment 3). That is, the participants in the problem analysis group developed more complete mental models of basic database concepts than the participants in the control group. At the end of the workshop, the inservice teachers in the problem analysis group had statistically significantly higher scores on the section of Database Examination related to programming problems in the dBase language than the inservice teachers in the control group. However, there was no statistically significant difference between the two groups on the dBase language general syntax section of the Database Examination.
CHAPTER V. SUMMARY, DISCUSSION AND RECOMMENDATIONS

The study involved 100 inservice teachers enrolled in a basic computer programming workshop at The Institute for Secondary Schools Teachers in Taiwan. Two workshops were conducted in this study (43 teachers in first workshop and 53 teachers in second workshop). Each workshop consisted of 36 hours instruction over five days. Three subjects were covered in each workshop: basic computer concepts (BCC), Chinese word processing, and database activities.

The purpose of this chapter is to summarize the research study, provide a discussion of the significance of the study, and present recommendations for future research in the area of computer programming instruction. The chapter is organized into three sections. The first section is a brief summary of the background and methodology of the study. Next, a discussion of the major findings of the study is presented. Finally, recommendations are made for future research.

Summary

Computer programming is a complex process that requires students to develop and combine syntactic and semantic knowledge of programming language and apply this knowledge with general problem solving skills to solve specific problems. Several studies have identified problems of novice programmers in learning programming (Du Boulay, 1986; Galloway, 1990; Pea, 1986; Putnam et al., 1986; Segal et al., 1992; Sleeman et al., 1986; Soloway et al., 1983). Novice programmers' problems can be categorized into three areas:
language-independent conceptual problems, language-dependent problems, and basic computing misconceptions.

Helping novice programmers overcome their problems is a difficult task. In this difficult task, it is important to draw from learning theory to support novice programmers' learning. Constructivism offers a way to understand how students acquire new knowledge. Constructivist theory suggests that learning is a constructive process. Knowledge is not separable from action. Moreover, facts are not simply facts to be remembered in isolation; context is an integrated part of meaning. When learning in context, learners can actively interact with their environment to develop usable knowledge.

Based on the constructivism, the goal of computer programming instruction is to provide experiences from which the learner can build meaningful perceptions and to help the learner focus learning on design strategies and processes. Learning computer programming is "... conceptual in nature," involving an understanding of the language syntax, concepts, and principles of programming, and "... procedural in nature," involving how to apply problem solving skills to solve a specific problem (Oliver, 1993, p. 299). When applying the constructivists' ideas to computer programming instruction, computer programming instruction should build the learners' mental models of domain content and focus the design strategy on problem solving.

In computer programming instruction, several conceptual models were used to enhance students' mental models of learning computer programming (Bayman & Mayer, 1986; Dalbey & Linn, 1986; Hooper & Thomas, 1990; Upah & Thomas, 1993). The conceptual models were successful in enhancing novice programmers' mental models of
computer concepts but not in helping to strengthen fragile programming knowledge. To help students overcome their fragile knowledge, studies that used problem-solving oriented instruction were conducted (Black et al., 1994; Volet & Lund, 1994). The problem-solving oriented instruction focused on problem-solving skills and emphasized on when and where to use knowledge to solve problems. Study results suggested that problem-solving oriented instruction helped students integrate their fragile programming knowledge and problem-solving skills.

The purpose of this study was to determine the effectiveness of using the problem analysis learning model in computer programming instruction. The problem analysis model of computer programming instruction included a conceptual model to enhance students' mental models of basic database concepts, and a holistic instructional approach to focus on problem-solving of database programming.

In this study a pretest/post-test control-group design (Campbell & Stanley, 1963) was used. The participants (N=47) in the first workshop, served as the control group and were taught using a traditional programming instruction. The participants (N=53) in the second workshop served as the problem analysis group and were taught using a problem analysis model of computer programming instruction. The problem analysis model combined a conceptual model (a Database Simulation) and holistic instructional methods.

In this study, both the diagnostic and control group were taught basic computer concepts and Chinese word processing using traditional computer programming instruction. The basic computer concepts and Chinese word processing units consisted of a general introduction to computer hardware and software to help the participants use the computers
and basic Chinese word processing. In the database unit for the control group, a traditional instructional approach to teaching database concepts and dBase language programming was used. That is, learning database concepts and dBase computer programming was accomplished by first teaching language syntax, then semantics, and finally problem solving.

In the problem analysis group, the problem analysis model was used to teach database concepts and dBase language programming. That is, to learn basic database concepts, the participants received a conceptual model of databases (via Database Simulation) and then a holistic instructional method was used to teach dBase language programming.

At the beginning of each workshop, all participants completed the Computing Survey. The Computing Survey included three sections: personal background, basic computing knowledge, and attitudes toward computers and problem solving. The results of the Computing Survey showed that there was no significant difference between the two groups on basic computing knowledge and attitude toward mathematics, using computers and problem solving, and preferred working styles. Basic computing knowledge and attitude were used as covariates to control for individual differences in later data analyses.

There were five sections in the database unit in each workshop. At the end of each section, the participants received an assignment. At the end of each workshop, the Database Examination was given to all participants to measure their learning. The Database Examination consisted of two sections: general syntax and programming problems. The general syntax section measured the participants’ factual knowledge of dBase language structure. The programming problems section contained items to measure the participants’ ability to integrate design and programming skills. The results of the data analyses indicated
that using the problem analysis model of computer programming instruction enhanced the
participants’ mental model of basic database concepts and their ability to program in dBase.

Discussion

The purpose of this study was to investigate the effectiveness of using the problem
analysis model of computer programming instruction to teach computer programming. From
the results of the data analyses, each research hypothesis is discussed below.

Hypothesis 1: Students taught computer programming with the problem analysis model will
develop more complete mental models of basic database concepts than students taught with
traditional computer programming instruction as measured by their achievement on
database assignments.

The results of database assignment 1 showed that there was a statistical significance
(p<.001) between the control and the problem analysis groups. This database assignment was
designed to measure the participants’ mental models of basic database concepts. The basic
database concepts were incorporate in the Database Simulation used by the problem analysis
group. The purpose of the Database Simulation used in the problem analysis group was to
build the students’ mental models of basic database concepts such as raw data, organized data,
data categorization, the relationship of a field and a record, the concept of a relational
database, and the relationship between data in different fields. Consistent with Black et al.
(1994) study, the Database Simulation offered a learning environment for students to anchor
their previous experiences with databases, to learn about raw data, and to manipulate a
relational database to construct their knowledge about basic database concepts.

From the results of the statistical analysis of database assignment 1, it was concluded
that the combination of the conceptual model and the holistic instructional method (i.e. the
problem analysis model) was helpful in improving the learners' mental model of basic database concepts. The results were consistent with previous studies of conceptual models (Bayman & Mayer, 1988; Hooper & Thomas, 1990;) and problem-solving oriented instruction (Volet & Lund, 1994). Students constructed their knowledge and enhanced their mental models through the manipulation of conceptual models of basic database concepts. Moreover, when student learning focused on problem-solving strategies, they were more likely to integrate their knowledge in a new domain (Linn & Clancy, 1992). Thus, the researcher concluded that when learning in a new content domain begins, it is helpful to provide novice programmers with a conceptual model of the fundamental learning concepts and a holistic instructional approach. That is, the achievement of the learners who actively construct their knowledge of the basic concepts of a new domain in the first learning stage (the conceptual component of the problem analysis model) can continuously reflect on those concepts when undertaking the next learning stage (the holistic approach of the problem analysis model).

The results of database assignment 2 showed that there was no statistically significant difference between the control and the problem analysis groups on mechanical steps of data processing. This result was similar with the results of the Hooper and Thomas's studies. Hooper and Thomas (1990) indicated that "... questions measuring knowledge of Pascal syntax, ability to locate run-time errors, and ability to hand-execute segments of Pascal code revealed no significant performance differences between the groups (p.447). When students were required to complete simple tasks, the control and experimental group performed at the same level. Database assignment 2 was designed to measure the learners' mental models about data processing (Appendix D). The reason there was no difference between these two
groups is probably due to the mechanical characteristics of data processing in this assignment. That is, when the learning skills focused on step-by-step procedures, there was no achievement difference between the control and experimental groups.

The results from a comparison of student performance on database assignment 3 showed that there was a statistically significant difference (p<.001) between the control and the problem analysis groups. When creating a database, this database assignment was designed to measure the learners' mental models about creating a student database. In this assignment, the participants needed to construct their basic conception of a database to design. The results were consistent with previous studies (Bayman & Mayer, 1988; Volet & Lund, 1994). Bayman & Mayer (1988) suggested that offering appropriated conceptual models of computer operations enhanced student learning in computer programming. Volet and Lund (1994) showed that when computer programming instruction provided step by step planning strategies, students improved their learning performance in programming.

The significant difference in achievement between the control and problem analysis group on database assignment 3 may have been because the participants in the experimental group had more complete mental models of basic database concepts and the ability to integrate their basic database knowledge to design a database. That is, participants in the experimental group constructed better databases than the participants in the control group by integrating their mental models of basic database concepts and the design principles of creating a database.

In summary, the problem analysis model of computer programming instruction included a conceptual model (Database Simulation) and a holistic approach. The conceptual
model was designed to enhance the participants' mental model of basic database concepts (through their manipulation of data in database) and help the participants construct their knowledge to solve problems about databases. After the mental models of basic database concepts were developed, the holistic instructional approach offers a complete view of database concepts and design to direct the participants to solve database problems. The results indicate that the problem analysis model (i.e. the combination of the conceptual model and the holistic approach) enhanced the participants’ learning and helped the participants to develops more complete mental models of basic database concepts than traditional computer programming instruction.

Hypothesis 2. Students taught computer programming with the problem analysis model will develop more complete mental models of managing multiple database files than students taught with traditional computer programming instruction as measured by their achievement on database assignments.

In a database application system, very often programmers must manage multiple database files to handle database file transfers or database report output. The purpose of the database assignments 4 and 5 was to test the ability of students to manage multiple database files. The results from a comparison of student performance on database assignments 4 and 5 showed that there was no statistically significant difference between the control and the problem analysis groups. The reason for no difference in achievement between the groups may be that the researchers assumed managing multiple database files was a sufficiently complex task for novice programmers in a 5 day workshop. The assignments on managing multiple database files were too easy for the control and experimental groups. The mean scores for database assignment 4 were 14.19 in the control group and 13.73 in the
experimental group, and database assignment 5 were 13.69 in the control group and 14.01 in
the experimental group (Table 3.4). Thus, the researcher suggests that giving more difficult
assignments may result in a more effective assessment of the differences between traditional
and problem analysis instructional approaches.

Hypothesis 3. There will be no significant achievement difference between participants in
the problem analysis model and traditional instructional model on the general syntax section
of the Database Examination.

The results from a comparison of student performance on the general syntax section of
the Database Examination showed that there was no statistically significant difference between
the control and the problem analysis groups on syntax. This result is consistent with findings
of Anderson (1983, 1988) and Hooper and Thomas (1990). According to Anderson, the
declarative knowledge (which consists of syntax, concepts and principles of a computer
language) can effectively be directly taught to novice programmers. The reason there was no
difference between the two groups may due to the mechanical characteristics of the general
syntax section of dBase language in Database examination.

Hypothesis 4: Students taught computer programming with the problem analysis model will
perform better on the programming problems of the dBase language section of the Database
Examination than students taught with traditional computer programming instruction.

When using BCC as a covariate, the results on the programming problems section of
the Database Examination showed that there was a statistically significant difference (p<.033)
between the control and the problem analysis groups. When using computing attitude as a
covariate, the results on the programming problems section of the Database Examination
showed that there was no statistically significant difference (p<.42) between the control and
the problem analysis groups. When both BCC and computing attitudes were used as
covariates, the results on the programming problems section of the Database Examination showed that there was a statistically significant difference (p<.04) between the control and the problem analysis groups. The purpose of this part of the Database Examination was to measure the learners' ability to integrate design and programming skills. From the results of data analyses, the researcher concluded that the combination of conceptual models and the holistic instructional approach had a positive effect on student learning of database domain subjects in computer programming. However, the effect of the problem analysis model of computer programming instruction had only a small effect size (.12) on student learning computer programming. The researcher suggests that the small effect size of the problem analysis model of computer programming instruction may be due to that short instructional period of the study (5 days). It is recommended that the problem analysis model be used in a full semester course.

Schwartz et al (1989) indicated that computer programming is "... precision-intensive", "... problem-solving intensive", and "... design-intensive" (p.264). Moreover, Oliver (1993) showed that computer programming is "... conceptual in nature", involving an understanding of the language syntax, concepts and principles of programming, and "... procedural in nature", involving how to apply problem solving skills to solve a specific problem (p.299). Thus, computer programming instruction needs to offer opportunities for students to construct their knowledge and build their mental models of a new content domain. Moreover, computer programming instruction should focus on computer program design strategies and problem solving in order to help students integrate their programming knowledge to solve problems. The findings of this study show that when a conceptual model
of basic database concepts is provided at the beginning of instruction to allow students to construct their knowledge, and further instruction focused on learning how to learn database programming is provided, novice programmers better learn computer programming concepts. In this study, through the manipulation of the conceptual component of the problem analysis model, novice programmers built their mental models of basic database concepts. When student learning focused on problem-solving oriented instruction of database programming, novice programmers were able to integrate their programming knowledge and problem solving skills to solve computer programming problems. Thus, using the problem analysis model in computer programming improved novice programmers' learning of database concepts and programming.

This study offered a potential learning model (problem analysis model) of computer programming instruction to help students learn computer programming. The results of this study may provide a useful conceptual framework for the design a computer programming course for teachers. Computer practitioners/teachers may re-organize their instructional methods based on existing materials to enhance student learning in computer programming. Future researchers may use the problem analysis model as a foundation to explore other subjects. Moreover, they could examine parts of this model in greater detail to identify specific items or procedures that contribute to student learning. Furthermore, this research study also provides a workshop structure that may help in-service teachers increase their computing proficiency, and may assist institutions in organizing their training programs. Thus, the significance of the study has practical and theoretical implication for teaching and learning computer programming.
Recommendations

The purpose of the study was to determine the effectiveness of the problem analysis model of computer programming instruction in an introductory computer workshop. The results showed that using the problem analysis learning model was more effective than traditional instruction in helping students learn basic database concepts and develop programming ability. The following recommendations are made for further study.

1. This study found using the problem analysis model of computer programming instruction was more effective than the traditional computer programming instruction in teaching basic database concepts and developing programming ability. This result could be considered as a basis to investigate whether the problem analysis model is effective with other domain contents. It is recommended to test the problem analysis model be used with other subjects to test the reliability of this model.

2. This study found no significant difference between the problem analysis model and the traditional model on managing multiple database files in the introductory computer workshop. This result may be due to the short learning period used in the study. It is recommended that the problem analysis model be used in a semester-long course to further investigate the effects of the model.

3. The participants in this study were inservice teachers in Taiwan who had little computer knowledge. It is recommended that the problem analysis model be used with a variety of learners with various levels of computer experience and knowledge to examine the potential of this model to facilitate learning at all levels.
4. The instruments used to measure student learning were developed and used for the first time in this study. Thus, the researcher recommends that future research focus on refining these instruments to improve their sensitivity for gathering student learning data. Specifically, the researcher suggests modifying the database assignments to include verbal and written explanations of students’ mental models of various computer programming concepts. Such data would allow for researchers to better understand the mental models developed through the use of the problem analysis model.

Conclusion

Numerous studies have been conducted that examine the effectiveness of using conceptual models to improve student learning in computer programming (Bayman & Mayer, 1988; Dalbey & Linn, 1986; Hooper & Thomas, 1990; Shih & Allesi, 1993; Upah & Thomas, 1993). The use of computer programming instruction based on conceptual models was successful in helping students improve their computer programming; however, additional instruction was needed to help students fully use or integrate their programming skills to solve complex programming problems.

To enhance students’ ability to integrate programming knowledge, research on the effectiveness of using problem-solving oriented instruction in computer programming has also been conducted (Volet & Lund, 1994). Empirical research studies indicated that learning computer programming is a complex process which combines mental models of domain content and the integration of programming knowledge (Anderson, 1980, 1982, 1983; Bitter
Instruction in computer programming needs to build the learners’ mental models of computer domain content and focus on the design strategy of problem solving. This study investigated the effectiveness of the problem analysis learning model of computer programming instruction. The problem analysis learning model included a conceptual model and problem-solving oriented approach to computer programming.

Findings of this study show that when learning in a new content domain begins, it is helpful to provide novice programmers with a conceptual model of the fundamental learning concepts and a holistic instructional approach. Through the manipulation of the conceptual component of the problem analysis model, novice programmers built their mental models of basic database concepts.

When student learning focused on problem-solving oriented instruction with database programming, novice programmers were able to integrate their programming knowledge and problem-solving skills. Thus, using the problem analysis model of computer programming instruction improved novice programmers’ learning of database concepts and programming. However, although the problem analysis learning model of computer programming instruction was helpful in database learning, it is recommended that future research examine the problem analysis model with different subjects of computer programming to test the reliability of this model.
APPENDIX A. COGNITIVE INFORMATION PROCESSING AND THE ACT* THEORIES
Cognitive Information Processing

Cognitive psychology is "... the study of the mental operations that support people’s acquisition and use of knowledge" (Reed, 1996, p. 4). Cognitive psychologists propose that learners are active in processing information and reactive to the environment. Knowledge can be described as a network of mental structures and procedures (schemata) and the associations between them. Understanding plays a key role in learners’ learning and their prior knowledge is critical in new learning (Andre, 1995). Cognitive Information Processing, a contemporary view of cognition, attempts to develop models to describe the specific processes that control learning and the use of knowledge. In CIP models, the mind is viewed as a central information processor to process information between input from the environment and output from the individual (Figure 2.1).

The central information processor in the mind consists of several major components: (a) a sensory registor, (b) working memory, (c) long term memory, (d) executive routine, and (e) behavior (Figure 2.2). According to CIP, the sensory registor acts as a buffer and stores information for about one half second. Working memory (or short term memory) is similar to the idea of consciousness or cognitive processing capacity. It has limits in capacity approximately two to seven items and durations of 30 seconds.

In contrast to working memory, long term memory (LTM) is unlimited and has an indefinite length of storage. Long term memory is composed of episodic memory which consists of personal experiences in a spatial organization, and semantic memory which consists of generalized abstracted knowledge schemata in a network organization. The
executive routine handles the allocation and the flow of information processing. The behavior component is the result of central informational processor.

Pressley and McCormick (1995) explained the distinction between semantic memory and episodic memory:

Semantic memory is knowledge of the world that is now independent of specific experiences. Episodic memory is memory of personally experienced events. Thus, your semantic memory contains an image of a cat. You also have episodic memories of specific cats, perhaps including a pet or one that you have seen hanging around your neighborhood. (p. 56)

Episodic or analogical knowledge is used for mental imagery, whereas semantic knowledge is general and abstract. In general, semantic knowledge includes declarative knowledge and procedural knowledge. Declarative knowledge consists of facts, concepts, and principles used for naming, recognizing, and explaining; procedural knowledge consists of motor skills and general and specific problem-solving skills used for completing tasks.

In the CIP model, working memory organizes information that comes in from the outside world through the sensory system as well as information that is already inside long term memory. Learning is a product of the interaction between information from the outside world and individual knowledge structures (Hargrave, 1993). Knowledge is perceived, encoded, stored, and retrieved through the assimilation and accommodation of self-contained knowledge structures (Piaget, 1983). Therefore, the mental representation of knowledge reflects a hierarchically organized network of individual knowledge structures (Phye & Andre, 1986). This interpretation of learning may help in explaining learning in the area of computer programming. Historically, using behavioral theory to teach computer programming, students
Figure 2.1. General models of the mind (Andre, 1995)

Figure 2.2. Central information processor (Andre, 1995)
are expected to show what they have taught based on instructional design. In behaviorism, the content is simplified and regularized, and is transferred to learners. The characteristics of learners are always ignored during their learning. From the framework of the CIP theory, computer programming may refer to the ability to perform various intellectual procedures related to different knowledge structures. That is, learning computer programming may be regarded as the acquisition of cognitive skill. In past, the ACT* theory provides an explanation of how skill acquisition occurs.

The ACT* Theory

The ACT* theory is a knowledge system that uses productions to represent cognitive rules (Anderson, 1976, 1980, 1982, 1983, 1987). It describes forms of knowledge and the relationship of knowledge to structures of the mind. At the basic level of the ACT* theory, productions are represented in an “if-then” form. A production consists of two parts: a condition and an action. It specifies an action and when the action should occur. “The sequence of productions that apply in a task correspond to the cognitive steps taken in performing the task.” (Anderson, 1982, p. 370).

According to Anderson’s ACT* theory, the distinction between declarative knowledge and procedural knowledge is fundamental. Declarative knowledge is factual information, or knowing “that”, whereas procedural knowledge is knowledge of how to perform tasks, or knowing “how” (Pressley & McCormick, 1995). “... Declarative knowledge consists of knowledge of facts, concepts, or principles ... whereas procedural
knowledge consists of knowledge of how to process or manipulate information to accomplish a given task” (Shih, 1991, p. 9).

ACT* theorists claim that “... knowledge in a new domain always starts out in declarative form and is used interpretively” (Anderson, 1982, p. 375). According to Anderson, a student can learn how to do something in a new domain in the following way. The first stage is the interpreting stage. Students learn from declarative knowledge and call on their knowledge of existing procedures (domain-general knowledge) to direct their behavior in a task. Then, by building upon procedures to perform this specific task, their cognitive structure processes a compilation of knowledge. Finally, students acquire new knowledge and skills in the new domain.

Knowledge compilation can be divided into two subprocesses: composition and proceduralization. When completing a specific task, composition collapses several sequences of productions into a single production that has the effect of the sequence. By creating new operations that embody the sequences of steps, this procedure accelerates one’s ability to solve the particular task. The second process, proceduralization, creates a new domain-specific production that combines the domain-general productions with the declarative knowledge. This procedure reduces the cognitive load on working memory to hold long-term memory information. The consequence of knowledge compilation is the ability to transform declarative knowledge into a procedural form (Anderson, 1982).

When students start with a verbal rehearsal (declarative form) of a procedure to complete a task, the task is carried out slowly, because the procedure requires information to be retrieved from long-term memory and held in working memory. After knowledge
compilation, the knowledge for completing the task is converted to a procedural form. The procedural form of knowledge eliminates the need to hold long-term memory information in one’s working memory, and the knowledge “. . . undergoes a process of continual refinement of conditions and raw increase in speed” (Anderson, 1982, p. 403).

According to the ACT* theory, the acquisition of cognitive skill occurs through the process of knowledge compilation. The process is converted from domain-general productions with declarative knowledge into domain-specific productions. That is, the development of skill is domain-specific to its task. The same declarative knowledge combined with different domain-general productions will have different domain-specific productions. Moreover, different uses of the same declarative knowledge will result in different productions. Possessing the same declarative knowledge does not guarantee one’s ability to acquire the same procedural knowledge. But, if two skills involve the same productions, there will be a positive transfer effect between these two skills (Anderson, 1987). This concept of cognitive skill acquisition and transfer ability has a far-reaching implications of teaching and learning complex tasks such as computer programming. According to Anderson (1982), when teaching and learning a complex skill, “. . . in the first stage, the learner receives instruction and information about a skill. The instruction is encoded as a set of facts about the skill” (p. 369-370). That is, first declarative knowledge (facts and principles) is taught to learners, because declarative knowledge can be encoded directly. Then, the learners learn how to apply declarative knowledge. Finally, with practice, the learners acquire a complex skill.
The ACT* theory is applied to an artificial intelligence system that attempts to mirror human cognition. It emphasizes that declarative and procedural knowledge are stored in different parts of long-term memory. Furthermore, in long-term memory, procedural and declarative knowledge are not linked; they are stored separately. Procedural and declarative knowledge work together only when they are activated in working memory. Thus, according to Anderson (1983), procedural and declarative knowledge are independent and belong to different cognitive databases. That is, declarative and procedural knowledge are separately existed and acquired. The way to teach computer programming based on the ACT* theory is to separately teach language syntax, semantic, and problem solving. Learning computer programming is required to master language syntax, mastering language semantic, and solving programming problems. In this way of teaching and learning computer programming, students are easy handling different learning stage but they acquire fragile knowledge not integrated ability. This concept of learning and knowledge is different from the ideas of constructivists who believe knowledge is inseparable from actions (Brown, Collins, & Duguid, 1989).
APPENDIX B. WORKSHOP SCHEDULES
Workshop Schedule

Date:
1. Workshop 1: Apr. 22 - Apr. 27 (a control group)
2. Workshop 2: Apr. 29 - May. 4 (a problem analysis group)

Subject:
1. The population of this study was inservice secondary school teachers in Taiwan.
2. The first week workshop: of the 53 inservice teachers initially enrolled, 47 participants completed all learning activities.
3. The second week workshop: of the 56 inservice teachers initially enrolled, 53 participants completed all learning activities.
4. There are one instructors in the sections of Basic Computer Concept and Chinese word processing.
5. There are two instructors in the database unit.

Schedule:

<table>
<thead>
<tr>
<th>Description</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Section 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Section 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Section 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Break</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Section 5</td>
<td>The Computing Survey</td>
<td>Basic database concepts</td>
<td>A student database</td>
<td>dBase language programming</td>
<td>Integrated application</td>
</tr>
<tr>
<td>Section 6</td>
<td>Basic Computer Concept</td>
<td>Basic database concepts</td>
<td>A student database</td>
<td>dBase language programming</td>
<td>A Workshop Suggestion Survey</td>
</tr>
<tr>
<td>Section 7</td>
<td>Basic Computer Concept</td>
<td>Create a database</td>
<td>A student database</td>
<td>dBase language programming</td>
<td>Database Examination Part I</td>
</tr>
<tr>
<td>Section 8</td>
<td>Basic Computer Concept</td>
<td>Create a database</td>
<td>A student database</td>
<td>dBase language programming</td>
<td>Examination Part II</td>
</tr>
</tbody>
</table>

Workshop 1 Schedule: Apr. 22 - Apr. 27

Exercise 1
Exercise 2
Exercise 3
Exercise 4
Exercise 5
**Workshop 2 Schedule: Apr. 29 - May. 4**

<table>
<thead>
<tr>
<th>Description</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section 1</td>
<td>Chinese word processing</td>
<td>Data Processing</td>
<td>Managing multiple database files</td>
<td>Sorting programming</td>
<td></td>
</tr>
<tr>
<td>Section 2</td>
<td>Chinese word processing</td>
<td>Data Processing</td>
<td>Managing multiple database files</td>
<td>Sorting programming</td>
<td></td>
</tr>
<tr>
<td>Section 3</td>
<td>Chinese word processing</td>
<td>Data Processing</td>
<td>Managing multiple database files</td>
<td>Sorting programming</td>
<td></td>
</tr>
<tr>
<td>Section 4</td>
<td>Chinese word processing</td>
<td>Data Processing</td>
<td>Managing multiple database files</td>
<td>Sorting programming</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Exercise 2</td>
<td>Exercise 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Break</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Section 5</td>
<td>The Computing Survey</td>
<td>Database Simulation I</td>
<td>A student database</td>
<td>dBase language programming</td>
<td>Integrated application</td>
</tr>
<tr>
<td>Section 6</td>
<td>Basic Computer Concept</td>
<td>Database Simulation II &amp; III</td>
<td>A student database</td>
<td>dBase language programming</td>
<td>A Workshop Suggestion Survey</td>
</tr>
<tr>
<td>Section 7</td>
<td>Basic Computer Concept</td>
<td>Create a database</td>
<td>A student database</td>
<td>dBase language programming</td>
<td>Database Examination Part I</td>
</tr>
<tr>
<td>Section 8</td>
<td>Basic Computer Concept</td>
<td>Create a database</td>
<td>A student database</td>
<td>dBase language programming</td>
<td>Examination Part II</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Exercise 1</td>
<td>Exercise 3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX C. HUMAN SUBJECTS APPROVAL FORM AND CONSENT FORM
Checklist for Attachments and Time Schedule

The following are attached (please check):

12. ☐ Letter or written statement to subjects indicating clearly:
   a) purpose of the research
   b) the use of any identifier codes (names, #s), how they will be used, and when they will be
      removed (see Item 17)
   c) an estimate of time needed for participation in the research and the place
   d) if applicable, location of the research activity
   e) how you will ensure confidentiality
   f) in a longitudinal study, note when and how you will contact subjects later
   g) participation is voluntary; nonparticipation will not affect evaluations of the subject

13. ☐ Consent form (if applicable)

14. ☐ Letter of approval for research from cooperating organizations or institutions (if applicable)

15. ☐ Data-gathering instruments

16. Anticipated dates for contact with subjects:

<table>
<thead>
<tr>
<th>First Contact</th>
<th>Last Contact</th>
</tr>
</thead>
<tbody>
<tr>
<td>4/22/96</td>
<td>5/4/96</td>
</tr>
</tbody>
</table>

17. If applicable: anticipated date that identifiers will be removed from completed survey instruments and/or audio or visual tapes will be erased:

6/30/97

18. Signature of Departmental Executive Officer Date Department or Administrative Unit

19. Decision of the University Human Subjects Review Committee:

☐ Project Approved ☐ Project Not Approved ☐ No Action Required

Patricia M. Keith 7-25-96 PM Keith

Name of Committee Chairperson Date Signature of Committee Chairperson
Dear Teacher:

Thank you for your interest in participating in the Computer Workshop sponsored by the Institute for Secondary School Teachers in Taiwan.

As part of the workshop, I plan to conduct research to improve computer programming instruction. I am conducting this research in partial fulfillment of my doctorate degree. The purpose of this study is to examine the effectiveness of a diagnostic learning model to enhance student learning in database programming. The study activities will be fully integrated into the Computer Workshop; that is, there will be no additional activities for you to complete by participating in the study. As a participant in the Computer Workshop, you may volunteer to participate in the study by signing your name below.

As a participant in the study, all of the information or responses you provide will be kept strictly confidential; moreover, no individual data or names will ever be reported.

As you know, the Computer Workshop will take place for one week and will consist of the following units: Basic Computer Concepts, Chinese Word Processing, and Database Programming. At any time during the Computer Workshop, you have the right to no longer participate in the study which will not in any way effect your participation or successful completion of the Computer Workshop.

Sincerely,

Hsiao-shen Wang

Yes, I would like to participate in the Computer Programming study

name ___________________________ date ___________________________
APPENDIX D. LESSON PLAN AND DATABASE ASSIGNMENTS
The lesson plans of database activities in problem analysis group.

Section 1: Create a Database (consisting of two 50-minute activities)

*Problem introduction* - At the beginning of the database activities in the holistic approach, the students were given a school grading book and asked specific questions such as “Who has the highest math score?” “What is the average science score among all of the students?” and “What is Linda’s rank in history?”

*Problem Diagnosis* - Participants were requested to design a method to solve the problems. After making their design, students participated in whole class discussions to generalize some procedures to answer the questions. For example, students suggested “using computers to create a student database” and “using computer commands figure out the answers to the problems.”

*Learning activities*

1. What is a database?
2. What is the difference between data and a database structure?
3. What should be considered before a database is created?
4. Creating a database (personal database) in a computer environment.
5. Creating a database using a dBase language.
6. Entering data to a dBase database.

*Database assignments* - After students completed their activities, they were given the assignment: Creating a Database. The purpose of this assignment was to gather data on students' mental models of the concept and structure of a database.
Section 2: Data processing (consisting of four 50-minute activities)

Problem introduction - When the students had the ability to create a database, they were first asked what they would do in the database, and then they were asked to think about what will happen to the data in the database.

Problem Diagnosis - Students recorded their thinking processes on paper. Through whole class discussions, they revised their ideas and generalized some questions for the further learning.

Learning Activities

1. Using an existing database.
2. Appending, editing, deleting, and browsing data in a database.
3. Listing or finding specific data in a database.
4. Replacing data in one field with data from other fields.
5. Summing and averaging numeric data in a database.
7. Saving a dBase database.

Database Assignments - After students completed the activities in this section, they were given a Data Processing assignment. The purpose of this assignment was to gather students' mental models of when and where to use specific sorting commands as different tools in solving different database situations.

Section 3: A Student Database (consisting of four 50-minute activities)

Problem introduction - After the students processed the data in the database, they were encouraged to reflect on the database structure of the school’s grade book.
Problem Diagnosis - Students were requested to record their ideas about what kind of data should be included in a student (grade) database. They were also asked to think about the kind of problems that will result if a database structure is not designed well.

Learning Activities

1. What kind of data should be included in a student (grade) database?
2. What kind of data types should be considered in the student database?
3. Define fields, data types, data width, and data decimal format for the student database.
4. Use the dBase language tool to create the student database.
5. Use index files to sort data in the student database.
6. Use Sort and Replace commands to sort data and place rank in the student database.
7. Use the Append From command to merge several databases into one large database.

Students participated in a whole class discussion and generalized the main considerations of the student database. Then the students used the dBase language as a tool to create and sort their database according to the problems they encountered.

Database Assignments - At the conclusion of the section, the students were given a Student Database Assignment. This purpose of this assignment was to gather information about students' mental models of an integrated design of a database.

Section 4: Managing Multiple Database Files (consisting of four 50-minute activities)

Problem introduction - After completing the activities in the previous sections, students would have acquired the knowledge and experience to a process single database file. Then, the students were given two different documents (customer list and customer billing documents) that included some common data fields, (e.g., Identification No., Last name and
The workshop participants were also asked a question to induce their data input. The question asked if they input data to the two database files, what kind of monotonous or mechanical operations will be needed. One of the mechanical operations was to repeat the entry of data such as Identification No., Last name and First name.

*Problem Diagnosis* - Students were encouraged to think about the previous problems and to develop their solutions to overcome them. For example, some participants suggested reorganizing the data structure of the customer list and customer billing documents.

*Learning Activities*

1. Create and use multiple database files.
2. List data in multiple database files.
3. Use the *Total* and *Update* commands to summarize and update the data in multiple database files.

*Database Assignments* - At the end of this section, the students were given an assignment related to multiple database files. The purpose of this assignment was to gather information about the students' understanding of when and how to use multiple database files.

*Section 5: dBase Language Programming (consisting of eight 50-minute activities)*

*Problem introduction* - The purpose of the first four sections of the database activities was to process databases in a *Dot command mode*. The *dot command* mode is a learning environment in the dBase language. This learning environment is designed to use command-by-command in the command line to process a database. One can only execute one command at a time. Thus, students were asked to think about what kinds of inconvenient situations they had encountered in this mode. One disadvantage of the *Dot command* mode suggested by the
students was that they needed to continually repeat the same commands to complete the same task.

*Problem Diagnosis* - Students discussed and recorded their approaches to solve the problems in the Dot command mode.

*Learning Activities*

1. Writing and executing dBase language programming.
2. Collecting dot commands into a program file.
3. Variables and programming concepts.
4. Loop1: *Do while* ... *enddo*
5. Loop2: *If* ... *else* ... *endif*
6. Loop3: *Do case* ... *endcase*
7. Simple programs.

In order to learn how to solve computer programming problems, students were first requested to design and record their approaches in a top-down format. Then, they were encouraged to discuss their approaches with their peers to explain how and why they broke the problems into subproblems and the relationships that existed between the sub-problems. Finally, they used the dBase computer language as a tool to solve the problems.

*Database Assignments* - After the students completed the activities in section 5, they received a dBase Language Programming Assignment. Then the students submitted their design approaches and dBase programming codes and results. These data were used to understand their mental models of how to solve computer programming problems.
Database Assignments

Section 1 Creating a Database Assignment

1) What is the benefit of data categorization? (at least mention two benefits, each two points, total 4 points)

2) What kind of relationship will have between records and fields? (at least mention two benefits, each two points, total 4 points)

3) Please give a database example. (4 point for structures, records, fields, and data)

4) Please indicate database structures, records, fields and data in above example database. (2 points for each, total 8 points)

Section 2 Data Processing Assignment

1) Please explain what is the difference between `Index` and `Sort` commands while sorting data in a database. (at least mention two difference, each five points, total 10 points)

2) Please use person database to do questions below: (10 points)

   A. append three more data. (3 point)

   B. list all males who live in "Taipei" in this database. (3 point)

   C. create a index file to sort data by first name in ascending order. (4 point)

Section 3 A Student Database Assignment:

1) What should be considered in designing your local student database? (each 2 points for purpose, field name, field type, field width, and decimal position, total 8 points)

2) When we use `Sort` and `replace` commands to sort and place students' rank, do you find any problems? (2 points: they cannot sort same scores in same rank.)

3) Please design and implement our workshop database. (5 points for database structure design, 5 points for implementation, total 10 points)

Section 4 Managing Multiple Database Files Assignment
1) What kind of disadvantages are there when using previous person and student databases together? (at least mention two items, each 5 points, total 10 points)

2) Please redesign and reimplement person and student databases. (5 points for database structure design, 5 points for implementation, total 10 points)

Section 5 dBase Language Programming Assignment

1) What is the advantage of writing programs instead of using dot commands? at least mention three items, each 2 points, total 6 points)

2) Please write a program to print out the sum from 1 to 10. (9 points)

3) Please write a program to complete following tasks. (5 points)
   * Open student database (1 point)
   * List all data in the database (1 point)
   * Sort all data to a new math database according to math scores in descending order. (1 point)
   * Open the math database (1 point)
   * List all data in the math database (1 point)
APPENDIX E. THE COMPUTING SURVEY
A Computing Survey

The purpose of this survey is to gather information about your experience, knowledge and attitudes toward computers and problems solving. The survey is divided into three sections: a personal background section, computer knowledge and experience section, and a computer and problem solving attitude section. Please answer all of the questions by recording your answers on the answer sheet provided. Your responses to this survey will in no way affect your achievement in this workshop.

Section I Background

The purpose of this section is to gather information about your personal background and teaching experiences. Please answer all of the questions by recording your answers on the answer sheet.

1. What is your gender?
   A. male
   B. female

2. What is your age?
   A. 20 or younger
   B. 21 - 23
   C. 24 - 31
   D. 32 - 39
   E. 40 or older

3. How many years of teaching experience do you have?
   A. 1-5
   B. 6-10
   C. 11-15
   D. 16 or more

4. Currently, what is the main subject you teach? (Please mark the item(s) you teach.)

<table>
<thead>
<tr>
<th>Chinese</th>
<th>English</th>
<th>Mathematics</th>
<th>Physics</th>
<th>Chemistry</th>
<th>Biology</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Health</th>
<th>History</th>
<th>Geography</th>
<th>Other (specify)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5. What is the highest level of education you have completed?
   A. Bachelors
   B. Bachelors + 15 credits
   C. Masters
   D. Masters + 15 credits
   E. Other (specify) 

6. When working on a project in your area of expertise, do you prefer to work:
   A. by yourself
   B. with a small group.

7. When learning mathematics concepts, do you prefer to work:
   A. by yourself
   B. with a small group.

8. Do you prefer solving mathematics problems:
   A. with a small group. go to question 9
   B. by yourself go to question 10

9. I enjoy solving math problems in a small group, because I:
   A. can learn a great deal from other people.
   B. enjoy the social interaction.
   C. can develop more accurate answers.
   D. can better understand the problem.
   E. All of above.
   F. Other: (Specify ______________) go to question 11

10. I enjoy solving math problems by myself, because I:
    A. can think through the problem at my own pace in my own way.
    B. can learn how to complete each step without confusing comments from others.
    C. think groups get off the task too much.
    D. can better understand the problem.
    E. All of above.
    F. Other: (Specify ______________)

11. Have you had any computer courses before this workshop?
    A. No  go to question 13
    B. Yes

12. How many semesters of total course work in computer literacy have you had?
    A. less than a full semester
    B. one semester
    C. more than one semester
    D. Other (specify ___________________________)

13. Have you had any computer experiences before this workshop?
   A. none
   B. some
   C. much
   D. Other (Specify ____________________________)

14. How comfortable are you in using a mouse with a computer?
   A. very comfortable
   B. somewhat comfortable
   C. somewhat uncomfortable
   D. very uncomfortable

Section II General Computing Concepts

The purpose of this section is to gather information about your knowledge of general computing concepts. Please answer all of the questions by recording your answers on the answer sheet.

15. Which of the following is not considered computer software?
   A. Operation System
   B. Word Processor
   C. Spreadsheet
   D. Disk

16. Which of the following is not considered computer hardware?
   A. Keyboard
   B. Monitor
   C. Personal Editor II
   D. Mouse

17. A keyboard is an example of which type of computer device?
   A. Input device
   B. Output device
   C. Computation device
   D. Memory device
18. A printer is an example of which type of computer device?
   A. Input device  
   B. Output device  
   C. Computation device  
   D. Memory device

19. A CPU (Central Processing Unit) is an example of which type of computer device?
   A. Input device  
   B. Output device  
   C. Computation device  
   D. Memory device

20. RAM (Random-Access Memory) is an example of which type of computer device?
   A. Input device  
   B. Output device  
   C. Computation device  
   D. Memory device

21. In a computer system, how many bits equal a byte?
   A. 2 bits  
   B. 4 bits  
   C. 8 bits  
   D. 16 bits

22. In a computer system, 1 KB = _______ bits.
   A. 128  
   B. 256  
   C. 512  
   D. 1024

23. Which of the following is a function of RAM (Random-Access Memory)?
   A. to temporarily hold a program or data  
   B. to provide system power  
   C. to print a character on the printer  
   D. to convert keystrokes to binary numbers

24. If a computer has a 288 MB hard disk, the hard disk is equal to how many 1.44MB floppy disks?
   A. 100  
   B. 200  
   C. 300  
   D. 400
25. Which statement below accurately describes the function of ROM (Read-Only Memory) with data?
   A. Data in this memory are temporary.
   B. Data in this memory can be read and written.
   C. Data in this memory cannot be erased.
   D. Data in this memory will be erased when the computer turns off.

26. Generally, we can use which type of devices to connect a local computer to the Internet?
   A. Keyboard
   B. Mouse
   C. CPU
   D. Modem

27. When we say that a computer is a 486, what does 486 refer to?
   A. Memory size
   B. Monitor type
   C. CPU model
   D. Disk capacity

Section III Computer Supported Learning Attitude Survey

The purpose of this section is to gather information on your attitudes about computer supported learning. Please answer each item on the questionnaire by recording your answers on the answer sheet. If you have questions concerning the questionnaire, raise your hand and a proctor will assist you. When you have completed the test, turn it in with your answer sheet.

Using a scale where 1 = Strongly Disagree, 2 = Disagree, 3 = Agree and 4 = Strongly Agree, read each statement and circle the item that best represents your opinion of the statement. Remember, there are no "right" or "wrong" answers. If you agree, but not completely, then circle 2. If you disagree, think how strongly you disagree; a little bit? then circle 3. If you disagree a lot, then circle 4.

<table>
<thead>
<tr>
<th></th>
<th>1 Strongly Disagree</th>
<th>2 Disagree</th>
<th>3 Agree</th>
<th>4 Strongly Agree</th>
</tr>
</thead>
</table>

28. Working in the area of mathematics is very exciting. 1 2 3 4

29. The domain of computer science is logical and well organized to me. 1 2 3 4
30. Learning how to solve problems is more important than solving the problem itself.

31. Mathematics is a very important subject to know.

32. When learning, I like to discuss problems with my classmates.

33. Mathematics is very difficult to understand.

34. An effective sequence for learning problem solving is to break problems into sub-problems, analyzing their relationships, and then learn the skills to solve the sub-problems.

35. Mathematics is not an important subject to know.

36. Working with computers is very exciting.

37. The domain of mathematics is logical and well organized.

38. Computer science is a very important subject to know.

39. When learning, I enjoy working on problems by myself.

40. Working with computers is frustrating.

41. Classroom discussions are a waste of class time during learning.

42. Knowledge of computers is not important.

43. To solve a mathematics problem, I first need to understand the problem.

44. Learning to solving a problem itself is more important than learning the ways of problem solving.

45. Computer concepts are very difficult to understand.

46. If I have difficulty learning or understanding a concept, I ask my friends to help me.
47. An effective sequence for learning problem solving is to learn basic math skills, learn advanced math skills, and then learn to solve problems.

48. Mathematics is frustrating.

49. Classroom discussions are an important activity during learning.

50. To solve a mathematics problem, I first need to understand the math.

51. If I have difficulty learning or understanding a concept, I try to figure it out by myself.

52. I enjoy using computers to communicate with friends.
APPENDIX F. THE DATABASE EXAMINATION
A Database Examination

The Database Examination includes two parts: general syntax problems and programming problems of dBase language. You have two 50-minute sections to complete this examination.

I. General syntax problems (total 50 points)
There are 10 questions in this part of Database Examination. Please use correct syntax of the dBase commands to answer following questions.

1. How would you create a new database file, called sales.dbf in a dBase language? (5)

2. If a database file (sales.dbf) has a field (ID), how would you create an index for the database on the field? (5)

3. How would you modify a field structure in a database file (sales.dbf)? (5)

4. Write dBase commands to list all data in a database file (sales.dbf)? (5)

5. After completing the following dBase commands, which record will be deleted in person.dbf? (5)

   [Commands]
   . USE PERSON
   . GOTO 3
   . DELETE NEXT 2

6. After completing the following dBase commands, which record will be modified in person.dbf? (5)

   [Commands]
   . USE PERSON
   . GOTO 4
   . GOTO 10
   . EDIT
7. If we would like to use second working area of computer memory for person.dbf, how would you do it in dBase commands? (5)


8. If we would like to append data from person1.dbf and person2.dbf into person.dbf, how would you do it in dBase commands? (5)


9. If we would like to sort data of person.dbf to a new file (new1.dbf) according to the field of last name (Lname) in a ascending order, how would you do it in dBase commands? (5)


10. If we would like to clear data on the screen and list all database files in directory, how would you do it in dBase commands? (5)


II. Programming problems (total 100 points)

1. 1) Please create a database (total.dbf) using the data shown below. (10)

2) Write a program to print out all data in the database (total.dbf) and sort data to a new file (price.dbf) according to the field (price) in a descending order. (10)

total.dbf

<table>
<thead>
<tr>
<th>Last name</th>
<th>First name</th>
<th>Price</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lee</td>
<td>Paul</td>
<td>156.00</td>
<td>3/1/96</td>
</tr>
<tr>
<td>Wilson</td>
<td>Annie</td>
<td>118.00</td>
<td>3/3/96</td>
</tr>
<tr>
<td>Thompson</td>
<td>John</td>
<td>35.00</td>
<td>3/5/96</td>
</tr>
<tr>
<td>Carter</td>
<td>Mary</td>
<td>67.00</td>
<td>3/7/96</td>
</tr>
<tr>
<td>Lin</td>
<td>Doris</td>
<td>106.00</td>
<td>3/15/96</td>
</tr>
<tr>
<td>Hung</td>
<td>Saly</td>
<td>98.00</td>
<td>3/17/96</td>
</tr>
<tr>
<td>Thomas</td>
<td>Smith</td>
<td>135.00</td>
<td>3/19/96</td>
</tr>
</tbody>
</table>
2. Please write a program, using the database below to print out a list which math score is greater than 80 and science score is greater than 90. (20)

<table>
<thead>
<tr>
<th>STUDENT.DBF</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>NAME</td>
<td>MATH</td>
<td>SCIENCE</td>
<td>ENGLISH</td>
<td>HISTORY</td>
</tr>
<tr>
<td>1</td>
<td>LIN</td>
<td>89</td>
<td>90</td>
<td>87</td>
<td>92</td>
</tr>
<tr>
<td>2</td>
<td>WANG</td>
<td>78</td>
<td>85</td>
<td>84</td>
<td>92</td>
</tr>
<tr>
<td>3</td>
<td>LEE</td>
<td>83</td>
<td>91</td>
<td>92</td>
<td>84</td>
</tr>
<tr>
<td>4</td>
<td>HSU</td>
<td>87</td>
<td>93</td>
<td>82</td>
<td>98</td>
</tr>
<tr>
<td>5</td>
<td>HUANG</td>
<td>71</td>
<td>78</td>
<td>92</td>
<td>91</td>
</tr>
</tbody>
</table>

3. Please write a dBase program to calculate and print out the result of \((13 + 14 + \ldots + 19)\).
   A. Write down how you analyze the problems (explain in Chinese). (5)
   B. Write a dBase program to solve the question. (10)
   C. Explain the dBase program (explain in Chinese line by line, or block by block). (5)

   for example
   ```
   use person (open person.dbf database file)
   ...
   ```

4. A bookstore will use computers to run its business. (20)
   A. Please design a database for the bookstore. (explain this database structure) (8)
   B. Write three questions that will be easy to answer using your database. (explain in Chinese) (6)
   C. Write three questions that will be difficult but possible to answer using the database. (explain in Chinese) (6)

5. Please write a program to complete the following tasks.
   A. fill total scores in below database. (10)
   B. fill students' rank based on their total scores. (10)

<table>
<thead>
<tr>
<th>student.dbf</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>NAME</td>
<td>MATH</td>
<td>SCIENCE</td>
<td>ENGLISH</td>
<td>HISTORY</td>
<td>TOTAL</td>
<td>RANK</td>
</tr>
<tr>
<td>1</td>
<td>LIN</td>
<td>89</td>
<td>90</td>
<td>87</td>
<td>92</td>
<td>364</td>
<td>92</td>
</tr>
<tr>
<td>2</td>
<td>WANG</td>
<td>78</td>
<td>85</td>
<td>84</td>
<td>92</td>
<td>357</td>
<td>92</td>
</tr>
<tr>
<td>3</td>
<td>LEE</td>
<td>83</td>
<td>91</td>
<td>92</td>
<td>84</td>
<td>350</td>
<td>92</td>
</tr>
<tr>
<td>4</td>
<td>HSU</td>
<td>87</td>
<td>93</td>
<td>82</td>
<td>98</td>
<td>350</td>
<td>92</td>
</tr>
<tr>
<td>5</td>
<td>HUANG</td>
<td>71</td>
<td>78</td>
<td>92</td>
<td>91</td>
<td>332</td>
<td>92</td>
</tr>
</tbody>
</table>
Database Material Outline

Session 1: basic database concept and create a database

Content:
1. Data and its' meaning
2. The relationship between raw data and organized data.
3. The difference between good and bad data categorization.
4. What is a file, a record, and a field?
5. The relationship between a record and a field.
6. What is a database?
7. What is the difference between data and database structures?
8. What should be considered before a database being created?
9. How to create a database using a dBase language.
10. How to enter data to a dBase database.
11. How to modify dBase data structures.

Session 2: data processing

Content:
1. How to use an existing database in dBase language.
2. How to append, edit, delete and browse data in a database.
3. How to list or find specific data in a database.
4. How to replace fields' data with other fields.
5. How to sum and average numeric data in a database.
6. How to index or sort data in a database.
7. How to save a dBase database.

Session 3: create a student database

Content
1. What kind of data should be included in a student (grade) database?
2. What kind of data types should be considered in the student database?
3. Define fields, data types, data width, and data decimal format for the student database.
4. Create the student database.
5. Create index files to sort data in the student database.
6. Using Sort command to sort data and place rank.
7. Using Append From command to merge several databases to a big database.
Session 4: managing multiple database files
1. What kind of situations are needed for managing multiple database files?
2. Create a customer list database (custlist.dbf) and an index file (custlist.ndx).
3. Create a charged database (charges.dbf) and an index file (charges.dbf).
4. Open and use multiple database files.
5. List data in multiple database files.
6. Total & Update commands

Session 5: dBase language programming

Content
1. How to write and execute dBase language programming?
2. Collect dot commands into a program file.
3. Variables and programming concepts.
4. Loop1: do while ... enddo
5. Loop2: if ... else ... endif
6. Loop3: Do case ... endcase
7. Simple programs
APPENDIX H. DATABASE REFERENCE MATERIALS
Database Reference Materials

Session 1: basic database concept and create a database

Reference materials

1. Data and its' meaning
   Data can be defined as two or more items of factual information that have a definable relationship to each other. For instance, '39' without a context in which to place it, it has little meaning. It could be 'a street address', 'a debit', or 'a football player's number.' If the number 39 related to other data items, such as name and age below
   
<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benny</td>
<td>39</td>
</tr>
</tbody>
</table>

   now the number 39 has meaning- it's both a person's reported age and one of the longest running gags in the history of American comedy.

   The whole concept of data depends on the linkage of facts. Salaries are linked with people, and tax brackets are linked with salaries; teachers are linked with children, children with parents, and parents with teachers through the children they have in common.

2. What is a file, a record, a field?
   (For example, a file cards contain nothing except preprinted blanks for a name and a city.)
   * The term file is reserved to mean the totality of a group of cards.
   * Each individual file card is a record.
   * The term field is reserved to mean a specific piece of data within a record.

3. What is a database?
   A database is a collection of useful information organized in a specific manner. such as
   - a personal telephone directory
   - a teacher-students directory

4. What is a database structure?
   A database structure is a structure to build up the database. For instance, a personal database structure could include Last name, First name, Sex, Telephone, Address.
5. What is the difference between data and a database structure?

For example: in a student database

<table>
<thead>
<tr>
<th>Last name</th>
<th>First name</th>
<th>Sex</th>
<th>Telephone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wang</td>
<td>Linda</td>
<td>female</td>
<td>515-2921357</td>
</tr>
<tr>
<td>Lin</td>
<td>Paul</td>
<td>male</td>
<td>515-2949997</td>
</tr>
<tr>
<td>Lee</td>
<td>Dam</td>
<td>male</td>
<td>515-2964312</td>
</tr>
</tbody>
</table>

A database structure includes last name, first name, sex and telephone.
Data: Wang, Linda, female, 515-2921357, etc.

6. What should be considered before a database being created?

* What is the purpose of this database?
* What kind of data will be included in this database?
* Field name, Field type, Field width?
* Data type
  * Character data consists of alphabetic characters (letters A through Z), numbers (0 through 9), and some special symbols (such as # and $) and is stored as strings.
  * Numeric data can be quantified and is represented by the set of numeric digits.
  * Logical data (Boolean): Logical or Boolean values, named after George Boole, an English mathematician of the 1800s, are those data items that represent one of two mutually exclusive conditions.
  * Dates: Variables declared as dates are represented as eight-character data strings in the format 'mm/dd/yy.' Data variables resemble integers and can be subjected to date arithmetic, such as addition and subtraction.
  * Memos: It is a text entry of up to 4,096 characters to be entered "in" the memo field.

7. How to create a new database using a dBase language?

Before actual creating a new database using a dBase language, several things must be decided.

1) Database filename?
2) Database structure?
3) Data type?

After above things are decided, then we must be familiar with a dBase language environment.

1) What is a dBase language?
2) How to run dBase in PC computer?
3) How to quit dBase in PC computer?
4) What's a command mode and an assisted mode in a dBase environment?

The way to create a new database in a dBase language

Command syntax: Create database_filename
Example: Create students
8. How to modify a database structure?
Using Modify Structure command to modify an existed database structure.
Command syntax: Modify database_filename
Example: Modify students

[ Creating a new dBase database example ]

. create person (creating a new database)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Type</th>
<th>Width</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ID</td>
<td>Character</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>2 Lname</td>
<td>Character</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>3 Fname</td>
<td>Character</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>4 Address</td>
<td>Character</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>5 City</td>
<td>Character</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>6 State</td>
<td>Character</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>7 Zip</td>
<td>Character</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

Press the Ctrl+End key, dBase will display message

*Press ENTER to confirm. Any other key to resume*

Press the Enter key. On your screen, you will now see that the computers asking if you want to

*Input data records now? (Y/N)*

Type N to answer No and return to the dot prompt.

. append (append data into a database)

ID
Lname
Fname
Address
City
State
Zip
(Input below data)

<table>
<thead>
<tr>
<th>reco#</th>
<th>ID</th>
<th>LNAME</th>
<th>FNAME</th>
<th>ADDRESS</th>
<th>CITY</th>
<th>STATE</th>
<th>ZIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>480231234</td>
<td>Smith</td>
<td>Dave</td>
<td>1221 Hawthorn court</td>
<td>Ames</td>
<td>IA</td>
<td>50010</td>
</tr>
<tr>
<td>2</td>
<td>481987456</td>
<td>Wang</td>
<td>Linda</td>
<td>105 University Village</td>
<td>Ames</td>
<td>IA</td>
<td>50011</td>
</tr>
<tr>
<td>3</td>
<td>4803487416</td>
<td>Doe</td>
<td>Ruth</td>
<td>1245 Hawthorn Court</td>
<td>Ames</td>
<td>IA</td>
<td>50013</td>
</tr>
<tr>
<td>4</td>
<td>481187457</td>
<td>Lee</td>
<td>Paul</td>
<td>222 Lincoln Way</td>
<td>Ames</td>
<td>IA</td>
<td>50015</td>
</tr>
<tr>
<td>5</td>
<td>480983151</td>
<td>Smith</td>
<td>Betsy</td>
<td>1105 Kellogg Av.</td>
<td>Ames</td>
<td>IA</td>
<td>50121</td>
</tr>
<tr>
<td>6</td>
<td>481987456</td>
<td>Smith</td>
<td>Betsy</td>
<td>461 Adams St.</td>
<td>Ames</td>
<td>IA</td>
<td>50015</td>
</tr>
</tbody>
</table>

Press the Enter key to finish data entry.

. list (list data in a dBase database)

<table>
<thead>
<tr>
<th>reco#</th>
<th>ID</th>
<th>LNAME</th>
<th>FNAME</th>
<th>ADDRESS</th>
<th>CITY</th>
<th>STATE</th>
<th>ZIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>480231234</td>
<td>Smith</td>
<td>Dave</td>
<td>1221 Hawthorn court</td>
<td>Ames</td>
<td>IA</td>
<td>50010</td>
</tr>
<tr>
<td>2</td>
<td>481987456</td>
<td>Wang</td>
<td>Linda</td>
<td>105 University Village</td>
<td>Ames</td>
<td>IA</td>
<td>50011</td>
</tr>
<tr>
<td>3</td>
<td>4803487416</td>
<td>Doe</td>
<td>Ruth</td>
<td>1245 Hawthorn Court</td>
<td>Ames</td>
<td>IA</td>
<td>50013</td>
</tr>
<tr>
<td>4</td>
<td>481187457</td>
<td>Lee</td>
<td>Paul</td>
<td>222 Lincoln Way</td>
<td>Ames</td>
<td>IA</td>
<td>50015</td>
</tr>
<tr>
<td>5</td>
<td>480983151</td>
<td>Smith</td>
<td>Betsy</td>
<td>1105 Kellogg Av.</td>
<td>Ames</td>
<td>IA</td>
<td>50121</td>
</tr>
</tbody>
</table>

. list structure (list a existed database structure)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Type</th>
<th>Width</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ID</td>
<td>Character</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>2 Lname</td>
<td>Character</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>3 Fname</td>
<td>Character</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>4 Address</td>
<td>Character</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>5 City</td>
<td>Character</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>6 State</td>
<td>Character</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>7 Zip</td>
<td>Character</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

. modify structure (modify a existed database structure)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Type</th>
<th>Width</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ID</td>
<td>Character</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>2 Lname</td>
<td>Character</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>3 Fname</td>
<td>Character</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>4 Sex</td>
<td>Character</td>
<td>1 &lt; modified database structures</td>
<td></td>
</tr>
<tr>
<td>5 Address</td>
<td>Character</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>6 City</td>
<td>Character</td>
<td>15</td>
<td></td>
</tr>
</tbody>
</table>
Press the Ctrl+End key to finish data entry.

- **browse** (browse and edit data in a database)
- **list to print** (print out data)
- **list structure to print** (print out data structures)

**Session 2: data processing**

**Reference materials**

1. **How to use an existed database in dBase language?**
   Command syntax: Use `database_filename`

2. **How to append, edit, delete and browse data in a database?**
   Command syntax:
   - Append
   - Edit `<record no.>`
   - Delete `<record no.>`
   - Browse

3. **How to list or find specific data in a database?**
   Command syntax: `List `<scope>` [ `<operation>` ][ `For `<condition>` ] [ `off` ] [ `to print` ]
   **List structure** (list a database structure on screen)
   Examples:
   1) list
   2) list record 5
   3) list next 3
   4) list rest
   5) list Last_name, First_name
   6) list Last_name, First_name for address="Taipei"
   7) list for sex ="male" to print

4. **How to replace fields’ data with other fields?**
   Command syntax: `Replace `<scope>` [ `<field_name>` ] With `<operation>` [ , `<field_name>` With `<operation>` ]...[ `For `<condition>` ]
   Examples:
   1) replace all math with 0
   2) replace all total with Chinese + math
   3) replace all average with total / 2
   4) replace math with 90 for last_name="wang" and first_name="linda"
   5) replace rank with recno()
5. How to sum and average numeric data in a database?
Command syntax:

**Sum** [<scope>][<operation>][To <variables>][For <condition>]

Examples:
1) set talk on (display message on screen)
   sum math / 10
2) sum Chinese/6, math/6, average/6 to C,M,A

**Average** [<scope>][<operation>][To <variables>][For <condition>]

Examples:
1) average Chinese, math
2) average math * 4
3) average Chinese, math, average for sex

6. How to index or sort data in a database?
Command syntax:

**Index**

* **Index on** <field_name> to <index_filename> [Unique]
* **Set Index to** <index_filename>
* **Use** <database_filename> **Index** <index_filename>

Examples:
1) index on last_name to index_name
2) set index to index_name
3) use person index index_name

**Sort to** <new_filename> on <field_name> [/A][/C][/D],[ <field_name> [/A][/C][/D]... [For <condition>]

Examples:
1) sort to score_math on math /D
2) sort to name_list on last_name /A
3) sort to sort_a on average /D for average >= 60

7. How to save a database?
Command syntax: **Use**
Session 3: create a student database

Reference materials

1. A example student database structure:

<table>
<thead>
<tr>
<th>Fieldname</th>
<th>Type</th>
<th>Width</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ID</td>
<td>Character</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>2 Lname</td>
<td>Character</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>3 Fname</td>
<td>Character</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>4 English</td>
<td>Numeric</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>5 Math</td>
<td>Numeric</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>6 Science</td>
<td>Numeric</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>7 History</td>
<td>Numeric</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>8 Total</td>
<td>Numeric</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>9 Average</td>
<td>Numeric</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>10 Rank</td>
<td>Numeric</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

2. Create index files to sort data in the student database.
   Examples:
   . use student
   . index on math to math_index
   . index on science to science_index
   . set index to math_index
   . set index to science_index

3. Using Sort & Replace commands to sort data and place rank.
   Examples:
   . use student
   . sort on math to math_1
   . use math_1
   . replace rank with recno()
   . list
   . use

4. Using Append From command to merge several databases to a big database.
   Command syntax: Append from <database_name> [For <condition>]
   Examples:
   . use data1
   . append from data2
Session 4: managing multiple database files

Reference materials

1. What kind of situation are needed to manage multiple database files?
   Some business applications require multiple database because any given record in one
database might have several items of information related to it.
   For example: a account database.

<table>
<thead>
<tr>
<th>Lname</th>
<th>Fname</th>
<th>Address</th>
<th>Amount</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wang</td>
<td>Linda</td>
<td>1234 Duff Ave</td>
<td>50.00</td>
<td>1/1/96</td>
</tr>
<tr>
<td>Wang</td>
<td>Linda</td>
<td>1234 Duff Ave</td>
<td>123.00</td>
<td>1/9/96</td>
</tr>
<tr>
<td>Wang</td>
<td>Linda</td>
<td>1234 Duff Ave</td>
<td>73.00</td>
<td>2/3/96</td>
</tr>
<tr>
<td>Lee</td>
<td>Mike</td>
<td>1478 Ocean St.</td>
<td>221.00</td>
<td>2/19/96</td>
</tr>
<tr>
<td>Lin</td>
<td>Paul</td>
<td>208 Kellogg Ave</td>
<td>29.00</td>
<td>2/12/96</td>
</tr>
<tr>
<td>Lee</td>
<td>Mike</td>
<td>1478 Ocean St.</td>
<td>21.00</td>
<td>2/20/96</td>
</tr>
<tr>
<td>Lin</td>
<td>Paul</td>
<td>208 Kellogg Ave</td>
<td>56.00</td>
<td>2/21/96</td>
</tr>
</tbody>
</table>

   This is a poorly designed database because of the following problems:
   1. The database wastes disk space by repeating name and address for each transaction.
      These repetitions are particularly space-consuming in a database with 10,000 or more
      records!
   2. The database creates extra work for input operators who must type the repeated
      information each time a new record is added.
   3. Because names and addresses would be entered each time a new record is added, the
      likelihood of typographical errors is high, and therefore the likelihood of billing errors is
      also high.

   A better way to store these data would be to break the information into two databases.
   First database: Custlist.dbf

<table>
<thead>
<tr>
<th>CustNo</th>
<th>Lname</th>
<th>Fname</th>
<th>Address</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>Wang</td>
<td>Linda</td>
<td>1233 Duff Ave</td>
<td>Ames</td>
</tr>
<tr>
<td>1002</td>
<td>Lee</td>
<td>Mike</td>
<td>1478 Ocean St.</td>
<td>Pomona</td>
</tr>
<tr>
<td>1003</td>
<td>Lin</td>
<td>Paul</td>
<td>208 Kellogg Ave</td>
<td>Ames</td>
</tr>
</tbody>
</table>

   Second database: Charges.dbf

<table>
<thead>
<tr>
<th>CustNo</th>
<th>Amount</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>50.00</td>
<td>1/1/96</td>
</tr>
<tr>
<td>1001</td>
<td>123.00</td>
<td>1/9/96</td>
</tr>
<tr>
<td>1001</td>
<td>73.00</td>
<td>2/3/96</td>
</tr>
<tr>
<td>1002</td>
<td>221.00</td>
<td>2/19/96</td>
</tr>
<tr>
<td>1003</td>
<td>29.00</td>
<td>2/12/96</td>
</tr>
<tr>
<td>1002</td>
<td>21.00</td>
<td>2/20/96</td>
</tr>
<tr>
<td>1003</td>
<td>56.00</td>
<td>2/21/96</td>
</tr>
</tbody>
</table>
The customer number, called *a key field*, is the ideal common field because it does not take up much disk space. It is also very easy to assign each customer a unique number. It is important that each record has a unique customer number to relate it to the transactions in the Charges.dbf database.

2. **Create a customer list database (custlist.dbf) and an index file (custlist.ndx).**
   Examples:
   ```
   create custlist
   custlist.dbf
   
   Field name | Type   | Width | Dec
   1 Custno   | Numeric| 9     | 0
   2 Lname    | Character| 15   |
   3 Fname    | Character| 10   |
   4 Address  | Character| 25   |
   5 City     | Character| 10   |
   
   . index on custno to custlist.ndx
   ```

3. **Create a charged database (charges.dbf) and an index file (charges.dbf).**
   Examples:
   ```
   create charges
   charges.dbf
   
   Field name | Type   | Width | Dec
   1 Custno   | Numeric| 9     | 0
   2 Amount   | Numeric| 4     |
   3 Date     | Date   | 8     |
   
   . index on custno to charges.ndx
   ```

4. **Open and use multiple database files**
   Examples:
   ```
   select 1
   use custlist index custlist
   select 2
   use charges
   set relation to custno into custlist
   ```

5. **List data in multiple database files**
   Examples:
   ```
   list custno, custlist->lname, custlist->fname, amount
   ```
6. Total same customer amount in charges.dbf.
   Command syntax:
   ```
   total to <filename> on <key_field> [scope][fields<field>][for <condition>]
   update on <key_field> form <filename> replace <field_name> with <operation>
   [<field_name> with <operation>]
   ```
   Examples:
   ```
   . select 1
   . use charges index charges
   . total to charges1.dbf on custno
   ```

**Session 5: dBase language programming**

**Reference materials**

1. How to write dBase language programming?
   Command syntax: **Modify command** <filename>
   ```
   Do <filename>
   ```
   Examples:
   ```
   . modify command test
     clear
     ? "Hello. I'm a test program."
     press <Ctrl+Enc> to save the file
     . do test
   ```

2. Collect dot commands into a program file.
   Examples:
   ```
   . modify command program1
     use student
     list
     use person
     list
     press <Ctrl+Enc> to save the file
     . do program1
   ```

3. Variables and loop concepts in a program.
   Command syntax:
   ```
   1. store 1 to x; (x = 1)
   2. Do while <condition> ... enddo
   3. If <condition> ... else ... endif
   4. Do Case
      Case <condition>
      Otherwise
      endcase
   ```
Examples: 1
CLEAR
SET TALK OFF
STORE 1 TO X
DO WHILE X <=10
  ? X
  X=X+1
ENDDO
? "ALL DONE"

Examples: 2
CLEAR
ACCEPT "Input number: (1-10)" to number
IF NUMBER < 1 OR NUMBER > 10
  ? "Incorrect number."
ELSE
  CLEAR
  ? "Number."
  ?? number
ENDIF

Examples: 3
CLEAR
INPUT "Enter a number from 1 to 4 : " to X
DO CASE
  CASE X=1
  ? "You entered one."
  CASE X=2
  ? "You entered two."
  CASE X=3
  ? "You entered three."
  CASE X=4
  ? "You entered four."
  OTHERWISE
  ? "Incorrect data."
ENDCASE
REFERENCES


ACKNOWLEDGMENTS

I wish to express my sincere appreciation to Dr. Constance Hargrave, my major professor, for her help, support, valuable guidance, and assistance in the development and completion of this study. Without her help, I could not complete this dissertation. Special appreciation is expressed to Dr. Ann Thompson for her priceless support and help. Thank you also Dr. Thomas Andre for providing excellent learning experiences in cognitive psychology. I would also like to thank Dr. Rex Thomas and Dr. Pete Boysen for their expert advice and patience in testing and offering suggestions for the improvement of the database simulation for this study. Thank you Dr. Thomas for your classes about constructivism; I really learned a great deal from your teaching. Thank you Dr. Boysen for your object-oriented programming class and the expert experience in the development of ABC object-oriented programming systems.

A note of special thanks goes to my many friends with whom I had the good fortune to work and share many good times. To Paul Reinhart and Annie Grieshop, I am fortunate to have you as friends to share my learning experiences about eastern-western cultures and so many new things in the USA. I would also like to express my appreciation to all my friends in the department of Curriculum and Instructional Technology. I really enjoyed studying here and have learned so much. I am also grateful to Pat Hahn for providing so much time and assistance in editing this dissertation.

I would also like to thank National Science Council of the Republic of China (ROC), Taiwan Provincial government, and The Institute for Secondary School Teachers in Taiwan.
for providing me with the scholarship which enabled me to complete my studies. I consider myself lucky to be given this opportunity to complete my education at this stage of my career.

Most important, I would like to thank my family members for providing me with support and encouragement. I greatly appreciate my grandfather and grandmother for their never-ending help no matter what I need. To my parents and mother-in-law, I express my sincere appreciation for their long-term support in my study. I also thank my brothers and sister for looking after my affairs while I was away from home.

Finally, I am deeply thankful to my wife, Chien-pin, and my two children, Yi-xuan and Hwei-xuan. Your support and encouragement are always my strength help in every stage of my learning. I love you.
感謝

能完成這篇論文，首先要感謝我的論文指導教授 Dr. Constance Hargrave，由於她的支持、協助、指導，以及無時無刻的提供意見和修改論文，才能使我完成論文研究，以及此篇論文。在此亦要特別感謝我的共同指導教授 Dr. Thompson 辛苦的詳讀此篇論文，並給予許多寶貴的修改意見，讓此篇論文的架構及用詞更臻完善。另外要感謝我的指導委員，心理學教授 Dr. Thomas Andre 在認知心理學上給予我許多啓發及教導，讓我在發展論文研究中，能依循學習心理學的理論建立整篇論文的基礎架構。

我還要特別感謝另外兩位指導委員，電腦系教授 Dr. Rex Thomas 和 Dr. Pete Boysen。感謝他們兩位在我發展的三套資料庫學生學習模擬系統期間，提供許多專業上的意見，並耐心的進行無數次的專業測試，沒有他們的協助，我是無法完成這些電腦系統的設計及開發。除了感謝兩人在發展系統上的協助之外，還要感謝他們兩人在專業課程上的指導。感謝 Dr. Thomas 的課程，結合電腦系統設計和建構主義的學習理論，使我獲益甚多。感謝 Dr. Boysen 的軟體物件導向設計，以及提供由他個人所開發在大電腦、個人電腦、以及麥金塔電腦上能互相轉換的 ABC 物件導向語言軟體的發展心得，使我增進許多軟體發展的實務經驗。

在學習的過程當中，特別要感謝我的同學 Paul Reinhart 和 Annie Grieshop 許多的鼓勵、協助，以及共同分享東西方的文化差異及學習經驗。另外也要感謝系上的許多朋友在學習期間的共同討論、研究、及幫忙。在此還要特別感謝 Pat Hahn 花費無數的時間及精力，協助此篇論文的編輯及訂正。

我還要特別感謝中華民國國科會、臺灣省政府，以及臺灣省中等學校教師研習會，提供我往赴美國進修博士的機會。我深感慶幸能獲有此機會，完成求學的最後階段。我同時要感謝我的家人給我許多的支持及鼓勵。特別感謝我的外祖父母給與我永無止盡的關愛及協助，我的父母及岳母在我研究期間的充足支援，我的兄弟姐妹在我
求學期間，無時無刻的幫忙及協助，以及我的岳家弟妹，在我離家期間，幫忙照顧兩位孩子的生活及各項課外學習的接送事宜。

最後我要致上最深忱的謝意給我的妻子，倩萍。沒有妳在身旁的協助，無怨無悔的付出，以及獨自一人扛下許多家庭重任，我一定無法完成十多年來的求學生涯。這十多年來，從無數夜晚的等門，準備任何再晚都不會糊掉的即刻麵食，於我公費往赴曼谷亞洲理工學院，修讀電腦碩士期間，獨自操持家務照顧怡萱，以及來美國修讀博士學位期間，辛苦帶領怡萱、惠萱生活學習，和擔負家庭事務的重擔。倘若沒有妳如此任勞任怨，完全解除我後顧之憂，我是無法在這幾年之間，雖辛苦尚且順利的完成各階段學業，謝謝妳的辛苦，妳亦完成艱辛的學位工作，PHD（Push Husband Degree：協助先生獲得學位）永遠銘記於心，謝謝。

曉萱 寫於 美國愛荷華州立大學
一九九六年七月二十二日