I am the author of the thesis entitled

Factors Affecting Success in Undergraduate Computer Programming

submitted for the degree of

Master of Commerce

This thesis may be made available for consultation, loan and limited copying in accordance with the Copyright Act 1968.

Name: Annegret Goold
Signed: Signature Redacted by Library
Date: 26/7/99
TO: ALL USERS OF THIS THESIS

Please sign this form to indicate that you have used this thesis in accordance with the disposition signed by the author of this thesis.

Thank you.

SUE McKNIGHT
University Librarian

<table>
<thead>
<tr>
<th>Name</th>
<th>Signature</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAVID McCORMICK</td>
<td></td>
<td>22/8</td>
</tr>
<tr>
<td>Elicia Lanham</td>
<td></td>
<td>26/5/03</td>
</tr>
<tr>
<td>JOHN DONALD</td>
<td></td>
<td>15 - 8 - 04</td>
</tr>
</tbody>
</table>
FACTORS AFFECTING SUCCESS IN UNDERGRADUATE COMPUTER PROGRAMMING

Annegret Goold
Bachelor of Applied Science (EDP)
Graduate Diploma of Business

Submitted in total fulfillment of the requirements for the degree of Master of Commerce

Faculty of Business and Law

July 1999
DEAKIN UNIVERSITY

CANDIDATE'S CERTIFICATE

I certify that the thesis entitled

Factors Affecting Success in Undergraduate Computer Programming,

and submitted for the degree of

Master of Commerce

is the result of my own research, except where otherwise acknowledged, and that this thesis in whole or in part has not been submitted for an award including a higher degree to any other university or institution.

Name: ANNEGRET GOOLD

Signature: ...

Date: July, 1999
ACKNOWLEDGEMENTS

I am indebted to Mr. Alastair Anderson for his insight and guidance in the initial phase of this study. Dr. Russell Rimmer's later role as the supervisor brought this research to its completion. I greatly appreciate his technical advice and the constant encouragement he gave.

The major findings in the research depended on the participation of many Deakin University students and staff members. Without them and others from a number of tertiary educational institutions, I could not have applied the theories involved here.

To Lindsay, my family and friends, who so generously supported me throughout the preparation of this thesis - I thank you.
# TABLE OF CONTENTS

## CHAPTER 1

**INTRODUCTION**

1.1 Computer Programming: A Short History
1.2 Computer Programming Research
1.3 Background and Significance of This Research
1.4 Structure of the Thesis

## CHAPTER 2

**COMPUTER PROGRAMMING AND HOW NOVICES LEARN**

2.1 The Activity of Computer Programming
  2.1.1 Tasks in the Programming Process
  2.1.2 Skills in Programming Tasks
  2.1.3 Programming Tasks and the Environment
  2.1.4 Measuring Programmer Performance

2.2 Computer Programming and the Novice
  2.2.1 Chunking Information
  2.2.2 Cognitive Models
  2.2.3 Semantic versus Syntactic Knowledge
  2.2.4 Programming Plans and Strategies
  2.2.5 Why Don’t Novices Do Well?

2.3 Computer Programming and Individual Characteristics
  2.3.1 Mathematics Ability
  2.3.2 Gender
  2.3.3 Personality Factors
  2.3.4 Motivation

2.4 The Role of Problem-Solving in Computer Programming

2.5 Achievements in Programming Courses

2.6 Looking to the Preliminary Study
CHAPTER 3

THE PRELIMINARY SURVEY

3.1 Introduction

3.2 Background

3.3 Research Method

3.4 The Survey
  3.4.1 Computer Anxiety
  3.4.2 Computer Playfulness
  3.4.3 Locus of Control
  3.4.4 Computer Experience and Knowledge
  3.4.5 Background Variables

3.5 Results of the Preliminary Survey
  3.5.1 Demographics of the Two Groups
  3.5.2 Computer Knowledge and Experience
  3.5.3 Computer Anxiety, Computer Playfulness and Locus of Control
  3.5.4 Experience, Anxiety, Playfulness and Locus of Control

3.6 Conclusions

CHAPTER 4

EXPLORING THE COLLECTIVE VIEW OF THE EXPERTS

4.1 Introduction

4.2 Choosing a Group Knowledge Acquisition Technique

4.3 Focus Groups

4.4 Cognitive Mapping

4.5 Graphics COPE Software

4.6 Eliciting Information from Experts
  4.6.1 The Focus Group Session
  4.6.2 Cognitive Maps from Graphics COPE

4.7 Views of the Experts, Structuring Student Interviews

4.8 Implications for Further Research
CHAPTER 5

INFORMATION FROM COMPUTER-PROGRAMMING STUDENTS

5.1 Introduction 56
5.2 Background 56

5.3 Research Method
5.3.1 Format and Content of the Interviews 57
5.3.2 Survey Used during the Interview 58
5.3.3 Measuring Aptitude for Computer Programming 60

5.4 Results of the Study
5.4.1 Performance in Computer Programming 61
5.4.2 Interviews 61
5.4.3 Survey Results 64
5.4.4 Results of Aptitude Test 68

5.5 Conclusions 69

CHAPTER 6

THE MAIN STUDY

6.1 Introduction 70
6.2 Background 70

6.3 Research Method 73

6.4 The Survey
6.4.1 Learning Style 74
6.4.2 Knowledge and Experience of Programming 75
6.4.3 Problem Solving 77
6.4.4 Background and Education 78

6.5 Looking Forward 81
CHAPTER 7

RESULTS OF THE MAIN STUDY 83

7.1 Introduction 83

7.2 Comparison of Business and Science Students 83
7.2.1 Demographics of the Two Groups' Responses 83
7.2.2 Learning Style 85
7.2.3 Computer-Programming Knowledge and Experience 90
7.2.4 Problem Solving 92
7.2.5 Educational Background 92

7.3 Regression Analysis 94
7.3.1 Success in the First Programming Unit 95
7.3.2 Programming Over Two Semesters 99

7.4 Conclusion 102

CHAPTER 8

CONCLUSIONS 104

APPENDIX A: PRELIMINARY SURVEY 109

APPENDIX B: FOCUS GROUP 122

APPENDIX C: STUDENT INTERVIEW DOCUMENTS 126

APPENDIX D: MAIN STUDY 137

REFERENCES 144
FIGURES, MAPS AND TABLES

FIGURE 1.1: THE STRUCTURE OF THE THESIS 5
FIGURE 2.1: FRAME FOR TASK ANALYSIS 6
FIGURE 2.2: THE TASKS OF PROGRAMMING 8
FIGURE 2.3: LONG-TERM MEMORY MODEL 14
FIGURE 4.1: GROUP KNOWLEDGE ACQUISITION TECHNIQUES 42
FIGURE 6.1: NORMAL STUDY SEQUENCE FOR BUSINESS COMPUTING 71
FIGURE 6.2: NORMAL STUDY SEQUENCE FOR COMPUTER SCIENCE 72
FIGURE 7.1: LEARNING STYLE AND OCCUPATIONAL FIELD 89

MAP 4.1: PERSONAL MOTIVATION 48
MAP 4.2: BASIC ACADEMIC ABILITY 49
MAP 4.3: PROBLEM SOLVING SKILL AND GOAL ORIENTATION 50
SUMMARY

The aim of the research is to investigate factors that may explain success in elementary computer programming at the tertiary level.

The first phase of the research included the identification of possible explanatory factors through a literature review, a survey of students studying introductory computing, a focus-group session with teachers of computer programming and interviews with programming students.

The second phase of the research that was called the main study, involved testing the identified factors. Two different groups of programming students – one group majoring in business computing and another majoring in computer science – completed a survey questionnaire.

The findings of the research are as follows.

Gender is of little significance for business students but there is an adverse gender penalty for females in computer science. Secondary school assessment is inversely related to outcomes in business computing but directly influences outcomes in the first programming unit in the computer science course. As in prior research, previous knowledge and experience were demonstrated to matter. A range of other variables was found to be of little importance. The research suggests that different problem-solving techniques might be relevant in business compared with those of use in computer science.
CHAPTER 1
INTRODUCTION

1.1 Computer Programming: A Short History

Computing and programming are relatively recent fields. Fifty years ago the first computers were developed and the people programming them were the scientists and mathematicians who used the machines. During the 1960s business began to realise the value of computing. Applications were developed for some procedures such as invoicing, payroll and inventory control. People were employed to write programs and the occupation of computer programmer emerged.

Initially hardware costs were of primary concern. However computer sales increased and with advances in technology, hardware costs decreased. Computer programming became more sophisticated and the development of larger and more complex systems resulted in growing software costs (Tatnell 1994).

Few formal courses for programming existed before 1960. Business corporations often developed their own aptitude tests designed to determine whether a prospective employee had the necessary attributes and skills for programming. If successful in these tests the applicant would then be trained on-the-job. Aptitude tests often consisted of arithmetic reasoning, number series and relationship recognition much like standard IQ tests. Other tests included flow charts requiring the ability to follow step-by-step instructions. They were criticized as having no measurable validity; all they could measure was the trainability of a candidate (Weinberg 1971). Another criticism was that the tests did not measure what programming was about. For example speed was often a factor in the tests but speed is not an essential part of the programming process (Weinberg 1971).

Tertiary institutions began to offer computing courses in the 1960s. The curricula were developed by university mathematics faculties and became known as Computer Science courses. At the same time training courses in programming and analysis and design were developed. These became Information Systems or Business Computing courses.
In Australia, the first computers were found in universities - the universities of Sydney, Melbourne and New South Wales - and training courses were offered in the programming techniques appropriate to each type of machine. A few years later training courses, originally developed for the Public Service Board, were established to teach systems analysis and design and programming (Tatnell 1994).

1.2 Computer Programming Research

Research in computer programming originally focussed on the machine aspects of programming. The emphasis was on improving the programming process using new techniques and tools.

The purpose of these studies was rapid assessment of tools on the sole basis of finished products and no attempts were made to get insight into the activity itself (Hoc et al. 1990 p. 4).

Very little consideration was given to the human side of the programming process. Generally there were no references to psychological theory or methodology in such research.

It has often been stated that Weinberg's book *The Psychology of Computer Programming* (1971) was the first to examine computer programming from a human perspective. Weinberg emphasized programming as a *human activity* and argued that the psychological emphasis was absent from research. The book, essentially the author's opinion and not always substantiated by scientific fact nevertheless started a new field of study - the psychology of programming. Many contemporary areas of research are orientations mentioned in the book - especially those relating to the need to increase knowledge of the cognitive processes underlying the computer-programming task and how programming is learned (Hoc et al. 1990).

The need for a psychological approach and a focus on human cognition in programming was argued from other perspectives (Mayer 1975, Brooks 1977, Tracz 1979, Shneiderman 1980, Sheil 1981). Subsequently research techniques such as protocol analysis and cognitive modelling were adopted from these fields. These methodologies allowed researchers to explore the differences between the mental representations of experts and novices in the programming process. They also shed light on how novices learn to program a computer.
1.3 Background and Significance of This Research

The domain of this investigation is Computer Science Education. It draws on the
disciplines of computer science, cognitive psychology, education, management science
and social science. The research is therefore not confined to a particular framework, and
the guidelines and research methods of a single discipline are not used exclusively.

Computer Science Education has a short tradition (about 30 years) but there is growing
research in the area as educators grapple with the ever-present problem of teaching
computing in a meaningful and effective manner. The SIGCSE (Special Interest Group
in Computer Science Education) is one group that provides a forum for university
educators to discuss concerns and publish research about computer science programs.
Several conferences are held around the world each year including an annual conference
in Australia. Interest groups have emerged on the Web such as Psychology of
Programming (PPIG).

Teaching introductory computer programming presents particular challenges to the
educator. Introductory units are generally seen as hurdles because they differentiate
among those students who can program and those who can’t. In many computing
courses the first programming unit must be completed successfufly for the student to
continue with computing studies. Computer programming must be understood. Rote or
surface learning is not an efficient or effective method for success. Here is an attempt to
explain factors affecting performance in programming studies. Analyzing samples of
programming students drawn from the business and science disciplines allows this.

1.4 Structure of the Thesis

Chapter 2 is the review of previous research. This literature review and the three
chapters that follow seek factors for analysis in the estimation stage of the study.

- Chapter 3 describes a survey of students prior to their computer programming
  studies;

- Chapter 4 describes a cognitive mapping session with a focus group of educators
  involved in teaching computer programming;

- Chapter 5 is an account of eliciting information from students undertaking a first
  unit of computer programming.
The research in these three chapters comprises the Preliminary Study. The results are the framework for the Main Study described in Chapter 6. In Chapter 7 data from the main study are analyzed in a series of regressions. Finally, Chapter 8 draws conclusions and discusses recommendations for further research. A summary research plan indicating the linkages between the stages of the whole study is set out in Figure 1.1
Figure 1.1: The Structure of the Thesis
CHAPTER 2
COMPUTER PROGRAMMING AND HOW NOVICES LEARN

2.1 The Activity of Computer Programming

Computer programming is a diverse and cognitively demanding activity. It is a design task in which the goal is to solve a specified problem using the design language (Green 1990). The problem is usually in another application area such as mathematics, accounting, manufacturing. It is the programmer's task to understand the problem in the application domain and solve it in the implementation domain. This requires knowledge of the application domain and extensive knowledge in the implementation domain. As well as knowledge it also '...involves a variety of cognitive activities, and mental representations related to program design, program understanding, modifying, debugging (and documenting)' (Rogalski and Samurçay 1990 p. 170). According to Rogalski and Samurçay programming activity consists of four related parts: the knowledge structure, problem solving, practice and cognitive tools.¹

CONDITIONS FOR COGNITIVE ACTIVITIES

![Diagram]

Figure 2.1: Frame for Task Analysis

¹ Figure adapted from Rogalsky and Samurçay (1990 p. 159).
2.1.1 Tasks in the Programming Process

Programming consists of a variety of subtasks that involve several kinds of specialized knowledge and different types of cognitive activity. These tasks include:

- statement of the problem
- problem formulation
- design of a solution
- translation of the solution into a programming language
- debugging, testing and verification of the solution (program)
- maintenance and modification

Basic programming tasks described by Pennington and Grabowski (1990) are shown in Fig 2.2. The figure shows some basic processes that are present throughout the tasks: composition and comprehension. The mental processes required for each task and the knowledge domains associated with them are also shown. This characterization of programming supports the Brooks (1983) framework in which the programming process is described as '…serial mapping from one knowledge domain to another, beginning with the problem domain, through several intermediate knowledge domains, and ending with the programming knowledge domain' (Pennington and Grabowski 1990 p. 47).

In the preceding descriptions, programming essentially involves the same tasks. The phases of these follow a linear pattern: each starting on completion of the previous one. In practice however, the phases follow a non-linear pattern. This is particularly true of more complex programs where substantial interactions take place.

During the testing phase, problems and shortcomings are discovered requiring the programmer to return to previous design and coding phases and to repeat the steps. The types of activities performed in each phase are distinct although there is a strong interrelatedness between them.
Figure 2.2: The Tasks of Programming

Specification

The specification phase is one where a problem in the domain of the user/client is analyzed to establish the requirements and determine what functions the program(s) need to perform. Domain knowledge is ‘...critical for providing a context for interpreting requirements, detecting incompleteness, constraining initial design solutions and developing a global design model’ (ibid. p. 48).

Knowledge about the problem domain aids program comprehension; it is important to document the rationale behind the specifications for a program (Brooks 1983). The specification phase requires goal clarification, elaboration and questioning. Expertise and extensive knowledge are essential (Dalbey and Linn 1985).
Goal clarification allows programmers to form representations of the problem. Once clarified the programmer can identify similarities with previous problems and develop a solution plan. The ability to translate mental representations into external representations in the form of texts or diagrams is an aim and crucial outcome of specification.

Design

Design requires creative, selective and broad sweeping minds (Weinberg 1971). Further it is a planning task demanding a systematic, methodical approach and skill in decomposing problems (Newell and Simon 1972, Dalbey and Linn 1985, Scholz and Weidenbeck 1992). Requirements and specifications are decomposed into design strategies and structures and then into more detailed specifications for program code.

Ideally, the strategy for decomposition is a top-down, step-wise refinement where each task is broken down into smaller tasks for which there is a standard programming solution. Often this decomposition task is not straightforward and opportunistic strategies are employed. The programmer may alternate between different levels of planning and abstraction if sub-problems have design faults, particular difficulties, or are critical to success (Visser and Hoc 1990). Planning at lower levels may guide planning at a more abstract level. The characterization of this is more multi-directional than top-down (Pennington and Grabowski 1990). Because '...the program must work correctly for a whole range of input values and respond appropriately to a variety of possible circumstances' (Dalbey and Linn 1985, p. 256) design involves formal reasoning and abstract thinking.

Coding

Successful coding entails extensive knowledge and accurate application of the syntax and rules of a programming language. Thus, '...the mind which is clever at small things now excels' (Weinberg 1971 p. 169). Possibilities for individual interpretation abound. Weinberg and Schulman (1974, p. 715) found that '...a programmer will have an almost infinite number of choices in terms of how they will write a program in order to meet certain specifications'.

Brooks (1980) found that experienced programmers possess a knowledge base of tens of thousands of rules in their programming language. The knowledge is highly organized and importantly solutions to previous problems (algorithms) are stored.
Programming knowledge can be thought of as a collection of units—frames, paradigms or schemata. Each is organized as a program fragment, abstracted to some degree, together with a set of propositions about its behavior and rules for combining it with other fragments. They are indexed in terms of the problem classes for which each applies (Sheil 1981). When coding a program the appropriate algorithm needs to be retrieved and applied.

Coding requires the ability to efficiently access knowledge of the programming language from memory and the ability to formulate precise expression (Dalbey and Linn 1985). Acquisition of a large body of diverse knowledge, or a limited body of knowledge in diverse situations, may take many years (Brooks 1977).

Debugging
This phase consists of two main parts -
• testing - written programs are tested for errors
• debugging - errors found in tests are eliminated

As much as 75 percent of a programmer's time is thought to be involved in identifying and correcting mistakes. Personnel who were not involved in writing particular programs often carry out testing and debugging. Appropriate personnel are technically competent to understand the code including the meaning and purpose of program statements, the execution sequence and the data flow (Pennington and Grabowski 1990). They should possess refined abilities in deduction, inference, responding to feedback and identifying commonalities and inconsistencies (Dalbey and Linn 1985).

Testing is a diagnostic task. It entails generating and evaluating hypotheses concerning the problem. Once an error is detected it is repaired and the system tested again (Pennington and Grabowski 1990). Testing therefore involves systematic and methodical approaches.

Professionals responsible for testing must judge whether a program executes as expected. To do this they must understand what code does and evaluate this against what it is supposed to do (Détiéanne 1990). Assistance is available in the form of testing tools, like test-data generation. However logic and reasoning activities, such as simulation of program execution and devising hypotheses, also play a role.
Youngs (1974) distinguished four types of programming errors: syntactic, semantic, logical and clerical (typographical). Some are found easily. Syntax errors are generally flagged by a compiler or other diagnostic tool and are resolved using the language manual. Semantic errors and logic errors require more effort and time (Youngs 1974). Finding these requires investigative skills and diagnostic strategies with a ‘...network of knowledge about which kinds of errors and programmer actions result in which kinds of program bugs’ (Pennington and Grabowski 1990, p. 56).

The ability to recognize common bugs is one of pattern recognition. Success at debugging depends on the choice of strategy as much as possessing the relevant knowledge structures (Gilmore 1990). Weinberg (1971) thought a special gift – an *eye for wholeness* – is necessary when debugging. The gift is the general feeling that something is out of place when in fact it is.

### 2.1.2 Skills in Programming Tasks

Each phase of the programming task therefore requires various cognitive skills, varying in complexity. Some appear to be opposite in nature. The programmer is expected to display creativity and think laterally. During other phases of the programming process, the programmer needs to be pedantic, precise and pay attention to detail. The programmer must decompose some problems and synthesize others. A good programmer should follow a top-down approach by defining what needs to be done at a higher level and how it is to be done at a lower level. Alternately the approach is bottom-up when working routines (modules) are combined to construct a working program.

It is advantageous to break programming processes into separate tasks and identify the cognitive skills required to achieve each. Yet most programming tasks are not a succession of sequential and continuous tasks. Programmers often have incomplete knowledge about a task and must elaborate their understanding by returning to a previous task. For example the activity of coding may force the programmer to reconsider how some components were designed (Pennington and Grabowski 1990). This interrelatedness between the programming tasks calls on the programmer to perform at different levels of abstraction.
The smooth transition from one task to the next is sometimes interrupted by the necessity to incorporate modifications to the original specifications. These arise because the user requests them, because of external pressures, such as new legislation or business mergers, or because new hardware or software are introduced. These influences induce uncertainty. Programmers must adapt, allow for change and take decisions with incomplete information.

2.1.3 Programming Tasks and the Environment

Rarely does one professional do the specification, design, coding and debugging of a large program. In practice the work is shared among a team. Shared understanding of the problem, the design, the code and other issues must be established among team members. Although documentation is kept, knowledge sharing via informal communication occurs (Pennington and Grabowski 1990). These channels are particularly important where different teams have distinct tasks in the programming process.

2.1.4 Measuring Programmer Performance

Performance measurement is difficult. One reason is that some development projects are large and usually complex. Another reason is the difficulty of defining a 'good' program. Obviously a program must perform the task without error but it is difficult to define good and then judge how good a program is. Generally accepted benchmarks are:

- ease of maintenance and modification
- efficient use of resources, such as execution time
- convenience and utility of the program
- effectiveness of the user interface

Quantitative measures of programming performance have been developed such as McCabe's (1976) measure of cyclomatic complexity and Halstead's (1977) metric. Also used are indicators such as lines of executable code and function points.

2.2 Computer Programming and the Novice

The psychology of programming focuses on how novices acquire programming skills. Much of the research concerns differences between experts and novices and the reasons the differences exist.
The meaning of 'novice' varies in the literature. It could mean an individual who has never used a computer, or a student who has done one or more programming units.

2.2.1 Chunking Information

The approach to cognitive differences between novices and experts was initially adopted from research on chess skills (Chase and Simon 1973). Master chess players are able to replicate briefly presented, mid-game board positions. This is a result of chunking information. Less experienced players use far smaller chunks. It was suggested that experts organize chunks hierarchically. Experts do not perform better than novices when chess pieces are randomly positioned (Bateson et al. 1987).

Shneiderman (1976) replicated this study using programming experts and novices. The interpretation is that

... the expert has no better memory than the novice does, but rather an elaborate knowledge structure in terms of which correspondingly structured items can be very efficiently encoded (Sheil 1981 p. 706).

The notion of chunking is supported by Tracz (1979), Shneiderman and Mayer (1979), Adelson (1984) and Allwood (1986). A study by McKeithen et al. (1981) concluded that subjects with expert knowledge use remarkably similar ways of organizing useful information chunks. These differ from the chunking and organization of novices.

2.2.2 Cognitive Models

Another approach to how novices assimilate programming knowledge involves models of long-term/short-term memory (Tracz 1979, Brooks 1977, Shneiderman 1980). There are differences but for each model:

- information enters the cognitive system via short-term memory
- information from long-term, stored memory is retrieved and integrated with the short-term memory
- the integrated information is used

The components shown in Figure 2.3 are based on Shneiderman (1979). Short-term memory capacity is relatively limited, perhaps to seven chunks (Miller 1956), while long-term memory is unlimited. The Shneiderman model includes working memory that is less permanent than long-term but more permanent than short-term memory. Working memory is where the integration of long-term and short-term memory takes place.
Underlying Schneiderman's approach is Bartlett's (1932) cognitive theory of understanding and remembering.

![Long-term Memory Model](image)

**Figure 2.3: Long-term Memory Model**

### 2.2.3 Semantic versus Syntactic Knowledge

Syntactic knowledge is the knowledge of a programming language required for coding and implementation. Syntactic knowledge is arbitrary, instructional and acquired by rote. Semantic knowledge is acquired by intellectual, demanding, meaningful learning and allows additions to knowledge. Problem analysis requires semantic knowledge. It is *independent* of a programming language. Experts tend to use a semantic knowledge to understand problems; novices tend to use syntactic knowledge (Shneiderman 1980, Atwood and Ramsay 1978, Adelson 1984).

Magliaro and Burton (1988) showed that adolescents learn and organize meaningful knowledge about programming in much the same way as adults do. Those with more programming experience develop expertise in recalling *semantic* rather than *syntactic* procedures. Mayer (1975, 1981) suggested through physical or mental models, novices learn the semantics of programming and abstract semantic knowledge. According to Shneiderman and Mayer (1979) long-term memory consists of both semantic and syntactic components.
2.3.4 Programming Plans and Strategies

Programming plans are schemas in which the main elements of a program have been abstracted and presented explicitly as action sequences. Experts use plans as knowledge structures in which information relating to a function is grouped (Soloway and Ehrlich, 1984). When presented with new problems, experts retrieve plans from long-term memory and integrate them into solutions. Provided program statements are grouped meaningfully, experts can recall modules better than novices can because they retrieve the appropriate programming plan. Novices do not form programming plans. When program statements are presented in random order novices do as well as experts (ibid.).

As well as programming plans, expert programmers also have rules of programming discourse. Soloway and Ehrlich (1984) found that when test programs were plan-like advanced programmers performed better than novices; their recall ability is earlier for programs with plans (compared with programs that do not); and knowledge of programming plans and rules of programming discourse can have significant impact on program comprehension. Bateson et al. (1987) classified programming plans as:

- strategies specifying the overall structure of an algorithm
- tactics for solving problems
- implementations which translate strategies and tactics into program statements

Rist (1986) modeled program generation using a top-down approach when knowledge is available, and a bottom-up process (focal expansion) when knowledge is not available. Experts retrieve stored schemata and a programmer's strategy will change from focal expansion to schema expansion as expertise develops (Davies 1993 on Rist).

Programming experience determines which strategies are selected and utilized. It is more important to know when to use them rather than having them. Practice leads to mastery. As well as programming plans, successful programmers use cross-referencing strategies that alternate between the program and the real-world problem (Gilmore 1990). Experts whose comprehension of programming is high employ these fundamental strategies (Pennington 1987). Difficulties arise when a programming language construct requires a cognitive strategy that differs from the preferred strategy.
Soloway and Ehrlich (1984) investigated novice, intermediate and advanced programmers. They found that programmers wrote programs correctly more often when languages matched their cognitive strategy. Accuracy, sensitivity and preference for a particular strategy could shift with experience. A problem arises when programmers program in an unknown language; they adopt existing programming plans, appropriate in the known language, to obtain a working program. These might fail to take advantage of the strengths of the new language. This may occur if the new language is flexible enough to cope with less than ideal plans (Scholtz and Wiedenbeck 1992). Pre-existing, high-level and language-independent plans may not provide a good fit to the new programming language.

2.2.5 Why Don’t Novices Do Well?

Widowski and Eyferth (1986) compared the performances of experts and novices using both stereotypical (semantically simple) and atypical (semantically difficult) programs. Experts read more flexibly. They read stereotypical code in long but infrequent runs; and, atypical programs in short and frequent bursts. These findings ‘...suggest that the former strategy reflects a top-down or a conceptually driven comprehension process, while the latter represents a bottom-up and heuristically oriented strategy’ (Davies 1993 p. 247). Novices tend to employ the same strategy for both types of programs.

Perkins and Martin (1986) coined the phrase fragile to describe knowledge a novice has but fails to use when appropriate. In a protocol analysis Gilmore (1990) showed that novice students know plans, but they cannot use them successfully. Gilmore described the efforts of one student who oscillated between iterative and recursive plans five times before a simple error was fixed.

Perkins et al. (1988) note two further difficulties among novice programmers:

- a deficit of elementary problem-solving ability
- attitudinal problems affecting confidence and control

In the same vein Doerner (1980) refers to their inability to pursue a chain of thought to a natural conclusion. When a novice is successful in one area, that area is overworked and cognitive forces are not reallocated. These individuals blame the outside world, make fewer plans and lack reflective capacities.
These deficiencies mean novices lack application strategies and do not understand the need for persistence and precision. Compare this with spelling. Misapplication of a rule or failure to apply obscure rules might reduce the grade to 90 percent in a test. That might be regarded as acceptable. Perkins et al. (1988) suggest that a program in which 90 percent of statements are correct is of an unacceptable standard. Further, there may be several interacting errors and debugging will be difficult to achieve.

To become experts novices must often acquire or refine unfamiliar cognitive skills. Novices may have cognitive difficulties because the programming domain is dissimilar to everyday environments, the computer's operation may not be transparent, they only naively link the context of a problem with abstract approaches to its solution, they lack semantic knowledge or they are deficient in solution strategies. Experts form abstract, conceptual representations of problems describing the operations to be performed. These representations are easy to work with, to modify, and allow experts to find optimal solutions (Adelson 1984). However, novices form concrete representations retaining elements of why the problem works.

Pedagogic suggestions are given in Rogalski and Samurçay (1990) and Shackleford (1993). Learners need conceptual representations, particularly for structuring data and modelling problems. By comparison processes are relatively easily mastered. Further programmers should know about a variety of paradigms and approaches, such as a range of looping strategies.

2.3 Computer Programming and Individual Characteristics

2.3.1 Mathematics Ability

Initially many university courses in programming were in mathematics departments. It was assumed that mathematics aptitude equated with aptitude for programming. The perception was that computers were essentially number crunchers.

The research findings here are divided. Alsopagh (1972) found that a mathematics background was a major influence in programming achievement because the programs were about mathematics. There is evidence suggesting a relationship between mathematics-related variables and computer-programming achievement (Mayer 1975, Soloway et al. 1982, Nowaczyk 1984).
An improvement in understanding mathematics may improve achievement in general mathematics, problem-solving and programming among younger novice programmers (McCoy and Burton 1988). As noted in Chapter 7 the quantitative evidence in the current research does not support a link running from mathematics to programming competence.

2.3.2 Gender

In Maccoby and Jacklin’s (1974) review of the literature on sex differences, boys scored higher on mathematical and spatial ability. Combined with the perception that mathematics is strongly related to computing, it appeared that females were not suited to programming. Fennema’s (1983) analysis of the weaknesses in Maccoby and Jacklin’s research contradicted this conclusion. (See also Thronson 1984; Guinan and Stephens 1988; Hattie and Fitzgerald 1987; Clarke 1990, Clarke and Chambers, 1989). Females do as well as males in programming but social and cultural differences are deterrents (Dalbey and Linn 1985, Hattie and Fitzgerald 1987, Ethington, 1988).

The image of computing is predominantly male (Clarke 1990). In the current research it appears that some learning environments may not be so male oriented.

2.3.3 Personality Factors

The requirements of computer programming were discussed in Section 2.1, but there is some further evidence that should be examined. Teague (1996) reports that a high proportion of maintenance and development programmers prefer to work alone. They are careful about detail, are logical and are well organized. Teague (ibid. p. 160) notes that ‘...where traditional programming has become less important and programmers are expected to perform analysis and design tasks, there is a place also for conceptualizing and planning skills’.

The Ability to Tolerate Stressful Situations

Programmers’ work often means that tight, externally imposed schedules need to be maintained (Weinberg 1971). Maintenance tasks where modifications are necessary to existing programs are potentially very stressful; they rely on the programmer being able to achieve an emergency solution quickly and accurately. The capacity to work under and tolerate stress is required (Shneiderman 1980).
Neatness and Precision

For Weinberg (1971, p. 150) a modicum of neatness allows a programmer to organize documentation and reference resources when required. Shneiderman (1980) emphasized precise attention to detail: neatness and precision in design, coding and documentation provides better communication for others interacting with the programmer.

Humility

Humility is being without a high ego (Shneiderman 1980). It is essential (Weinberg 1971). Its absence can adversely affect the performance of a programming team.

Assertiveness

Assertiveness enables a person to get things done (Weinberg 1971). It is also the ability of a programmer to see a task through, ensuring that difficulties are overcome.

Anxiety

Computer programming is demanding and anxiety can interfere with ability to concentrate on the task. According to Shneiderman (1980) a moderate level of anxiety is best. A highly anxious person is more likely to make mistakes. A study of 117 undergraduates in which computer anxiety was measured against performance in a programming task found that individuals who are lower on computer anxiety generally achieved higher scores on the programming task (Chen and Vecchio 1992).

Character Traits

- People with high reflectiveness, low impulsiveness and low sociability are more likely to succeed in programming (Alspaugh 1972).

- Introverted people can think things through, although extroverts are often valuable members of programming teams (Shneiderman 1980, Teague 1996). People low on extroversion tend to perform better on certain programming tasks (Chen and Vecchio 1992).

- Risk-taking along with open-mindedness may be important (Shneiderman 1981).

2.3.4 Motivation

Motivation drives people to improve performance. In a complex task performance can be increased but will fall away rapidly. Motivating factors are money, participation in goal setting or quality of work in programming itself (Weinberg 1971). Highly motivated individuals are better at programming (Shneiderman 1980).
Kearney (1969) found that Australian students with well- or fairly well defined goals performed about equally in their courses; those who could not identify their goals performed significantly worse. Pentony (1968) studied students with home difficulties and noted a general lack of sense of purpose and direction.

In a study by Crebbin et al. (1994) when students were asked which features distinguished tertiary study from secondary school, 70 percent nominated self-motivation and individual responsibility. Beard (1980 p. 87) cites a study in which motivation was a measure of short and long-term goals and reasons for entering university.

Some major determinants of motivation in university students are the suitability of courses to their varied needs, their initial experiences and the adoption of study methods appropriate to higher education (ibid. p. 191).

2.4 The Role of Problem-Solving in Computer Programming

Problem solving means a number of things depending on context. Researchers from behavioral, cognitive and philosophical perspectives consider problem solving. Early studies focused on pattern recognition, cryptarithmetic, puzzles, chess problems and the like. Subsequent studies have focussed on problem solving in domain areas such as mathematics, physics and artificial intelligence.

Problem solving is required in many aspects of programming tasks. It arises when the programmer conceptualizes a problem and translates it into an appropriate algorithmic solution. Applying the appropriate solution from semantic knowledge is acquired through deep learning. According to Shneiderman and Mayer (1979) this process requires problem-solving ability. Gray and Anderson (1987) argue that coding a computer program is a problem-solving process where the problem statement represents the initial state and the completed program is the goal state. Code comprehension also requires problem-solving ability.

There is consensus that problem solving is a goal-directed activity that involves a sequence of stages. Polya (1957) defined these stages as:
• understanding the problem
• devising a plan
• carrying out the plan
• evaluation of the solution – looking back.

During each stage the solver uses analysis and reasoning based on understanding the
domain of the task. Shute’s (1991) study of problem solving in the programming
environment investigated the roles of understanding using algebra-word problems.
Problem identification and sequencing were predictors of performance in acquiring
programming skills.

Some aspects of problem solving are similar across domains; other aspects are clearly
domain specific. Transferability of problem-solving skills from one domain to another is
a weak method because it is not as effective or efficient as the domain-specific strong
methods (Perkins et al. 1991).

Breaking a problem into its component parts or trial-and-error analysis are examples of
general methods. They provide much less leverage in problem solving compared with
domain specific methods, where a large repertoire of local knowledge has been
compiled into schemata.

Smith (1991) found external factors affecting problem-solving performance to be
problem context, structure and the social environment of the problem solver. Internal
factors are domain-specific knowledge, general problem-solving knowledge, experience
and personal characteristics such as self-confidence, enjoyment, motivation, cognitive
development and field dependence.

A field-independent individual is one whose perceptions are analytical and is therefore
not dominated by the prevailing field. A field dependent individual generally chooses to
work in areas that feature tasks requiring involvement with people (Witkin 1977).

Much of the empirical research in problem solving has been motivated by Newell and
Simon’s 1972 information processing theory. In this theory human problem solving is a
complex mental activity involving the interaction between a task environment and the
individual problem solver.
Burton and Magliaro (1988) suggested that the problem solver evaluates all the possible choices and strategies to reach a solution by assimilating facts and processes.

2.5 Achievements in Programming Courses

There is a direct relationship between self-perception of programming ability and achievement (Thronson 1984). Many researchers attempted to identify the attributes of good programmers (Du Boulay 1989, Lee and Pennington 1994). In a series of applied studies Kagan (1988) examined 1009 students undertaking programming and computer literacy courses. A tendency to be hard-driven, ambitious and time-urgent gave students of computer literacy a temporary edge. Conversely these traits grew more significant for programming students over time. As studies of programming progressed, greater motivation and perseverance were required.

Kurtz (1980) measured abstract reasoning ability in introductory programming students. This measure was directly correlated to their performance in programming tests. Subsequently Barker and Unger (1983) found weaker support for this. Success in first-year programming may be associated with attitudes, motivation, general learning skills, reasons for choosing courses and financial position (Jones 1983). Past computer experience and academic performance in English and mathematics were predictors according to Nowaczyk (1984).

Shute (1991) found programming ability depended on working memory, skills in algebra, problem identification and sequencing and active learning. Chen and Vecchio (1992) suggested that personality and aptitude constructs partially account for performance in programming.

2.6 Looking to the Preliminary Study

In this chapter theories underlying the aims and methodology of the thesis were reviewed. In subsequent chapters, for each aspect of the study, the reviews are further extended. For example in the next chapter business and science students taking an introductory unit in Information Technology are surveyed. Questions for the survey were motivated by the research reviewed in the current chapter and by specific papers cited in Chapter 3.
CHAPTER 3

THE PRELIMINARY SURVEY

3.1 Introduction
The survey described in this chapter was the first stage of a preliminary study (see Figure 1.1). It sought to explore the relationships between some of the factors identified from the literature as being significant indicators of performance in introductory units on computer usage and software.

The objective of the survey was to determine whether background and other factors were important in samples drawn from students in two units. This chapter describes the preliminary survey and the main findings. Indicators for the later research are highlighted. The survey is included as Appendix A.

3.2 Background
Two groups of students participated in the survey. The first group consisted of students who enrolled in a business faculty. Some of these students were taking a six-unit business computing major over three years. The second group consisted of students enrolled in a science faculty, some of whom were majoring in computing. Their major study was eight units of computing over three years. It should be emphasized that there were both computing and non-computing majors within each student group.

The students were undergraduates and most were in their first year of study at university. Although the business and the science students studied different computing units in their first year, the units were very similar, especially in terms of the practical component. Both groups used the same integrated software package consisting of a word-processor, a spreadsheet and a database. Competency in using the packages was the major criteria for practical assessment. Neither unit had a computer-programming component. The groups of students and the units of computing they studied are described further.
Group 1: Business Students and Business Information Systems

These students were undertaking degrees in business with majors such as accounting, marketing, human resources and business computing. All business students undertake a common sequence of units in the first year of their studies including the introductory unit, Business Information Systems. In the second semester of their first year, students commence studies relating to their course major. For example, those majoring in business computing undertake a unit of computer programming. More units relating to course major are taken in the second and third years of the course.


There are two different independent sections of the unit. The first sections aims to develop an appreciation of the importance of and some of the roles played by information technology in the world in general and, more particularly, in business. The second section aims to equip students with general competency using microcomputers.

Business Information Systems was delivered in the form of one two-hour lecture and one two-hour practical (workshop) session per week. The assessment was by examination (50 percent), assignment (45 percent) and workshop achievement (5 percent).

Group 2: Science Students and Introduction to Information Technology

These students were undertaking degrees in science or applied science with majors such as biology, chemistry and computer science. In the first semester of their studies they undertook an introductory unit, Introduction to Information Technology.

Introduction to Information Technology was delivered in the form of two one-hour lectures and one two-hour practical session per week. The assessment was by examination (60 percent) and progressive assessment (40 percent). According to the Deakin University Undergraduate Handbook (1994 p. 435) the content of the unit consisted of:
Introduction to information technology  Computer networks
How to start a computer  Information systems in organizations
Architecture of computer systems  Computer and society
Operating systems and their roles  Professional ethics
Approaches to systems development

3.3 Research Method

Permission to survey students was obtained from the Deakin University Ethics Committee. The surveys were mailed out with accompanying letters asking students to volunteer for the study. Surveys were sent to almost 60 percent of the groups of students enrolled in the two units in Week 9 of the 13-week semester. Students were selected randomly from enrolment lists with numbers provided by a random number generator function in a spreadsheet package. The surveys were anonymous and individual students could not be identified from their responses. The number of surveys and the returns are summarized in Table 3.1.

<table>
<thead>
<tr>
<th></th>
<th>Enrolled Numbers</th>
<th>Surveys Mailed</th>
<th>Surveys Returned</th>
<th>Percentage Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Business Students</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-computing</td>
<td>293</td>
<td>166</td>
<td>58</td>
<td>35</td>
</tr>
<tr>
<td>Computing</td>
<td>61</td>
<td>36</td>
<td>16</td>
<td>44</td>
</tr>
<tr>
<td><strong>Science Students</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-computing</td>
<td>236</td>
<td>140</td>
<td>60</td>
<td>43</td>
</tr>
<tr>
<td>Computing</td>
<td>56</td>
<td>33</td>
<td>11</td>
<td>33</td>
</tr>
</tbody>
</table>

Table 3.1: Survey Mail-outs and Returns

Among the 354 enrolled business students, 17.2 percent were enrolled in computing majors. For science students there were 292 students, of whom 18.9 percent were undertaking computing majors.
The random sampling procedure was stratified to preserve these relative weightings of computing/ non-computing students when surveys were mailed. The sixteen responses received from the business computing students represents 21.6 percent of completed and returned surveys. That is, computing majors are over represented compared with their weight in enrolments (17.2 percent). On the other hand the eleven responses received from the computing majors in science degrees under represent the computing majors in science enrolments. The eleven returns were only 15.5 percent of all replies (compared with a population weight of 18.9 percent).

3.4 The Survey

The survey was designed to provide details on student demographics such as gender, age, course major and secondary school performance so that student profiles could be determined. The survey was also designed to measure:

- Computer anxiety
- Computer playfulness
- Locus of control
- Computing experience/ knowledge

Anderson (1992) previously used some of these measures on other groups of students studying Business Information Systems. The measures had therefore been trialled and tested. Not all parts of the Anderson (1992) survey were included. For example, questions relating to the value of lectures and tutorials were omitted.

3.4.1 Computer Anxiety

Computer anxiety is a term that has been used to express the sense of fear that people who are unfamiliar with computers can experience. It is an intense anxiety about the use of computers that can produce unpleasant emotional reactions. Raub (1981) defined computer anxiety as the tendency of an individual to be uneasy, apprehensive, or fearful about the current or future use of computers in general. Raub (1981) developed an instrument to measure computer anxiety and her anxiety scale has been employed extensively in studies on computer anxiety (Howard 1986, Igbaria and Parasuraman 1989, Parasuraman and Igbaria 1990).
Prior to the mid-1980s, computer anxiety and attitudes towards computers were treated as synonymous concepts (Igbaria and Parasuraman 1990). However Howard’s research in 1986 which sought to uncover the psychological mechanisms triggering computer anxiety, suggested that computer anxiety could predict attitudes towards microcomputers.


Following Anderson (1992), Howard (1986) and Howard and Smith (1986) the measure for anxiety was adapted from Raub (1981). There are 13 statements, such as

6. I hesitate to use the computer for fear of making mistakes
7. I am unsure of my ability to interpret a computer printout

Respondents were asked to circle the rating that best described their feeling about or reaction to each statement. A five-point Likert scale was used with ratings from:

1 = Strongly Agree
2 = Agree
3 = Unsure
4 = Disagree
5 = Strongly Disagree

3.4.2 Computer Playfulness

The term computer playfulness, more specifically microcomputer playfulness, is the degree to which a person displays cognitive spontaneity in microcomputer interactions (Webster and Martocchio 1992). It is perhaps one of the most important aspects of human computer interaction, described as a possible indicator of a person’s capacity to master a computer and computing generally (Anderson 1992). According to Webster and Martocchio (1992) playfulness is both a character trait, a comparatively stable characteristic of an individual; and a situation specific trait. A person who is not normally playful may exhibit playfulness with a computer in particular situations.

---

2 See Section A of the survey in Appendix A.
A subsequent study by Martocchio and Webster (1992) demonstrated that those higher in cognitive playfulness exhibited higher learning, positive mood and satisfaction with feedback than those lower in cognitive playfulness.

The Computer Playfulness Scale (CPS) developed by Webster and Martocchio (1992) was shown to have good psychometric properties and was trialled on three survey studies and two training studies. For the current survey the 7 point Likert scale of the original CPS was reduced to a 5-point scale following Anderson (1992). There were 22 statements asking respondents to circle the rating which best matched a description of themselves when reacting with computers. Examples of these statements include serious, creative, curious, investigative (see Section B of the survey).

3.4.3 Locus of Control

Rotter (1966) promoted the notion of locus of control. Locus of control refers to a person’s perception of the world. Individuals with an external locus of control have a perception that forces and circumstances beyond their control govern life. Individuals with an internal locus of control are very much self-directed individuals who attribute outcomes to their own efforts. Rotter outlined the relationship between locus of control and the need for achievement. People with a high need for achievement, will probably have some belief in their ability to determine the outcome of their efforts.

According to Howard (1986) external locus of control is a significant correlate of attitudes towards microcomputers, although not directly correlated with computer anxiety. Other research has shown that external locus of control is directly correlated with computer anxiety (Dambrot et al. 1985, Igbaria and Parasuraman 1989).

The locus of control instrument of Rotter (1966) consists of twenty-nine pairs of statements in which respondents were asked to choose one of the two statements which best described what they believed to be true (See Section C of the survey). For example question 11 is:

\[ a \quad \text{Becoming a success is a matter of hard work, luck has little or nothing to do with it} \]

\[ b \quad \text{Getting a job depends on being at the right place at the right time.} \]

An individual with an external locus of control is likely to believe \( b \) to be true. Conversely a self-directed respondent is likely to believe \( a \) to be true.
3.4.4 Computer Experience and Knowledge

Experience with using computers and knowledge of computing may come from formal computing classes or it may result from informal activities. Clarke and Chambers (1989) describe some of these activities as: home ownership and usage, entering computer contests, playing arcade games, attending summer camps and courses and memberships of computer clubs. Experience and knowledge has also been described as computer usage, typing skills and computer competence measured using self-rated skills in using computers (Webster and Martocchio 1992). Howard and Smith (1986) suggested that lack of knowledge about capabilities and limitations of computers were likely to raise operational fears on how to use them. Chen and Vecchio (1992 p. 845) concluded that:

... computer background had significant effects on the success of end-user computing. Other results also indicate that computer experience and training are inversely related to computer usage.

Igbaria and Parasuraman (1989) suggested that the level of formal education influenced attitudes towards microcomputers (such as whether they improve efficiency or broaden information sources) indirectly through its effect on computer anxiety.

The questions in this study, relating to computer experience and knowledge were drawn from the Anderson study (1992) and related to experience with microcomputers and software, overall computing and programming knowledge, use of computers in employment and reasons for liking and disliking computers.

There were twelve questions that asked respondents about their computing experience and knowledge (see Section D of the survey). Respondents were asked whether:

- they had access to a microcomputer at home
- the computer had been bought because of tertiary studies
- whether or not the computer could be networked (had a modem).

The aim was to get an idea of the level of sophistication. Respondents were asked to indicate whether or not they had any prior experience with Microsoft Windows, spreadsheets, word-processing packages, database programs and computer-programming languages. They were asked to describe their overall computing knowledge and programming knowledge.
The students were asked to indicate whether they used computers in paid employment. Information on the type of involvement was sought if the answer was affirmative. Finally they were asked what they liked most (and least) about computers and computing. Most questions required a yes or no answer. Some questions required a rating response on a five-point Likert scale. The questions relating to reasons for liking and disliking computers required written responses.

3.4.5 Background Variables

Respondents were asked to provide details of gender, age, course major and whether enrolled full- or part-time. They were also asked to provide details of subject results at secondary school, their reasons for choosing the course and the University. They were asked what initial preference they gave for the course they were in; and whether they had attempted Business Information Systems or Information Technology previously.

Igbaria and Parasuraman (1989) report a complex pattern of interrelationships between background variables, anxiety, experience, knowledge and attitudes. Choices of major, university and preference rankings were not investigated in earlier studies. The reason for seeking this information was to establish a de-facto indicator of student motivation.

It might be argued that students wanting to come to Deakin University, who wanted to do this particular course and who made it their first preference, were more motivated than those students doing the course because it was ‘the only one they could get into’. Each question required the respondents to tick one of a number of boxes. Some questions were open-ended and required brief written answers.

3.5 Results of the Preliminary Survey

The surveys were collated, an identification number was written on each of them and the data were keyed in for use with SPSS. Descriptive statistics were obtained and a series of correlations was calculated.

---

2 See Section E of the survey in Appendix A.
### 3.5.1 Demographics of the Two Groups

Responses from the two groups relating to their background are summarized in Table 3.2. The responses relating to the question on course major were transcribed into a variable coded as 1 if the respondent was enrolled in a computing major and 0 if the respondent was not. Questions requiring written responses were not coded and the responses to the questions relating to secondary school performance were found unsatisfactory for coding.

<table>
<thead>
<tr>
<th></th>
<th>Business Students</th>
<th>Science Students</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number (Percent)</td>
<td>Number (Percent)</td>
</tr>
<tr>
<td></td>
<td>n = 74</td>
<td>n = 71</td>
</tr>
<tr>
<td><strong>Age</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Less than 20</td>
<td>53 (71.6)</td>
<td>52 (73.0)</td>
</tr>
<tr>
<td>20</td>
<td>11 (14.9)</td>
<td>9 (12.7)</td>
</tr>
<tr>
<td>21</td>
<td>5 (6.8)</td>
<td>3 (4.2)</td>
</tr>
<tr>
<td>Total less than 22</td>
<td>69 (93.2)</td>
<td>64 (90.1)</td>
</tr>
<tr>
<td><strong>Gender</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Male</td>
<td>41 (55.4)</td>
<td>27 (38.0)</td>
</tr>
<tr>
<td>Female</td>
<td>33 (44.6)</td>
<td>44 (62.0)</td>
</tr>
<tr>
<td><strong>Enrolled in:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a computing degree</td>
<td>16 (21.6)</td>
<td>11 (15.5)</td>
</tr>
<tr>
<td>a non-computing degree</td>
<td>58 (78.4)</td>
<td>60 (84.5)</td>
</tr>
<tr>
<td><strong>Study Mode</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Full-time</td>
<td>67 (90.5)</td>
<td>69 (97.2)</td>
</tr>
<tr>
<td>Part-time</td>
<td>7 (9.5)</td>
<td>2 (2.8)</td>
</tr>
<tr>
<td><strong>Preference for Degree</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>First</td>
<td>31 (53.4)</td>
<td>36 (57.1)</td>
</tr>
<tr>
<td>Second</td>
<td>14 (24.1)</td>
<td>8 (12.7)</td>
</tr>
<tr>
<td>Third</td>
<td>9 (15.5)</td>
<td>9 (14.3)</td>
</tr>
<tr>
<td>Total of preferences 1 - 3</td>
<td>54 (93.0)</td>
<td>53 (84.1)</td>
</tr>
</tbody>
</table>

1 16 business students and 8 science students did not respond to this question

Table 3.2: Student Profiles
The profiles of the two student groups are similar in terms of age. More than 70 percent of each were under 20. Females make up less than half the business group, but exceed 60 percent of the science group. The larger numbers of females in science is due largely to the predominance of females enrolled in a Health Promotion degree. As noted earlier, twenty-two percent of the business group was enrolled in computing majors, compared to 16 percent of the science students.

The proportion of part-time to full-time students was slightly higher for the business group as the business faculty offered many units, including Business Information Systems, at night. The science faculty did not offer this opportunity. Over 90 percent of the business students had given the course that they were enrolled in as preference one, two or three from seven possible preferences. Proportionately fewer science students (84 percent) gave preferences of one, two or three. Twice as many business students as science students did not respond to the question on course preferences. It is possible that most no longer recalled their preferences. Whether this is a source of bias could not be determined.

3.5.2 Computer Knowledge and Experience

Responses to questions on computer knowledge and experience (Questions 1, 2, 4, 5, and 6 in Section D of the survey) are summarized in Table 3.3. Question 5 consisted of several parts relating to experience with software. The respondents indicated whether their knowledge was none, a little, passable, quite good or superior. Where the answer was passable or higher the corresponding experience variable was coded as one. Where the respondent indicated no knowledge or a little knowledge the variable was coded as zero.

From Table 3.3, it is clear that the proportion of business students claiming ownership and software experience (Questions 2, 4 and 5) exceeds the proportion of science students who did so. There is little difference in the responses to the question on access (Question 1). Question 6 asked about overall knowledge of computing. It appears that similar numbers in the two groups rated themselves as having passable, little or no knowledge of computing.
Slightly more of the business students rated themselves as quite good or superior. Overall, on the evidence of Question 6 it would appear therefore that the business students are somewhat more likely to have higher levels of computing knowledge. However, the previously reported over representation of computing majors among respondents from the business (compared with under representation of computing majors among respondents from science) might account for this.

<table>
<thead>
<tr>
<th></th>
<th>Business Students</th>
<th>Science Students</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number (Percent)</td>
<td>Number (Percent)</td>
</tr>
<tr>
<td></td>
<td>n = 74</td>
<td>n = 71</td>
</tr>
<tr>
<td>Q1. Access to a computer¹</td>
<td>59 (79.7)</td>
<td>57 (80.5)</td>
</tr>
<tr>
<td>Q2. Computer acquisition because of studies</td>
<td>24 (32.4)</td>
<td>19 (26.8)</td>
</tr>
<tr>
<td>Q4. Modem with computer¹</td>
<td>11 (14.9)</td>
<td>4 (5.6)</td>
</tr>
<tr>
<td>Q5. Software experience</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Windows</td>
<td>40 (54.1)</td>
<td>34 (47.9)</td>
</tr>
<tr>
<td>Spreadsheets</td>
<td>47 (63.5)</td>
<td>36 (50.7)</td>
</tr>
<tr>
<td>Word-processing</td>
<td>58 (78.4)</td>
<td>52 (73.2)</td>
</tr>
<tr>
<td>Database</td>
<td>33 (44.6)</td>
<td>27 (38.0)</td>
</tr>
<tr>
<td>Q6. Self-evaluated knowledge¹</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Missing</td>
<td>0 (0)</td>
<td>1 (1.4)</td>
</tr>
<tr>
<td>None</td>
<td>7 (9.5)</td>
<td>9 (12.7)</td>
</tr>
<tr>
<td>A little</td>
<td>14 (18.9)</td>
<td>14 (19.7)</td>
</tr>
<tr>
<td>Passable</td>
<td>22 (29.7)</td>
<td>21 (29.6)</td>
</tr>
<tr>
<td>Quite Good</td>
<td>28 (37.8)</td>
<td>25 (35.2)</td>
</tr>
<tr>
<td>Superior</td>
<td>3 (4.1)</td>
<td>1 (1.4)</td>
</tr>
</tbody>
</table>

¹One business student and two science students did not respond to this question

Table 3.3: Computer Knowledge and Experience
Further investigations of the distributions of self-evaluated knowledge shows that the
great majority of responses from computing majors were from students who rated
themselves as 'quite good' or 'superior'. This was the case for both the science and the
business students. If the non-responses from science students taking computing include
a high proportion of 'quite good' or 'superior' knowledge then the observed differences
between the distributions are likely to disappear.

Responses to questions not reported in Table 3.3 relate to experience of computer
programming and use of computers in employment. Seventy percent of the business
students and more than 77 percent of the science students rated themselves as having
little or no knowledge of programming. Of the business students 28 (37.8 percent) used
computers in paid employment. Twenty-three of the 28 said they were users, while the
other five claimed to have higher levels of competence. Of the science students 23 (32.4
percent) said they used computers in paid employment and 20 of these 23 rated
themselves as users.

3.5.3 Computer Anxiety, Computer Playfulness and Locus of Control

The 13 questions on the anxiety measure and the 22 questions on the playfulness
measure required rating responses of Strongly Agree, Agree, Unsure, Disagree or
Strongly Disagree. The 29 questions on the locus of control measure required the
respondent to circle only one of two statements. Six of the statements were fillers and
coded as 0. The other questions were given score 1 if the respondent chose the answer
suggesting external influences. If the respondent chose the other possible answer
(suggesting self-direction and control) the score was 0. This is consistent with Rotter
(1966).

Following the example of Igbaria and Parasuraman (1989) averaging was used to obtain
single measures for anxiety and playfulness. For each respondent locus of control was
given a total score, obtained by adding his or her scores for each of the 29 questions.

The distributions and means of the anxiety, playfulness and locus of control measures
are summarized in Table 3.4.
<table>
<thead>
<tr>
<th></th>
<th>Business Students</th>
<th>Science Students</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$n = 74$</td>
<td>$n = 71$</td>
</tr>
<tr>
<td><strong>Computer Anxiety</strong>$^1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>First quartile</td>
<td>1.8</td>
<td>1.8</td>
</tr>
<tr>
<td>Second quartile</td>
<td>2.1</td>
<td>2.2</td>
</tr>
<tr>
<td>Third quartile</td>
<td>2.6</td>
<td>2.7</td>
</tr>
<tr>
<td>Mean</td>
<td>2.2</td>
<td>2.3</td>
</tr>
<tr>
<td><strong>Computer Playfulness</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>First quartile</td>
<td>3.3</td>
<td>3.0</td>
</tr>
<tr>
<td>Second quartile</td>
<td>3.5</td>
<td>3.4</td>
</tr>
<tr>
<td>Third quartile</td>
<td>3.7</td>
<td>3.7</td>
</tr>
<tr>
<td>Mean</td>
<td>3.5</td>
<td>3.3</td>
</tr>
<tr>
<td><strong>Locus of Control</strong>$^2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>First quartile</td>
<td>8.0</td>
<td>9.0</td>
</tr>
<tr>
<td>Second quartile</td>
<td>11.0</td>
<td>11.0</td>
</tr>
<tr>
<td>Third quartile</td>
<td>14.0</td>
<td>15.0</td>
</tr>
<tr>
<td>Mean</td>
<td>11.2</td>
<td>11.6</td>
</tr>
</tbody>
</table>

$^1$ Two science students did not respond to this question  
$^2$ Two business students and 8 science students did not respond to this question

**Table 3.4: Anxiety, Playfulness and Locus of Control Measures**

The distributions of anxiety scores are similar for the two groups. The locations of the quartiles are approximately the same and the distributions are slightly skewed to the right (as the means exceed the medians). The smallest possible value on this measure is one, indicating negligible anxiety about using computers. The maximum score is five, indicating the respondent is strongly anxious. That the medians and means are less than 2.5 suggests slightly less anxiety than would be found in a population where anxiety is normally distributed with mean at the mid-point of the interval 1 to 5.

Anderson (1994) found a mean anxiety score of 2.6 for a class of business students in 1992 doing an earlier version of the unit under investigation in this study. The decline of the mean to 2.2 in 1994 might be expected, as microcomputer usage became more widespread.
Some differences appear between the groups in relation to playfulness. First note that the value of the first quartile for the science students is smaller than for the business group. The distribution of playfulness for science students is left skewed; for the business group it is symmetric. The possible range of values is from one to five. The minimum score is 2.0 for science and 2.4 for business. It seems that in the science sample in particular the distribution of playfulness fails to match that for a population with symmetrically distributed playfulness.

The distributions of locus of control appear similar for the two samples. Locus of control has been measured so that the higher the score the more external is a respondent’s view of control of situations. The minimum possible score is 0 and the maximum is 23. For the two samples the mean scores lie between 11 and 12.

Rotter (1966) reports a number of studies of high school students, university entrants, prisoners and college applicants. The largest mean reported by Rotter for locus of control was 10.00 for male 18-year olds who were hopeful of going to college. Females in this group scored on average 9.00. The latter is the third highest mean in his table of 8 studies desegregated by gender. Rotter (1966, p.16) reasoned that ‘students in high school seeking to go to college are more internal than is an unselected high school population’.

The results for locus of control measures, gathered some thirty years later for this study, suggest that the students just beginning the two tertiary courses in business and science are close to Rotter’s values. At an interval of thirty years, after allowing for cultural, generational and educational differences, comparison with the results of Rotter is however highly speculative.

3.5.4 Experience, Anxiety, Playfulness and Locus of Control

In this section correlations between indicators of success in introductory computing are investigated. The measures discussed follow the work of Igbaria and Parasuraman (1989), Parasuraman and Igbaria (1989), Webster and Martocchio (1992) and Martocchio and Webster (1992).
Two indicators of experience were obtained from the coded responses. The first denoted COMPKNOW was obtained from the respondents' assessments of their overall computing knowledge (Question 6 in Section D of the survey). Where the respondent indicated passable or higher knowledge of computing, COMPKNOW was given a value of 1 and 0 otherwise. The second variable, COMPEXP, was obtained by summing the responses to questions relating to microcomputer hardware and experience with software (Questions 1, 2, 4 and 5 in Section D of the survey). The components of COMPKNOW and COMPEXP are detailed in Table 3.3.

<table>
<thead>
<tr>
<th></th>
<th>Computer Experience (COMPEXP)</th>
<th>Locus of Control</th>
<th>Computer Anxiety</th>
<th>Computer Playfulness</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Self-evaluated Knowledge (COMPKNOW)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Business Students</td>
<td>0.62 **</td>
<td>-0.07</td>
<td>-0.50 **</td>
<td>0.24 *</td>
</tr>
<tr>
<td>Science Students</td>
<td>0.59 **</td>
<td>0.10</td>
<td>-0.66 **</td>
<td>0.19</td>
</tr>
<tr>
<td><strong>Computer Experience (COMPEXP)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Business Students</td>
<td>-0.10</td>
<td>-0.42 **</td>
<td></td>
<td>0.31 **</td>
</tr>
<tr>
<td>Science Students</td>
<td>0.09</td>
<td>-0.53 **</td>
<td></td>
<td>0.19</td>
</tr>
<tr>
<td><strong>Locus of Control</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Business Students</td>
<td>0.14</td>
<td>-0.12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Science Students</td>
<td>0.14</td>
<td>-0.29 *</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Computer Anxiety</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Business Students</td>
<td>-0.36 **</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Science Students</td>
<td>-0.44 **</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(* is 5 percent level or better and ** is 1 percent or better)

Table 3.5: Non-Parametric Correlations

Table 3.5 sets out the results of a correlation analysis. The first entry in Table 3.5 shows the rank correlation between self-evaluated knowledge and the composite experience variable.
As expected the correlation between these variables is positive and significant at the one percent level for both groups. The magnitudes of the coefficients suggest moderate correlation.

Experience and knowledge produce similar correlations with the other indicators:

1. there is negligible association with locus of control;
2. there is evidence of negative correlations with computer anxiety;
3. for business students there is evidence of positive correlation with playfulness.

When there is evidence of significant correlation the magnitudes of the coefficients are less than 0.7.

Ray and Minch (1990, p. 485) and Anderson (1992) note that experience is widely found to be negatively correlated with anxiety. This is clearly the case for the two samples discussed here. Martocchio and Webster (1992) report a negative correlation between pre-training anxiety and experience, significant at the five percent level. They further report a positive correlation of 0.34 between experience and playfulness among full-time employees on microcomputer training courses. The employees were clerks and administrators with an average age of 41 and, on average, 7 years work experience. Despite the differences between the subjects of Martocchio and Webster (1992) and the business students included for this preliminary survey, both samples showed positive correlation between experience and playfulness.

Look now at the third row of Table 3.5. Locus of control shows low but insignificant correlation with computer anxiety. The positive signs of the correlations are consistent with the findings of Morrow et al. (1986) and Parasuraman and Igbaria (1990). However, the correlations of the Parasuraman and Igbaria study were significant at the 5 percent level. The subjects for their study were part-time MBA students with an average age of 31 who held down full-time jobs. The positive correlation indicates an association between increasing view of external control and anxiety. This contradicts the finding in the current study. The contradiction might be explained in terms of the immediacy of external influences (such as the demands of work) among more mature students.
The locus of control measure also displays negative correlation with computer playfulness. This suggests that students who see the external environment as controlling their outcomes are less inclined to exhibit playfulness. However, this result is significant only for the science students.

There is evidence in the final row of Table 3.5 of negative correlation between anxiety and playfulness. Martocchio and Webster (1992) also found significant negative correlation. They report that

... playfulness demonstrated higher incremental validity than traditional factors such as computer anxiety. That is, cognitive playfulness generally relates more strongly than computer anxiety or computer attitudes to learning, mood and satisfaction (p. 557).

3.6 Conclusions
For samples of science and business students doing an introductory information technology unit, few differences were found in terms of locus of control, computer anxiety and playfulness. A series of correlations significant at the five percent level or better were discovered among these three variables and indicators of computing knowledge and experience. The signs of the significant correlations with either knowledge or experience were the same. The magnitudes were similar. This is not surprising as knowledge and experience were positively and significantly correlated.

Consider an increase in either experience or knowledge. This is associated with reduced anxiety. In turn this is associated with increased playfulness. The net effect is that an increase in experience or knowledge is associated with playfulness rising. A significant correlation of this type is seen in Table 3.5 for the business students.

The inter-linkages between variables are represented in Figure 3.1 where a two-headed arrow is used as the connector between associated variables. The dashed arrow in the figure indicates an induced association between knowledge or experience and playfulness on the basis of the two correlations involving anxiety and playfulness that are labeled with negative signs.
For science students locus of control and playfulness are negatively correlated at the five percent level. It appears locus of control is unrelated to the other variables.

![Diagram](image)

**Figure 3.1: Relationships Among Survey Factors**

A focus group consisting of teachers of computing did not raise locus of control as important in student success. The views of the focus group are discussed in the next chapter. The finding of moderately strong correlations between experience, knowledge, anxiety and playfulness suggest that in regressions for programming performance (Chapter 7) only one of these variables should be included. This allows the construction of parsimonious models with little multicollinearity in the estimation of coefficients (Kennedy 1992). In the following analysis of students in programming units it might be reasoned that anxiety is reduced and productive playfulness is established in the introductory unit. This idea is confirmed in Chapter 5. Thus, either knowledge or experience might be used in the regressions.
CHAPTER 4
EXPLORING THE COLLECTIVE VIEW OF THE EXPERTS

4.1 Introduction
The first stage of the preliminary study, the survey described in the previous chapter, focussed on characteristics of business and science students who were undertaking an introductory unit of computing. The research described in this chapter comprised the second stage of the preliminary study. The aim was to identify factors that determine student performance in computer programming and then investigate the relationships between them.

To identify important factors a focus group consisting of teachers and lecturers of computer programming (the ‘experts’) was assembled. The main factors they agreed on were put to thirteen students taking an introductory programming course (Chapter 5). This was done for two reasons. First to decide whether there was agreement between students and their teachers. Second, to decide if the list of the experts might be modified for the later research on linkages (see Chapter 7). The existing literature usually identifies a small number of factors and investigates a smaller number of linkages between them. For example, Kurtz (1980), Nowaczyk (1984), Little (1984), Thronson (1984), Kagan (1988), Clarke and Chambers (1989), Chen and Vecchio (1992).

The people chosen as experts were teachers and lecturers in computing in Melbourne tertiary educational institutions. These experts were teachers of first-year computer programming and some had previously taught the students surveyed (see Chapter 3). Qualitative data was collected on what experts saw as personal constructs (see Kelly 1955, 1966) influencing performance in computing. In a wide ranging brainstorming session a very large number of constructs were identified. Among these the experts identified significant linkages and common themes. Cognitive mapping software was then applied to structure and organize the qualitative data in order to ‘... understand them in terms of similarities and differences’ (Anderson 1994, Ackermann 1994, Cropper et al. 1993).
4.2 Choosing a Group Knowledge Acquisition Technique

There are a number of ways of gathering information from a group. Tyson (1994) mentions 23 distinct approaches. Knowledge acquisition techniques vary from highly structured to those that are less structured and unstructured (see Figure 4.1).

<table>
<thead>
<tr>
<th>Structured</th>
<th>Unstructured</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structured workshops</td>
<td>Reference groups with/without SW</td>
</tr>
<tr>
<td>GroupWare</td>
<td>Teleconferencing</td>
</tr>
<tr>
<td>Inter-organizational networks</td>
<td>Group writing</td>
</tr>
<tr>
<td>Hypermedia</td>
<td>Teleconferencing</td>
</tr>
<tr>
<td>simulation models</td>
<td>Group writing</td>
</tr>
<tr>
<td></td>
<td>Teleconferencing</td>
</tr>
</tbody>
</table>

Adapted from Grabowski et al. 1992, p. 413.

**Figure 4.1: Group Knowledge Acquisition Techniques**

The focus group method was preferred. One reason was time. The participants were volunteers with busy schedules. Another consideration was the desire to explore as many factors as possible in the students' environment. Grabowski et al. (1992 p. 421) remarked:

After the problem is defined, utilizing a focus group of domain experts might prove helpful in orienting a knowledge engineer to the problem domain, better preparing him or her to subsequently use either observation or interviewing techniques to elicit knowledge. A combination of book knowledge and utilizing focus groups might also expose a range of thoughts, ideas or guidelines that might help in assessing appropriate knowledge representation schema.

This conforms to the strategy of using insights from the focus group as guidelines for the main study reported in Chapters 6 and 7.

Drawbacks in other studies involving focus groups concerned data management and the interpretation of findings. One method of dealing with this was to give an ordered account, supported by material quoted from transcripts (Lunt and Livingstone 1996; Grabowski et al. 1992). Another method was to systematically code transcripts using content analysis. Both of these methods were time-consuming and were subject to questions of reliability in interpretation (Lunt and Livingstone 1996).
The introduction of new modeling techniques, supported by software and the development of well-developed rules and practices for the modeling of a person's thinking (Cropper et al. 1993), provide an efficient and effective method for collecting and organizing data. To analyze the qualitative data on constructs, the Graphics COPE\(^4\) (COgnitive Perception Evaluator) package was used. Focus groups, cognitive mapping and the COPE software are discussed further in the following sections.

### 4.3 Focus Groups

Focus groups are commonly set up in the social sciences with the purpose of collecting qualitative data in the form of a group interview. The method involves organizing a group of individuals to come together to discuss a particular issue or to solve a particular problem. Generally the group members have not met together in the past, nor are they expected to reconvene in the future. One advantage of the focus group method is that it provides an opportunity to have direct contact with interviewees in a flexible, unstructured format. Other advantages according to Grabowski et al. (1992) and Lunt and Livingstone (1996) of the focus groups compared with one-to-one interviews are:

- they are quicker to implement
- more solutions are provided
- better solutions are possible because of broad based expertise and information.

During a focus group session, participants reveal and share insights, are likely to give more truthful responses and "... can expose underlying heuristics, knowledge, beliefs and information not previously elicited" (Grabowski et al. 1992, p. 421).

### 4.4 Cognitive Mapping

Cognitive mapping is a method for representing the ways in which individuals think about a particular situation or issue. Kelly (1955) held that each individual constructs a model of reality that enables him or her to decide on responses in relation to that model. Such models are constantly modified to provide improved predictions.

\(^4\) Since the research was completed Graphics COPE has evolved into software named 'Decision Explorer'.
Thus for Kelly the question in exploring, revising and replacing in the light of predictive failure which is symptomatic of scientific theorizing, is precisely what a person does in his attempts to anticipate events (Pope and Shaw 1981, p. 26). This approach underlies Personal Construct Psychology.

In 1969 Kelly suggested that a technology was required through which to express human intentions. A number of such tools evolved. For example, in education Pope and Shaw (1981) refer to the programs PEGASUS and SOCIOTRIDS. At least one of these involved interactive elicitation of information from subjects.

Following Kelly proponents of cognitive mapping software recognize that individuals see the world differently and each person gives different meanings, interpretations and significance to situations and events. This explains why it is difficult for groups to agree about the nature of issues and how to respond to them in management science. Cropper et al. (1993) discussed a decision support system that produces cognitive maps.

According to Cossette and Audet (1992 p. 327) a cognitive map is an image capturing ‘a set of discursive representations made by a subject with regards to an object in the context of a particular interaction’ (their italics). An investigator builds the image from the utterances of subjects. Cropper et al. (1993 p. 176) note an advantage:

Cognitive mapping can be used to capture verbal accounts and analyses of issues as they are presented in the natural language of those charged with deciding what to do about a problem or issue and to structure and represent them in a manner that is helpful to that decision process.

The maps consist of several elements. People supplying information express concepts in words or distinct phrases. These concepts are then linked to form a hierarchy, where the status of phrases with respect to each other is decided. A pair of phrases may be united in a single concept where one provides a meaningful contrast to the other. When the meaning of a concept is not clear, requesting information regarding the opposite pole may provide clarification in the context under investigation (Ackermann et al. 1993).
In the next section the COPE tool for obtaining cognitive maps is described. Then the approach of Ackermann et al. (1993) to eliciting qualitative information is described. This approach and COPE produced cognitive maps representing the experts’ views on student success in computer programming.

4.5 Graphics COPE Software

This analytical software, first developed by academics at the universities of Bath and Strathclyde, is a tool for managing qualitative information, resulting from complex issues or situations. It can be used on site in direct consultation where information is entered directly into the computer or information can be manually recorded and then subsequently transcribed onto the computer. The human brain can only cope with a limited number of concepts and connections between them. The number of concepts that a person can follow at any time is said to be seven plus or minus two (Miller 1956). The COPE software follows all elements and allows the user to see the information in condensed or detailed form. The complete model is displayed as a collection of views or maps. Those involved in the creation of the model are able to identify their contributions.

According to Banxia Software (n.d. p.2) who now market Graphics COPE, the software can help:

- Pull ideas together cohesively
- Discover the real issues
- Manage complexity without losing richness of detail
- Build feasible, practical and acceptable solutions
- Effectively present reasoning
- Maintain the creativity and focus at group meetings.

4.6 Eliciting Information from Experts

A group of educators in the field of computer programming was identified. The educators had extensive experience of teaching first-year programming students. They were academics from a variety of tertiary institutions in the Melbourne metropolitan area and are referred to as the ‘experts’ in the following sections. They were familiar with local institutional frames of reference, teachers’ constructs and the diversity of approaches and attitudes of students.
Thus, in the tradition of Kelly allowance is ‘...made for diversity of viewpoints and constructive alternatives among educators’ (Pope and Shaw 1981, p. 224). Among the experts were the coordinators of first-year computer programming from each of the business computing and computer science streams at Deakin University. Ten academics were invited to participate but two declined. No payment was made. Others present included the facilitator for the group and a video camera technician.  

4.6.1 The Focus Group Session

One meeting of the experts was organized. It took place at Deakin University in the early evening and refreshments were offered. The facilitator ensured that the procedures recommended by Ackermann (1994) were followed in eliciting information. The question posed was -

What are the factors affecting success in computer programming?

Programming success was defined as the achievement of a satisfactory grade - a clear pass - in elementary programming. The members of the focus group wrote their answers on yellow ellipses cut from paper. These were pasted on a whiteboard in no particular order. The participants were requested to keep the wording on the ellipses to less than ten words. Almost 50 elements of programming success were identified in the first 15 minutes and 73 elements were identified overall.

The facilitator suggested that the group assemble the answers into common groups and establish linkages, if any, between groups and among concepts within those groups. During the session the experts engaged in a lively discussion on all aspects of computer programming and they actively articulated and elaborated their views. A common understanding was reached. Discussion sometimes concerned what was meant by the words on a particular ellipse. In these cases the particular word or phrase had to be defined or clarified by the person who created it. For example, maths processes was grouped with maths ability but the ‘owner’ of the first phrase explained that this referred to knowledge about mathematics techniques rather than mathematics ability. Thus a distinction was drawn between students' knowledge and ability.

---

5 The Deakin staff taught the students surveyed for the research reported in Chapters 3, 5 and 7.
6 This was the researcher's principal supervisor at the time. The author observed the process.
It was interesting to note also that one of the responses - interest and motivation - was split into its two parts after a discussion among the participants. Motivation was considered subsequently to be much higher in the mapping hierarchy compared with interest. There was also a long discussion about whether the students' age was a factor in computer programming success. It was thought that age could be a proxy for a host of other reasons for studying. The group did not elaborate on those influences. The group session lasted approximately two hours.

4.6.2 Cognitive Maps from Graphics COPE

The data input to Graphics COPE was transcribed from the videotape that included clear depictions of the final groupings of the ellipses. Each ellipse was given an identification number. Both the phrases on the ellipses and the experts’ final grouping of them were input to COPE. The software generated three cognitive maps shown as Maps 4.1 to 4.3. Working independently, the experts were asked to review the maps and revise them. Only minor changes were suggested.

The conventions of Cropper et al. (1993) were followed in drawing the maps:

- an ellipsis is used to represent the phrase ‘rather than’, thus contrasting concepts
- the arrows mean ‘may lead to’ or ‘may imply’
- a link may be negative. (An example in Map 4.1 is where the negative sign from statistically more students means more success is a link to the contrasting pole college/teacher motivation and not personal motivation).

The three maps exhibit four superordinate constructs. These are constructs that appear at the tops of maps, arrows point towards them and none emanate from them. These superordinate constructs are personal motivation rather than college/teacher motivation in Map 4.1. Similarly basic academic ability in Map 4.2 and problem-solving skill and goal oriented in Map 4.3. The experts regard these superordinate themes as goals, that is, the good attributes that students should have to succeed.

Among other things the superordinate construct in Map 4.1 is directly linked to reason for studying. In the Ackermann et al. (1993 p. 5) tutorial on cognitive mapping, constructs such as this, which are immediately beneath the goals, are referred to as ‘key’ or ‘strategic’ directions.
Map 4.1
23 basic academic ability

71 understanding the use of 'arrays'
16 the chosen language
12 capacity for language acquisition

60 entry scores into courses

72 learning style

1 maths aptitude

45 perceiving 'errors' as the norm

33 has a breadth of discipline interests (not narrow minded)

82 ability to synthesize

50 able to identify underlying assumptions

53 there is a microscopic view and a bigger view

48 ability to see the whole from the parts

47 ability to handle and understand repetition

9 willingness to explore beyond the obvious

41 capable of proposing options and solutions

37 competent at building mental models

64 native intelligence

7 ability to think/work abstraction

36 capable of visualizing concepts such as data structures

49 ability to decompose a problem into sub-problems

62 ability to re-use existing programs

68 program construct recognition (familiarity)
These key constructs,

... have some or all of the following characteristics: long-term implications, high cost, irreversible, need a portfolio of actions to make them happen, may require a change in culture (ibid.).

Beneath key issues the constructs are referred to as '... portfolios of potential “options” which explain (and thus suggest) potential solutions to the key issues to which they are linked' (ibid.). For the key direction reason for studying in Map 4.1, some options are relevance to larger goals, interest/exposure to computers and knowledge of computers. Some areas for further exploration are deduced from the cognitive maps.

4.7 Views of the Experts, Structuring Student Interviews

Constructs from the cognitive maps were analyzed to establish direction for the subsequent interviews and surveys of programming students. In Map 4.1 the goal of personal motivation rather than university/teacher motivation is associated with two key directions: reason for studying and statistically more students means more success. The latter was seen as leading to an emphasis on university or teacher motivation. From the video of the meeting it is clear that this construct refers to the pool of students applying to enroll. The experts’ consensus was that the larger the pool of applicants the greater the likelihood of filling a fixed number of places with students likely to succeed.

To influence this strategic direction would require a change in the local culture that induces school leavers to seek admission to the more established universities before any others. The option explaining this key issue was given as resources. An influence on resources that emerged from the meeting was teaching style. From the video, resources however involved connotations of high cost.

Reason for studying was seen by the experts as a means influencing personal motivation. Strategically, to influence this would require wide-ranging societal change involving a range of initiatives. One change of the past 10 to 15 years is the promotion by government of a highly skilled computing workforce. On the other hand, as is made clear in the next chapter that reports interviews with students, while programming knowledge is thought to be important, the actual skills and how they are acquired are often only naively understood.
Options or influences on reason for studying were given as age, returning students, computers [being] the ‘in thing’ [in] the future, parental pressure and relevance to larger goals. The last of these is influenced by interest according to the experts. Underlying interest is a range of constructs impacting on attitude to and knowledge of computers as well as knowledge of basic mathematics processes. As mediated by the Graphics COPE software, it appears the experts agreed that knowledge of computers and mathematics influences motivation along an indirect path. The many constructs associated with this pathway is in contrast to those subordinate to statistically more students means more success. After a relatively short discussion the experts agreed that this strategic direction was a digression from the analysis of the question. The reason was that the experts were trying to answer the question given their knowledge of students currently attracted to programming courses.

In Map 4.2 strategic directions for the goal are:
- learning style
- entry scores into courses
- ability to synthesize
- capacity for language acquisition.

The key with the largest number of subordinates is clearly the ability to synthesize. The options are a composite of technical skills and intellectual capacities according to the experts. The ability to synthesize is seen as one of the higher-order intellectual capacities. In the language of Ackermann et al. (1993) this may be a strategic direction that is ‘irreversible’ or involves extremely “high cost”.

The strategic direction entry scores into courses has a relationship with the statistically more students means more success in Map 4.1. As noted above the latter is associated with the size of the school-leaver pool of applicants. Size of pool influences the mean and the standard deviation of entry scores for those who are admitted to the course. The effect is that an individual entry score is likely to be higher.

In Map 4.2 the key directions entry score and learning style (discussed in section 4.8) form a part of the quantitative analysis in later chapters.
There are two superordinate constructs in Map 4.3. Consider first goal oriented that was defined as the ability to see results or reach the end of a project. Strategic directions here are recorded as competitive and pride in finished product. The former refers to personal acceptance of challenge such as to perform better, or be faster, than other students do. Some students seemed to see competitiveness as a personal end in itself.

The goal problem solving skill is taken up in quantitative research in the next chapter. There is an interesting linkage between the goals in Map 4.3. Note that the implication running from methodical to clear thinking provides a linkage at a subordinate level between the goals. In the video there is evidence of the group ambiguously understanding what methodical meant.

In the next section a number of constructs are selected for assessment in interviews, a survey and in brief tests. These were conducted with business students.

4.8 Implications for Further Research

The cognitive maps provide a view of the consensus reached among eight experts about factors affecting student success in elementary programming. The ‘experts’ were teachers and lecturers of computer programming operating in the Melbourne tertiary education sector. Some features of the evolving consensus and the extent of agreement are not captured in the maps. One aspect of the emergence of this is clear from the videos.

Near the conclusion of the meeting of experts, one suggested a summary of findings. With one modification it was accepted that three things that affect success in elementary programming:

- students’ innate abilities, previous experience and motivation at commencement
- what they are taught
- teaching method, style and interaction with students’ learning style

With this summary as a guide, constructs from the cognitive mapping session were selected for analysis in the ensuing parts of the research. In the next and final stage of the preliminary study the following constructs were explored:
From Map 4.1:
The goal of personal motivation, the key (strategic) direction of reason for studying, the option relevance to larger goals and a number of its subordinates could be investigated. (See Chapter 5 and Appendix D.) It was not possible within the scope of any of this research to investigate the other strategic direction in Map 4.1 or any of its subordinates. Recall the view of the experts that this strategic direction was irrelevant to the question.

From Map 4.2:
A number of the subordinate constructs underlying the strategic goal are explored in the next chapter. For example, students were asked whether aptitude in mathematics and willingness to explore were important in their mastery of programming.

From Map 4.3:
The goal of problem-solving skill was tested with short-answer questions for all students in a programming class. These tests also examined the constructs of methodical and clear thinking. (Although in view of the uncertainty of the experts' definition of methodical no attempt was made to seek a quantitative correlation between them.) The importance of being goal-oriented was examined in interviews along with the key direction pride in finished product.

The concern over teaching and learning styles requires a few words of explanation. Because the strategic direction statistically more students means more success was dismissed in the context of students currently taking the course, little was said during the cognitive mapping session on its subordinate teaching style. Yet this emerged near the end of the meeting as a factor seen collectively as important. Rather more discussion occurred on learning style, as it was a strategic direction for the goal basic academic ability.

The example was cited of students from humanities and arts who take computing as a subsidiary or optional study. Most of them concentrated on non-science disciplines in their final years of high school. At the institutions of the experts the joint study of computer programming and humanities was possible. Experts reported 'learning shock' for such students because the teaching style is quite different in a hard science.
Together with the culture shock of university life in first year (when most students have their initial brush with programming) significant obstacles to effective learning might present for some students. The issue of learning style is taken up in Chapters 6 and 7. However, it was not possible to investigate its importance for students with interests predominantly in the humanities. Sample sizes were unfortunately too small.
CHAPTER 5
INFORMATION FROM COMPUTER-PROGRAMMING STUDENTS

5.1 Introduction

This chapter describes the third and final stage of the preliminary study of the factors affecting computing performance. The first stage was a survey in which business and science students taking an introductory unit of computing were profiled (see Chapter 3). The second stage was the cognitive mapping session in which the views of a focus group of experts were analyzed (see Chapter 4). In this final stage a number of students taking a unit in programming were interviewed; they completed a survey and as part of the final examination did a problem-solving test.

5.2 Background

Thirteen students were asked to participate in this stage of the preliminary study. The students were a sub-sample of first-year computer-programming students enrolled in a business-computing course. The unit coordinator selected them as representative of the first-year group. A range of perceived abilities in computer programming, with a gender balance, were criteria for selection. Prior to taking programming the students completed the introductory computing unit, the Business Information Systems (the unit discussed in Chapter 3) as well as commerce units.

A description of the programming unit Software Development is:

The aim of this unit is to develop in students a professional approach to software development and a sound understanding of structured programming concepts. Students will gain experience in designing and implementing software and in particular will develop skills in algorithm and program design, in writing programs for a variety of applications and become conversant with a commercial programming language. This unit is presented in two concurrent streams of programming theory and the study of a programming language (Deakin University Handbook 1994 p. 413).

Software Development was delivered in the form of one two-hour lecture and one two-hour tutorial session. The assessment was by examination (60 percent), assignment (30 percent) and tutorial exercises (10 percent).
5.3 Research Method

The student participants for this research were required to:

1. attend a semi-structured interview
2. complete a questionnaire
3. answer written questions designed to test problem-solving ability

The questionnaires were completed during the interviews. This occurred at Week 12 of the semester, prior to the end of semester examinations. The final examination marks in Software Development, the computer-programming unit, were collated with survey responses. The problem-solving question was attempted as part of the final examination.

5.3.1 Format and Content of the Interviews

The purpose of the semi-structured interviews was to elicit information from students regarding their computing background, their motivation in pursuing programming studies, how they learnt programming and what they perceived as important factors in mastering computer programming.

Questions asked of all participants were about:

- previous experience, knowledge and study in computing
- why they chose a major in programming
- why they enrolled in this particular university
- their prior expectations of programming and how these changed
- the value of lectures, tutorials, assignments and textbooks
- how they debugged programs
- what their strengths and weaknesses were in relation to the programming task
- whether or not they planned to continue with a computer-programming major
- where they saw themselves in five years' and the type of work they were doing

A schedule of the questions used during the interview is in Appendix C. At the conclusion of the interview, the participants were given a survey form to complete. The survey was similar to the survey described in Chapter 3.
5.3.2 Survey Used during the Interview

The survey sought to measure:

- computer anxiety
- computer playfulness
- locus of control
- ranking of factors for success in programming

The first three indicators are measured using the instruments discussed in Chapter 3 in sub-section 3.4.1 to sub-section 3.4.3. The questions relating to gender, age, secondary school performance and computer experience from the original survey were not included as these questions were part of the questions at interview.

The fourth survey section consisted of a list of factors that required the participants to rate the importance of each in mastering programming. These factors were collated from the focus group cognitive mapping session described in Chapter 4.

The fourth part of the survey asked students to rate the importance of the 32 factors listed in Table 5.1. The factors were drawn mainly from those described by the experts as being requirements for success in elementary programming (see Chapter 4). In the table the source of the factor is shown. For example, Factor 1 in Table 5.1 is prior experience with computers. In the columns with the heading ‘Source’, the choice of this factor is seen to be motivated by element 6 of Map 4.1.

It was important to ensure that the students being surveyed understood what each factor meant. For example, element 49 on Map 2 reads ability to decompose a problem into sub-problems which was translated for the students into ability to break down a problem into smaller tasks (Factor 11 in the table).

The students were asked to rate each of the factors and ratings were categorized according to a 5 point Likert scale:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Disagree = 2.</td>
<td>Strongly Disagree = 1.</td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Factor</td>
<td>Source</td>
</tr>
<tr>
<td>-----</td>
<td>---------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>1</td>
<td>Prior experience with computers</td>
<td>Map 1</td>
</tr>
<tr>
<td>2</td>
<td>Enthusiasm about computers</td>
<td>Map 1</td>
</tr>
<tr>
<td>3</td>
<td>Ability to play computer games</td>
<td>Other</td>
</tr>
<tr>
<td>4</td>
<td>Previous experience with application programs</td>
<td>Map 1</td>
</tr>
<tr>
<td>5</td>
<td>Maths Ability</td>
<td>Map 2</td>
</tr>
<tr>
<td>6</td>
<td>Motivation to succeed</td>
<td>Map 1</td>
</tr>
<tr>
<td>7</td>
<td>Lectures</td>
<td>Map 1</td>
</tr>
<tr>
<td>8</td>
<td>Tutorials</td>
<td>Map 1</td>
</tr>
<tr>
<td>9</td>
<td>Discussion with peers</td>
<td>Other</td>
</tr>
<tr>
<td>10</td>
<td>Textbooks</td>
<td>Map 1</td>
</tr>
<tr>
<td>11</td>
<td>Ability to break down a problem into smaller tasks</td>
<td>Map 2</td>
</tr>
<tr>
<td>12</td>
<td>Intelligence</td>
<td>Map 2</td>
</tr>
<tr>
<td>13</td>
<td>Subjects chosen at VCE level</td>
<td>Map 2</td>
</tr>
<tr>
<td>14</td>
<td>Ability to construct algorithms</td>
<td>Map 2</td>
</tr>
<tr>
<td>15</td>
<td>Computer anxiety</td>
<td>Map 1</td>
</tr>
<tr>
<td>16</td>
<td>Method of teaching computer programming</td>
<td>Map 1</td>
</tr>
<tr>
<td>17</td>
<td>Thoroughness</td>
<td>Map 3</td>
</tr>
<tr>
<td>18</td>
<td>Clear-thinking</td>
<td>Map 3</td>
</tr>
<tr>
<td>19</td>
<td>Problem-solving ability</td>
<td>Map 3</td>
</tr>
<tr>
<td>20</td>
<td>Ability to follow directions</td>
<td>Map 3</td>
</tr>
<tr>
<td>21</td>
<td>Precision</td>
<td>Map 3</td>
</tr>
<tr>
<td>22</td>
<td>Ability to analyze problems</td>
<td>Map 3</td>
</tr>
<tr>
<td>23</td>
<td>Methodical approach</td>
<td>Map 3</td>
</tr>
<tr>
<td>24</td>
<td>Pride in finished product</td>
<td>Map 3</td>
</tr>
<tr>
<td>25</td>
<td>Perseverance</td>
<td>Map 3</td>
</tr>
<tr>
<td>26</td>
<td>Ability to modify or re-use existing programs</td>
<td>Map 2</td>
</tr>
<tr>
<td>27</td>
<td>Reasons for undertaking the subject</td>
<td>Map 1</td>
</tr>
<tr>
<td>28</td>
<td>Ability to explore and experiment</td>
<td>Map 2</td>
</tr>
<tr>
<td>29</td>
<td>Ability to read and understand programs</td>
<td>Map 2</td>
</tr>
<tr>
<td>30</td>
<td>Understanding the use of &quot;arrays&quot;</td>
<td>Map 2</td>
</tr>
<tr>
<td>31</td>
<td>Ability to understand &quot;repetition&quot;</td>
<td>Map 2</td>
</tr>
<tr>
<td>32</td>
<td>Creativity</td>
<td>Other</td>
</tr>
</tbody>
</table>

Table 5.1: Sources of Survey Factors
Three factors were not selected from the cognitive maps. They are ability to play computer games, discussion with peers and creativity. Playing computer games was mentioned as one of the informal ways of gaining computer experience (Clarke and Chambers 1989). The discussion with peers factor was included after consultation with the computer-programming coordinator. Weinberg (1972) suggested the ability to create new programming ideas was important when deciding the overall design of a program and programmers lacking in creativity would be handicapped. It should be noted that creativity was not qualified in any way. If the words in program design had been included as part of the statement the responses may well have been different.

5.3.3 Measuring Aptitude for Computer Programming

Next the third set of data on the programming students is described. Recall from Map 4.2 that the experts had identified the ability to synthesize as an important theme and ability to identify underlying assumptions and pattern matching ability as possible factors in computer-programming success. The experts had also mentioned ability to follow directions and follow instructions within the ability to work methodically theme in Map 4.3. None of these factors were included in the survey. To examine them three short problem-solving questions were included on the end-of-semester examination paper. The questions were set in consultation with the computer-programming coordinator.

Two questions were adapted from the NCR Aptitude Test E51 (circa 1973), a test for ‘electronic data-processing programmers’. The problems were stated in the form of flowcharts where answers were obtained by following step-by-step instructions. The third question required some simple arithmetic knowledge and the ability to detect order in number and letter strings. There were ten parts to this question and they were adapted from CALIP: Computer Aptitude, Literacy, and Interest Profile (Poplin et al. 1984). CALIP is an instrument that identifies a variety of aptitudes appropriate to computer work. In the language of CALIP detecting order in a series of symbols is called ‘series aptitude’. A score for all three questions was compiled for all students undertaking the end-of-semester examination, including those who had been interviewed.
5.4 Results of the Study
Here the outcomes of each part of the study are presented.

5.4.1 Performance in Computer Programming
The final grades of the interviewed students are shown in Table 5.2 along with the distribution of results for all students taking the computer-programming unit. Two participants in the interviews did not sit the examination.

<table>
<thead>
<tr>
<th>Grade</th>
<th>Interviewed Students</th>
<th>All Students</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number</td>
<td>Percent</td>
</tr>
<tr>
<td>High Distinction</td>
<td>1</td>
<td>7.7</td>
</tr>
<tr>
<td>Distinction</td>
<td>2</td>
<td>15.4</td>
</tr>
<tr>
<td>Credit</td>
<td>3</td>
<td>23.1</td>
</tr>
<tr>
<td>Pass</td>
<td>3</td>
<td>23.1</td>
</tr>
<tr>
<td>Fail</td>
<td>2</td>
<td>15.4</td>
</tr>
<tr>
<td>Did not sit</td>
<td>2</td>
<td>15.4</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>13</strong></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.2: Performance in Programming

The sample mean mark of 60.4 is close to the mean of 61.3 for the whole class. The proportion of students in the sample who passed or did better was 69.5 percent. For the entire class the corresponding proportion was approximately 9 percent lower. However there is little difference between the proportions receiving distinctions or higher distinctions in the sample (23.1 percent) and the population (24.2 percent). Broadly speaking the sample is reasonably representative of the class.

5.4.2 Interviews
Each interview took about one hour. During this time each interviewee completed the questionnaire described earlier in this chapter. The interview sessions were recorded. The surveyed students were all less than 20 years old, all studying full time and five were female.
Previous Experience

Table 5.3 shows students’ performances in the unit, cross-tabulated with their previous computing experience. Previous experience was discussed during the interviews. The first row shows grades obtained by those students who reported in the interview that they had little experience with computers.

There is slight difference between those with little experience and those with experience of applications (such as use of spreadsheets). However, previous programming experience appears to shift the distribution of grades upwards.

<table>
<thead>
<tr>
<th>Previous Experience</th>
<th>HD</th>
<th>D</th>
<th>C</th>
<th>P</th>
<th>N</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Little</td>
<td>_</td>
<td>_</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>Applications</td>
<td>_</td>
<td>1</td>
<td>_</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Programming</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>_</td>
<td>_</td>
<td>4</td>
</tr>
</tbody>
</table>

1 Decreases the numbers in the sample who obtained:
**HD**: High Distinction; **D**: Distinction; **C**: Credit; **P**: Pass; **N**: Fail or Did Not Sit.

2 Little experience is limited use of computers and no programming experience; Applications is school experience with applications but no programming; and programming is did programming at school.

Table 5.3: Previous Experience and Performance

One might interpret this finding as arising because people possessing the innate characteristics which ‘make a natural programmer’ see programming as an attractive area of study prior to university. This hypothesis was not investigated. However, an alternative possibility is that those who did programming at school have become familiar with some basic skills, understand the working environment and know what to expect. In the interviews information was gathered on what expectations they had about programming and whether these expectations had been realized.

Expectations of Computer Programming

Most of the students interviewed spoke about programming, as being very challenging and much harder than they thought it would be.
Only those who had done programming before claimed some idea of what to expect. Even so for them the pace was faster and the computer language different.

Two students who did not sit for the examination spoke about feeling completely out of their depth. They had trouble understanding what it was all about from the first day. Almost half of the interviewed students were surprised at the heavy emphasis on practical work. Three students remarked on how programming appeared to be easy for some students yet other students really had to struggle with it.

_Students' strengths and weaknesses_

Some students could have developed basic programming skills during their school days. Also, some might be familiar with the working environment and with the working practices conducive to success in programming. Some evidence on these issues was gathered in the interviews. Students were asked a number of questions to discover what they saw as their strengths and weaknesses in mastering programming.

The students described strengths in mastering programming as:

- the ability to see things through
- problem-solving ability
- the ability of breaking down problems into small understandable tasks
- thoroughness
- persistence and perseverance
- clear thinking
- calmness (not getting stressed)
- the ability to follow instructions or a prescribed method
- a willingness to experiment
- having a logical mind
- using a methodical/systematic approach
- patience
- having the motivation to succeed

Weaknesses in mastering computer programming were described as:

- not putting in effort and giving up
- a lack of discipline in debugging
- getting frustrated when things don't work properly
• jumping from one idea to another without thinking things through
• lack of creativity in trying to solve a problem a different way

Among the strengths and weaknesses are many references to attributes broadly relating
to work practices such as thoroughness, persistence and perseverance or 'not putting in
effort and giving up'. They highlighted a number of mental and organizational
characteristics such as clear thinking, having a logical mind or not 'jumping from one
idea to another without thinking things through'. Certain basic skills were mentioned
such as problem-solving skills and ability to follow instructions or a prescribed method.

Job Aspirations
This information was sought as a proxy for motivation. Of the thirteen students only
eight thought that they would continue with a computing major. Of these eight, only
five expected to make a career as information technology professionals. One of the two
who failed expected to remain with computing as a career choice. Those not interested
in information technology careers expected to be salaried workers or self-employed in
accounting and managerial practices.

5.4.3 Survey Results

As noted above the students each completed a survey, including assessments of
computer anxiety, playfulness and locus of control. A comparison of the outcomes for
these measures with the values for the 74 Business Information Systems (BIS) students
reported in Chapter 3 is given in Table 5.4. The results for BIS were gathered in
semester one and will have included results for the 13 students taking computer
programming (providing these students returned their surveys). This comes about
because BIS is a prerequisite for the programming unit and was taken by all
programming students in the previous semester.

Within the BIS group many students did not take any further information technology
units. The interviewees were slightly less anxious and the inter-quartile range on this
measure was shorter than for the BIS students. The interviewees were more playful.
Most striking is the difference in the locus of control measure for the two groups. In
particular the interviewees considered that outcomes for them were more influenced by
external influences than factors under their own control. This might be explained by the
timing of the interview and survey in Week 12 of the semester.
<table>
<thead>
<tr>
<th></th>
<th>Interviewees</th>
<th></th>
<th>Business Information Systems Students (Previous Semester)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>n = 13</td>
<td></td>
<td>n = 74</td>
</tr>
<tr>
<td><strong>Computer Anxiety</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>First quartile</td>
<td>1.9</td>
<td></td>
<td>1.8</td>
</tr>
<tr>
<td>Second quartile</td>
<td>2.1</td>
<td></td>
<td>2.1</td>
</tr>
<tr>
<td>Third quartile</td>
<td>2.4</td>
<td></td>
<td>2.6</td>
</tr>
<tr>
<td>Mean</td>
<td>2.1</td>
<td></td>
<td>2.2</td>
</tr>
<tr>
<td><strong>Computer Playfulness</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>First quartile</td>
<td>3.5</td>
<td></td>
<td>3.3</td>
</tr>
<tr>
<td>Second quartile</td>
<td>3.6</td>
<td></td>
<td>3.5</td>
</tr>
<tr>
<td>Third quartile</td>
<td>3.9</td>
<td></td>
<td>3.7</td>
</tr>
<tr>
<td>Mean</td>
<td>3.7</td>
<td></td>
<td>3.5</td>
</tr>
<tr>
<td><strong>Locus of control</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>First quartile</td>
<td>11.3</td>
<td></td>
<td>8.0</td>
</tr>
<tr>
<td>Second quartile</td>
<td>14.5</td>
<td></td>
<td>11.0</td>
</tr>
<tr>
<td>Third quartile</td>
<td>16.5</td>
<td></td>
<td>14.0</td>
</tr>
<tr>
<td>Mean</td>
<td>13.8</td>
<td></td>
<td>11.2</td>
</tr>
</tbody>
</table>

1. An interviewed student and a BIS student did not respond to this question.

**Table 5.4: Anxiety, Playfulness and Locus of Control Measures**

At the time of the survey students had imminent deadlines for assignments and there were only two weeks until the examination. Alternatively, the results for the BIS students had been gathered approximately five months earlier when the environment and student attitudes might have differed significantly. For example, the BIS survey was conducted by mail. Therefore students probably completed the questions in the comforting environment of home. On the other hand the 13 interviewees filled in the survey following their interview in a university office. Further, during the interview many of their problems with programming had been extensively discussed. The correlations of anxiety, playfulness and locus of control are shown in Table 5.5.
<table>
<thead>
<tr>
<th>Locus of Control</th>
<th>Anxiety</th>
<th>Playfulness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Examination Mark</td>
<td>-0.17</td>
<td>-0.45</td>
</tr>
<tr>
<td>Locus of control</td>
<td>0.44</td>
<td>-0.10</td>
</tr>
<tr>
<td>Anxiety</td>
<td></td>
<td>-0.49</td>
</tr>
</tbody>
</table>

**Table 5.5: Correlations of Performance**

Despite the high level of external control noted above, there is little correlation (and that is negative) between examination and locus of control. Similarly for these programming students there is a slight (but insignificant) tendency for decreasing anxiety to be correlated with greater examination mark. In the second row of the table there is positive correlation between locus of control and anxiety. However, this correlation like all of those displayed in Table 5.5 failed to be significantly different to zero at the five percent level.

Recall that the sample size is 13. The final column of the table shows negative correlation between playfulness and the other three indicators. The first of these is consistent with declining examination result as tendency to play increases. The correlation with locus of control has small magnitude. The sign suggests that increasing playfulness is associated with internal control. That is, there is a tendency for the more playful to be those who take responsibility for their own actions. Finally the correlation between anxiety and playfulness is −0.49. This seems intuitively plausible in that increasing playfulness is associated with reduced anxiety. The results in the final two rows of this table reflect the findings for both BIS students and science students taking introductory technology units in the first semester.

**Computer-Programming Factors**

Section D of the survey asked respondents to list the importance of each of the factors in mastering programming. The respondents gave each of the 32 factors a rating from one to five where one indicates Strongly Disagree and five indicates strongly agree. The responses with the highest scores are listed in Table 5.6.

The scores are the aggregates of ratings given by the thirteen students. For example, a score of 52 indicates that thirteen students on average gave the factor a rating of four.
This is the average response for ‘method of teaching computer programming’. Thus, on average the 13 students agreed with the experts.

<table>
<thead>
<tr>
<th>Factor</th>
<th>Aggregate Score</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prior experience with computers</td>
<td>59</td>
<td>4.54</td>
</tr>
<tr>
<td>Ability to break down problems into smaller tasks</td>
<td>58</td>
<td>4.46</td>
</tr>
<tr>
<td>Clear thinking</td>
<td>58</td>
<td>4.46</td>
</tr>
<tr>
<td>Problem-solving ability</td>
<td>58</td>
<td>4.46</td>
</tr>
<tr>
<td>Tutorials</td>
<td>58</td>
<td>4.46</td>
</tr>
<tr>
<td>Perseverance</td>
<td>58</td>
<td>4.46</td>
</tr>
<tr>
<td>Enthusiasm about computers</td>
<td>57</td>
<td>4.38</td>
</tr>
<tr>
<td>Motivation to succeed</td>
<td>57</td>
<td>4.38</td>
</tr>
<tr>
<td>Ability to analyze problems</td>
<td>56</td>
<td>4.31</td>
</tr>
<tr>
<td>Discussion with peers</td>
<td>56</td>
<td>4.31</td>
</tr>
<tr>
<td>Ability to construct algorithms</td>
<td>55</td>
<td>4.23</td>
</tr>
<tr>
<td>Ability to modify or re-use existing programs</td>
<td>54</td>
<td>4.15</td>
</tr>
<tr>
<td>Ability to read and understand programs</td>
<td>54</td>
<td>4.15</td>
</tr>
<tr>
<td>Previous experience with application programs</td>
<td>53</td>
<td>4.08</td>
</tr>
<tr>
<td>Method of teaching computer programming</td>
<td>52</td>
<td>4.00</td>
</tr>
</tbody>
</table>

Table 5.6: Factors in Mastering Computer Programming

The students agreed or strongly agreed that each factor in the table was important to their mastery of programming (the means range from 4 to 4.54). Apart from ‘ability to play computer games’ the statements not shown in Table 5.6 were rated between 3 and 4. This indicates that responses ranged on average from unsure to agree. For ‘ability to play computer games’ the mean response was 2.38.

Interest in arcade games was one item in a list of indicators of experience obtained by Chambers and Clarke (1989) from a review of the literature. The students denied the relevance of playing computer games. It is interesting to note that the experts in the cognitive mapping session did not raise game playing.
5.4.4 Results of Aptitude Test

Recall that part of the final examination assessed students' aptitude for programming. Scores on the aptitude questions were disaggregated from the examination result. This was done for all students who sat the examination. Table 5.4 shows correlations between the two and of each with the overall score obtained in the prerequisite unit BIS.

<table>
<thead>
<tr>
<th>Correlation</th>
<th>Examination performance less aptitude component</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>All Students</td>
</tr>
<tr>
<td></td>
<td>(n = 28)</td>
</tr>
<tr>
<td>Performance in BIS</td>
<td>0.65 **</td>
</tr>
<tr>
<td>Aptitude score</td>
<td>0.28</td>
</tr>
</tbody>
</table>

** Indicates correlation is significant at the 1 percent level

Table 5.7: Aptitude, Computing Performance, Programming

All the correlations are positive and all but one are greater than 0.50. Only the correlation between BIS and programming score net of aptitude score is significantly different to zero (at the one percent level). That is, there is evidence of an increasing relationship between the programming and BIS results. For the interviewed students, the correlation coefficient is not significant at even the five percent level. However, the magnitude of this coefficient is similar to that for all students. The lack of significance for the smaller group is possibly due to sample size.

Over all students who sat the examination the correlation between aptitude and examination mark net of aptitude score is positive (0.28). This is approximately half the value obtained for the interviewed students. There is no apparent reason for this difference other than inadvertent bias in selection of students for interview. Neither coefficient is significantly different to zero at the five percent level.

Aptitude is negatively correlated with locus of control, anxiety and playfulness for interviewed students. The magnitudes of the coefficients lie between 0.2 and 0.4. None are significantly different to zero.
5.5 Conclusions

In this chapter teachers and students were found to agree on determinants for success in learning to program. Some of these are investigated in the main study (Chapters 6 and 7).

Among thirteen business students, previous computing experience appeared to influence their results. For these students anxiety was on average reduced and playfulness was on average increased relative to students taking the introductory unit. The thirteen students also previously took the introductory unit. In Chapter 3 it was conjectured that anxiety would be reduced and playfulness raised by doing the earlier unit. This would seem to be the case. One qualification is required. Anxiety is reduced among the most anxious while playfulness appears to be raised across the distribution. Given that there is support for the conjecture only indicators of knowledge and experience are used in the main study. This was foreshadowed in the conclusion of Chapter 3.

An aspect of problem solving was assessed for twenty-eight business students including eleven of the thirteen mentioned above. The test involved series aptitude and following instructions through flowcharts. Correlations with examination performance were positive but insignificant. In light of the large literature and the emphasis of the focus group on problem solving this matter was investigated further in the main study.
CHAPTER 6
THE MAIN STUDY

6.1 Introduction

This chapter describes the main study. The three previous chapters described preliminary studies involving a survey of business and science students, a focus group session from which cognitive maps were generated, and the elicitation of information from business students undertaking their first units of programming (Figure 1.1).

An aim of the main study was to determine whether the factors identified from the preliminary study were related to computer-programming performance among different groups of students. The study involved a survey. In it, unlike the survey in the preliminary study students clearly identified themselves so that examination and assignment performances in programming could be collated with their survey responses.

The preliminary study was undertaken throughout 1994. The main study was undertaken in the second semester of 1995. This chapter describes the methodology of the main study. The results of the study are discussed in Chapter 7.

6.2 Background

Two groups of undergraduate students participated in the survey. The first were students undertaking programming studies within management information systems or business computing. The second were students undertaking programming as part of a computer science degree. Programming units may also be undertaken as electives or optional units.

Group 1: Business Computing Students and Systems Implementation

The students undertaking the first unit in programming were generally majoring in business computing. During first semester they generally took three of the common sequence of business units as well as the introductory computing unit.

Prior to 1995, the emphasis in this programming unit had been on algorithms and implementation using the Cobol programming language.

7 The survey used is in Appendix D.
8 Business Information Systems
Under the name of *Software Development* a previous group of programming students described this version of the unit in the preliminary study interviews. In 1995 the content of the unit was revised, the name was changed to *Systems Implementation* and the emphasis was altered to design and writing of programs for microcomputers using Visual Basic. The sequence of units taken by the business computing students is shown in Figure 6.1

**First Semester**

**Second Semester**

**Figure 6.1: Normal Study Sequence for Business Computing**

*Systems Implementation* was designed to develop expertise in the implementation of software applications in a Windows-based, event-driven environment. The specific areas included:

- specification of application requirements
- the design of interfaces using both text and graphics
- data storage, retrieval and manipulation using a variety of files and data structures
- data normalization for relational databases
- program and event-driven procedures
- system maintenance and enhancement techniques
- system building tools
- design and implementation for maximum flexibility
- testing and documentation.

This unit was delivered in the form of one two-hour lecture and a three-hour workshop. The assessment was by examination (60 percent) and assignment (40 percent).
Group 2: Computer Science Students doing Basic Programming Concepts and Data Structures and Algorithms

The students undertaking these units were generally majoring in computer science. During the first semester of their studies they had usually undertaken the first computer-programming unit, Basic Programming Concepts and one of two computing (non-programming) technology units, either Microcomputer Applications or Introduction to Information Technology. Those students who had undertaken computing studies at the secondary school level enrolled in Microcomputer Applications while those who had no previous secondary school computing experience enrolled in Introduction to Information Technology.

First Semester

- One of two Information Technology Units
- Basic Programming Concepts

Second Semester

- Data Structures and Algorithms

Figure 6.2: Normal Study Sequence for Computer Science

The unit Basic Programming Concepts introduced operating systems and program development. Structured programming, modularization, problem solving and developing algorithms were emphasized. The C programming language was studied.
Data Structures and Algorithms extended the concepts introduced during the first unit in programming. The major data structures – stacks, queues, lists and trees – were introduced. Other topics included comparison of algorithms, iteration and recursion, searching and sorting, larger data structures, dynamic structures and program structures.

Both units were delivered in the form of three one-hour lectures, a one-hour tutorial session and a two-hour practical session. Assessment was by examination (60 percent) and continuous assessment (40 percent).

6.3 Research Method

Permission to survey students was obtained from the University Ethics Committee. Surveys were completed during classes in Week 9 of the 13-week second semester. As noted above the units were undertaken by distinct groups of students on different campuses of the same university.

At the time of the survey 80 students were enrolled in the business computing unit. Sixty-one students attended the class in Week 9 and all returned the questionnaire. The 61 returned surveys represents 76 percent of the enrolment. Two students completed the survey but did not sit the examination. For the purposes of the regression analysis in Chapter 7 these two students were eliminated from the sample.

Sixty-nine students sat the examination in this unit. All of them attempted the assignments set throughout the semester. It is difficult to know how many of the 80 enrolments were actively studying the unit at the survey date. If it is conjectured that 69 is the number actively involved in Week 9 then the response rate to the survey jumps to 86 percent.

The enrolment in Data Structures and Algorithms (the ‘science unit’) at the time of the survey was 68. Forty-two students attended the class and all completed the surveys in Week 9. The 42 responses represent 62 percent of total enrolments. One student who completed the survey did not do any assignments nor did that student sit the examination and was therefore omitted from the sample. One other student did not do assignments but did sit the examination and was included in the regression analysis.
Obviously only students in attendance on the day could take part. However of the 68 total enrolments, 11 students did not do the assignments nor did they sit for the examination. One of these 11 students completed a survey (see above). Exclusion of all 11 gives a response rate of 41 out of 57, or 72 percent.9

In this regard the groups of business and science students were selected on the basis of three common criteria:

1. A numeric mark was available for programming performance.
2. The students attended the class in Week 9.
3. The students submitted completed questionnaires.

Recall from Chapter 4 that a group of experts agreed that learning style was a key construct influencing the goal of basic academic ability (Map 4.2). They agreed that it is important to understand students' learning traits. They did not discuss what the effective style would be. Among the experts were lecturers involved in teaching either business or science students. They did not suggest that different styles might be effective in the science and business areas. However, the videotape of the experts' meeting reveals that they thought learning style might be a barrier to students from humanities and arts.

To investigate learning style, three methods were considered. An instrument by Honey and Mumford (1992) was thought (by the experts) to work for managers but was unsuitable for the student groups. The Myers-Briggs Type Indicator (Myers and McCaulley 1985) was costly and is not an instrument for directly determining learning style. The Learning Style Instrument of Kolb (1985) used in the survey, was inexpensive and aimed directly at assessing how individuals learn.

6.4 The Survey

The survey consisted of four sections:

1. Learning Style Inventory (Kolb 1985)
2. Questions on computer-programming experience and expectations relating to:
   - Knowledge of programming
   - Enjoyment of programming
   - Job ambitions

---
9 One other questionnaire was unusable because of incompleteness.
3. Exercises in problem solving
4. Questions on educational background relating to
   • Preference for the course at enrolment
   • Secondary school achievements in mathematics, English and information technology
   • Overall performance at secondary school.

The first page of the survey described the purpose of the research and included a consent form (Appendix D). The latter asked for name and address details and signature of the participant. Thus for each survey the respondent could be identified.

6.4.1 Learning Style

Learning styles are ways that individuals gather, store and retrieve information. At various stages in the life cycle, it is argued that a person prefers a particular style. Kolb (1979) posits that the most efficient and effective learning method should be the one that corresponds to an individual’s primary learning style. Kolb theorized that learning is a four stage cycle — concrete experience (feeling), reflective observation (watching), abstract conceptualization (thinking), and active experimentation (doing).

Kolb developed the Learning Style Inventory (LSI11) in 1976 and revised it in 1985 (LSI12). His instruments yield four scores representing the individual’s emphasis on each learning stage. Two combination scores are calculated – the first indicating the extent to which an individual emphasizes abstractness over concreteness and the second indicating the extent to which an individual emphasizes action over reflection. ¹⁰

There is evidence that certain learning styles characterize certain occupations and groups. A person who enjoys a career in computer science is likely to have a learning style that is high on the abstract conceptualization scale and high on the active experimentation scale. An Australian study of science and arts students by Willcoxson and Prosser (1996) found that science students, including those majoring in computer science, had a significantly higher score on the active experimentation scale.

¹⁰ Equations defining these measures are given in Chapter 7.
In Singapore, a study of 1032 final year undergraduates by Yeun and Lee (1994) found that the computer science cohort were high on the abstract conceptualization scale. Kolb and Smith (1986) suggest that there is a correspondence between learning style (acquired from earlier education) and undergraduate major. Some studies attempt to correlate individual learning style with learning and/or achievement in the computing domain. Scin and Robey (1991) administered the LSI2 to 80 novice computer users. They found that those individuals who had high scores on abstract conceptualization and active experimentation performed with greater accuracy than subjects with other learning styles.

A study of 94 undergraduates by Hudak and Anderson (1990, p. 233) found that ‘...the concrete style of learning, marked by lacking the use of theory and inference, is particularly maladaptive in statistics and computer science courses’.

Other studies have used a different learning style instrument, for example the Gregorc Style Delineator (1984). Thronson (1984) and Davidson, Saveyne and Orr (1992) used the Gregorc instrument. Thronson did not find any relationship between student learning style and achievement in beginning programming instruction. In the Davidson, Saveyne and Orr study, performance in computing applications was measured against student learning style and it was found that students who were abstract learners achieved higher total course points than those who were concrete learners.

Some studies used LSI2 to specifically examine the relationship between learning style and performance in programming. Myers and Munsinger (1996) investigated the relationship between learning style and programming achievement in two paradigms: imperative and functional. General ability in programming proved a much stronger indicator than learning style on success in programming courses. Wu (1993) investigated the effects of conceptual models and cognitive learning styles in 200 students in an introductory computer science course. He found in a study of recursion that students with an abstract learning style performed better than those with a concrete learning style did. The LSI2 used in this study consisted of twelve sentences with a choice of four endings. Respondents were asked to rank these using 4 for the sentence ending that described how they learned best down to 1 for the sentence ending that seemed least like the way they would learn. Respondents were asked to rank all endings and not to make ties.
An example of a sentence with endings is:

Q5. When I learn:

—— I am open to new experiences
—— I look at all sides of issues
—— I try to analyze things, break them down into their parts
—— I like to try things out.

Each of the endings correspond to the four learning modes – Concrete Experience (whose characteristic word is feeling), Reflective Observation (watching), Abstract Conceptualization (thinking) and Active Experimentation (doing).

6.4.2 Knowledge and Experience of Programming

Curtis et al. summarized the research to 1986 on individual differences among programmers. In their view, breadth of relevant experience better predicted performance than did length of experience as the former produces a broader understanding. However, Curtis et al. (1986, p. 1093) note that other researchers found length of experience did better in explaining performance than academic results and measures of ability.

Nowaczyk (1984) found that prior computer experience was a significant explanator of performance in an introductory programming unit where the language Watfiv was taught. On the other hand Nowaczyk (1984, p. 150) did not report a link between previous experience and course performance in more advanced units. This was consistent with his expectation that students in advanced courses should possess basic skills. Thus, to the extent that experience is a proxy for the acquisition of basic skills, early experience should diminish in importance for advanced study.

Chen and Vecchio (1992) concluded that usage of computers affects attitudes such as computer anxiety. These attitudes in turn influence success in end-user computing. They defined end-user-computing to mean ‘... users who develop, implement, maintain and utilize their own computer applications using high-level procedural languages, fourth-generation languages or a language which is part of a software package’ (ibid. p. 843). This definition of end-user programming is consistent with some of the aims of the business computer-programming unit (Section 6.2).
Recall from Chapter 4 that a focus group decided basic academic ability was a goal influencing performance in programming. In Map 4.1 prior experience with computers was seen by the experts as influencing knowledge of computers which in turn influenced interest or exposure to the end-use of computers. Via relevance to larger goals, reason for studying and personal motivation, the experts saw a link to performance.

A theme that emerged from the interviews with programming students (Chapter 5) was that some students were 'at home' and seemed to know what was going on in the unit. Others (including some of the interviewees) were 'at sea'. Recall the evidence in Table 5.3 that a little experience with programming at secondary school has a substantial effect on performance in business programming.

There were nine questions concerning the respondents' knowledge and experience. Some questions required the respondent to tick the appropriate box. Other questions required short written answers (Appendix D).

The respondents were asked to describe their knowledge of programming prior to attending their current course at university. If they indicated they had some knowledge, they were asked where they had obtained this knowledge and which programming languages they had used. They were also asked how they assessed their knowledge (in Week 9). The respondents were asked to indicate whether they used computers in paid employment. If they did, information was sought on the type of involvement. They were also asked how much they enjoyed programming, what grade they expected to get at the final examination and what job expectations they had.

6.4.3 Problem Solving

The relationship between specific problem-solving abilities and learning programming has been widely investigated (Chapter 2). Shute (1991) for example found that programming ability depended on skills in algebra, problem identification and sequencing. Above all her research emphasized cognitive abilities of working memory capacity, that is, the degree to which individuals can handle information in temporary storage while simultaneously processing new information.

---

11 He investigated two advanced courses - one in Cobol and the other an advanced computer science course.
Other research has shown a lack of support for the link between learning programming and problem solving. Reed and Palumbo (1988) suggest that problem-solving skills take time to develop and are hard to measure. Sufficient time may not have been available in earlier learning.

The experts in the focus group identified problem-solving skill as a goal in achieving success in elementary programming (Chapter 4). Strategic directions for achieving skill in problem solving they concluded were clear thinking, logic, working methodically, thoroughness and a capacity for precision. The 13 students interviewed rated problem solving and clear thinking highly on the list in Table 5.6.

Section 3 of the survey asked respondents to answer nine questions designed to test their problem-solving ability. The questions came from a variety of sources. Table 6.1 lists the type of questions and the sources.

<table>
<thead>
<tr>
<th>Question Number</th>
<th>Type</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 and 2</td>
<td>Logical Structure</td>
<td>Poplin et al. (1984)</td>
</tr>
<tr>
<td>3 and 4</td>
<td>Series Aptitude</td>
<td>Poplin et al. (1984)</td>
</tr>
<tr>
<td>6</td>
<td>Inference Problem with topological properties</td>
<td>Unknown but type cited in Wickelgren (1974)</td>
</tr>
<tr>
<td>7</td>
<td>Follow directions via flowchart</td>
<td>NCR Aptitude Test E51 (circa 1973)</td>
</tr>
<tr>
<td>8</td>
<td>Transformational/State-space</td>
<td>Nowacyzk (1984)</td>
</tr>
<tr>
<td>9</td>
<td>Algebra Word (Representation)</td>
<td>Mayer (1975)</td>
</tr>
</tbody>
</table>

Table 6.1: Problem-solving Questions

The first four questions were adapted from CALIP: Computer Aptitude, Literacy, and Interest Profile (Poplin et al. 1984). The goal of CALIP is to identify talented individuals who may want to specialize in a computer-related career, particularly to identify persons with a high potential for programming.
CALIP specifically identifies four aptitudes: estimation, graphic patterns, logical structures and series aptitude. Two questions, each from the logical structures and series aptitude, were included. Estimation involved a timed test and was therefore not suitable for this survey. Questions on graphic patterns required additional explanation and complex graphical drawings. They were excluded.

The logical structures questions encourage diverse thinking. They identify a person’s ability to find similarities between two apparently different pairs or groups of items. These items are words, letters or numbers, typically with an implicit logical pattern.

Series aptitude on the other hand rewards a person able to detect order in a letter or number sequence. The questions require some simple arithmetic in order to detect some order in strings.

Questions 5 and 9 are known as representation problems. They are typically found in mathematics texts. The problems require the translation of written problems into formulas. Question 9 was cited in Mayer (1975). He concluded that it was the strongest overall predictor of performance in a test of general quantitative reasoning.

The Loch Ness monster problem of Question 5 was created by Gardner (1978) and used by Nowaczyk (1984). Nowaczyk found that performance was significantly correlated to final grade in a study of 286 computer-programming students (0.20 and significant at less than 5 percent level).

The sixth question is an example of an inference problem where to solve the problem; an individual needs to recognize implicit information. Inference is usually one of the first methods used in solving a problem and ‘... what is necessary is to make that one critical transformation of the givens that essentially solves the problem’ (Wickelgren 1974, p. 23). This question required respondents to recognize the topological properties of a square. The source of the particular problem is unknown.

Question 7 was a flowchart derived from The NCR Aptitude Test E51 (circa 1973). It was one of the examples used and discussed in the front of the NCR booklet. The answer is obtained by following step-by-step instructions, requiring the ability to work methodically, a step by step approach and thoroughness – all attributes identified by the focus group in Chapter 4.
The transformational or state-space problem of question 8 was adapted from Greeno (1978) as described in the Nowaczyk (1984) study. This type of problem ‘... may offer a challenge to the programmer. First there may be more than one solution, second iterative and recursive procedures are often used and third the choice of legal operators that can be applied at various intermediate stages is not limited to one’ (ibid. p. 151). Wickelgren (1974) suggests that this type of problem can also be classed as the sub-goal method of solving problems where breaking up a problem into parts or analyzing the problem into sub-problems in order to facilitate solving the original problem. The experts in the cognitive maps suggested these attributes and others of that type.

In the Nowaczyk (1984) study of 286 programming students the relationship between course grade and the mark for the transformational problem was 0.17 with a significance of 5 percent or less.

Except for multiple-choice questions one to four, written answers were required.

6.4.4 Background and Education

Respondents were asked to indicate whether they were enrolled full-time or part-time. They were also asked to provide details of the year they were born. If they had completed school the previous year, their results for English, mathematics and information technology subjects were sought. They were asked to indicate what preference they had given the course before enrolment, what course and university they would have chosen had they had the necessary requirements for entry and whether they were planning to make any changes regarding their enrolment. The reason for requesting this information was to establish a de-facto indicator of student motivation. Students who gave their course a high preference, who wanted to come to this university to do this particular course and who were satisfied with their choice (no change of enrolment), were more likely to outperform those who enrolled because it was 'the one that they had got into'. This also applies to those who commenced studies but were hoping to change.
6.5 Looking Forward

The survey responses and regression analysis are in Chapter 7. From the regressions some clear influences on success emerge. Some variables investigated widely in the past do not matter. As noted already, the survey was applied to two samples consisting of students committed to satisfactorily completing the units. The respondents were attending classes in Week 9 of a 13-week semester. They did assignments (with one exception) and all sat the examination. It is likely that choosing such samples affects which variables have appreciable influences on performance. Against this may be set the opportunities to compare different types of programming students and to pursue the development of skills over all units in the first year.
CHAPTER 7

RESULTS OF THE MAIN STUDY

7.1 Introduction

In this chapter the survey described in Chapter 6 is used to compare outcomes in computer programming among students of business and of science.

The surveys were collated and identification numbers written on each of them. The data were keyed in for use with SPSS and MINITAB. Computer-programming performances in both the end-of-semester examination and overall performance were added to the file at the end of the semester. Academic histories of respondents were compiled and used to verify some details on background and education (see Section 6.4). Also added were scores of all other units taken at tertiary level.

First in Section 7.2 the survey responses are summarized. The regression analysis is described in Section 7.3. Finally a discussion of the regression results is in Section 7.4.

7.2 Comparison of Business and Science Students

The main survey was administered to business and science students. As discussed in Section 6.3, 59 surveys from business students and 40 surveys from science students were retained. These numbers correspond to response rates of 76 percent and 70 percent respectively. Recall that the surveys were conducted during classes in Week 9 of the semester. Most of the enrolled students who did not attend the classes were not actively involved in studying the units. That is, they neither did the assignments nor sit the examination. In the following sub-sections the responses to each section of the survey are analyzed separately.

7.2.1 Demographics of the Two Groups' Responses

Table 7.1 shows background variables for the two groups. The data on degree preference and secondary schooling were incomplete.

Compared with the computer science students, the business computing students were older and more likely to be studying part-time. Males were slightly more concentrated in science than in business.
<table>
<thead>
<tr>
<th></th>
<th>Business Computing</th>
<th></th>
<th>Computer Science</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number</td>
<td>Percent</td>
<td>Number</td>
<td>Percent</td>
</tr>
<tr>
<td><strong>Age</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Less than 20</td>
<td>24</td>
<td>40.7</td>
<td>16</td>
<td>40.0</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>6.8</td>
<td>4</td>
<td>10.0</td>
</tr>
<tr>
<td>21</td>
<td>4</td>
<td>6.8</td>
<td>4</td>
<td>10.0</td>
</tr>
<tr>
<td>Older than 21</td>
<td>27</td>
<td>45.7</td>
<td>16</td>
<td>40.0</td>
</tr>
<tr>
<td><strong>Gender</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Male</td>
<td>37</td>
<td>62.7</td>
<td>27</td>
<td>67.5</td>
</tr>
<tr>
<td>Female</td>
<td>22</td>
<td>37.3</td>
<td>13</td>
<td>32.5</td>
</tr>
<tr>
<td><strong>Study Mode</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Full-time</td>
<td>52</td>
<td>88.0</td>
<td>39</td>
<td>97.5</td>
</tr>
<tr>
<td>Part-time</td>
<td>7</td>
<td>12.0</td>
<td>1</td>
<td>2.5</td>
</tr>
<tr>
<td><strong>Secondary School</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Previous year</td>
<td>21</td>
<td>35.6</td>
<td>23</td>
<td>57.5</td>
</tr>
<tr>
<td>More than 1 year ago</td>
<td>18</td>
<td>30.5</td>
<td>10</td>
<td>25.0</td>
</tr>
<tr>
<td>Missing</td>
<td>20</td>
<td>33.9</td>
<td>7</td>
<td>17.5</td>
</tr>
<tr>
<td><strong>Preference for Degree</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>First</td>
<td>21</td>
<td>35.6</td>
<td>13</td>
<td>32.5</td>
</tr>
<tr>
<td>Second</td>
<td>15</td>
<td>25.4</td>
<td>6</td>
<td>15.0</td>
</tr>
<tr>
<td>Third</td>
<td>2</td>
<td>4.3</td>
<td>4</td>
<td>10.0</td>
</tr>
<tr>
<td>Four or more</td>
<td>8</td>
<td>13.6</td>
<td>10</td>
<td>25.0</td>
</tr>
<tr>
<td>Missing</td>
<td>13</td>
<td>22.0</td>
<td>7</td>
<td>17.5</td>
</tr>
</tbody>
</table>

Table 7.1: Student Demographics
Only 36 percent of the business computing students indicated that they had completed secondary school studies in the previous year. For the computer science group this figure was about 20 percent higher. Similar proportions obtained their first preference in choice of course. However 25 percent of the computer science group indicated a much lower preference for the course that they were now in (preference of four or more) compared with the business-computing group.

7.2.2 Learning Style

Kolb’s Learning Style Inventory (1985) provides four measures. The survey used the revised instrument (LSI2 – see Section 6.4.1). Table 7.2 shows the resulting raw scores for the four basic scales, CE, RO, AC and AE. With these indicators individuals are placed in a two dimensional spectrum as to how they perceive and process new information:

...learning requires abilities that are polar opposites, and... the learner must continually choose which set of learning abilities he or she will use in a specific learning situation. More specifically there are two main dimensions to the learning process which correspond to the two major different ways by which we learn: the first is how we perceive new information or experience, and the second is how we process what we perceive (Kolb and Smith 1986, p. 13).

Each of the measures is described as follows. Individuals with high scores for CE (Concrete Experience) perceive ‘...through their senses, immerse themselves in concrete reality and rely heavily on their intuition, rather than step back and think through elements of the situation analytically’ (ibid.). Compare this with AC (Abstract Conceptualization) where individuals prefer to analyze or plan in preference to relying on their intuition or sensations. AC and CE define a continuum spanning the concrete and abstract approaches to perceiving new information. A learner’s choice of approach can be placed on this one-dimensional scale.

How individuals process information is reflected in the score in LSI2 that an individual receives for the measures AE and RO. ‘...doers favor Active Experimentation [AE] while ...watchers favor Reflective Observation [RO]’ (ibid. p.14).
In everyday activities individuals will choose between a relatively concrete or abstract approach when taking in new information. They will subsequently process that new information in a relatively active or reflective way. The factors that determine the approach taken are personality type (such as introvert or extrovert), undergraduate major (business major compared with science major), career choice (such as norms or standards of conduct in a professional field) and current task or problem (effectively resolved with particular skills).

Two combination scales which indicate the extent to which the individual emphasizes abstractness over concreteness (AC – CE) and the extent to which he or she emphasizes actions over reflection (AE – RO) appear in Table 7.2. For comparison, the findings of Kolb and Smith on a sample of American professionals are also reproduced.

<table>
<thead>
<tr>
<th>Learning Style Inventory</th>
<th>Business Computing</th>
<th>Computer Science</th>
<th>Kolb and Smith (1986)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>S.D.*</td>
<td>Mean</td>
</tr>
<tr>
<td>Concrete Experience (CE)</td>
<td>25.2</td>
<td>10.3</td>
<td>25.9</td>
</tr>
<tr>
<td>Reflective Observation (RO)</td>
<td>30.1</td>
<td>6.4</td>
<td>29.6</td>
</tr>
<tr>
<td>Abstract Conceptualization (AC)</td>
<td>31.4</td>
<td>7.4</td>
<td>31.6</td>
</tr>
<tr>
<td>Active Experimentation (AE)</td>
<td>33.4</td>
<td>7.2</td>
<td>33.0</td>
</tr>
<tr>
<td>Relative Abstractness (AC-CE)</td>
<td>6.2</td>
<td>15.7</td>
<td>5.6</td>
</tr>
<tr>
<td>Relative Activity (AE-RO)</td>
<td>3.3</td>
<td>10.4</td>
<td>3.3</td>
</tr>
</tbody>
</table>

* S.D. = Standard Deviation

Table 7.2: The Learning Style Inventory

First note in Table 7.2 the similarity of the means for the four scales across the business, science and the Kolb and Smith (1986) samples. The greatest difference is that on average the business and science students less actively experiment than do the third group. The Australian groups display slightly higher abstract conceptualization. Differences in standard deviation also appear. The standard deviation for the two student groups on concrete experience is much greater than for the American professionals.
Row 5 of the table contains results for relative abstractness. As presented, the means and standard deviations decrease from left to right across the row. For relative activity the mean in the third column exceeds the other two. Overall, working American professionals appear to be relatively less abstract and relatively more active than either group of Australian students.

The mean scores on the two relative measures for the two Australian groups are not significantly different at the five percent level on a 2-tailed test. That is, there would appear no substantive difference between the two groups on average. Research on LS12 found that there is a relationship between individual learning style and the careers that people choose (ibid. Section 7). Australian students in faculties as far apart as business and science did not display learning style indicative of different career destinations.

With the measures AC – CE and AE – RO learning style can be shown in a two-dimensional diagram. Each quadrant represents a learning style. These are diverger (Quadrant 1), accommodator (Quadrant 2), converger (Quadrant 3) and assimilator (Quadrant 4). The incidence of these learning-style types for the two student groups is set out in Table 7.3 and is shown in Figure 7.1.

<table>
<thead>
<tr>
<th>Learning Style Type</th>
<th>Business Computing</th>
<th>Computer Science</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number</td>
<td>Percent</td>
</tr>
<tr>
<td>Accommodator</td>
<td>4</td>
<td>6.8</td>
</tr>
<tr>
<td>Diverger</td>
<td>17</td>
<td>28.8</td>
</tr>
<tr>
<td>Converger</td>
<td>20</td>
<td>33.9</td>
</tr>
<tr>
<td>Assimilator</td>
<td>16</td>
<td>27.1</td>
</tr>
<tr>
<td>Unknown (missing)</td>
<td>2</td>
<td>3.4</td>
</tr>
</tbody>
</table>

Table 7.3: Learning Style Types
Convergers occur most often in the business group while divergers and assimilators are about equally represented. Among the computer scientists, assimilators are the dominant group while divergers and convergers are roughly equally represented. Accommodators are the smallest sub-group in each sample.

According to Kolb and Smith (1986) computer science and data processing careers are in the Converger/ Assimilator area. Both samples have a predominance of these two learning styles: 61 percent for the business-computing students and 62 percent for the computer science students.

The learning styles for the business and science students are indicated in Figure 7.1 (adapted from Kolb and Smith (1986)). The axes intersect at the mean scores of relative abstraction and relative activity for a sample of 1446 American adults aged between 18 and 60 in a wide range of career fields and ethnicity. On average, members of this 1985 sample had two years of college education.

On the figure are the learning styles of some professional groups. In the original more professions are shown than are included here. Professions included relate to likely career destinations for students in the two groups.

Except for Accounting, none of the likely career destinations fall in the diverger or accommodator categories. That is most career destinations lie below the horizontal axis (Kolb's population means for relative concreteness). The means for the two samples display relatively greater abstraction with both falling in the assimilator quadrant. However, the students appear to be less abstract and relatively less active than the professions of computer science and data processing.
Figure 7.1: Learning Style and Occupational Field
7.2.3 Computer-Programming Knowledge and Experience

Each of the questions relating to experience was coded. For example, question 1 required the respondent to provide self-evaluation of their programming knowledge by ticking one of five boxes. Values for self-evaluated knowledge therefore ranged from 4 (very good) to 0 (none). Table 7.4 shows the summary of responses.

Nearly 56 percent of the business students said they had minimal or no knowledge of programming before coming to university compared with 45 percent of science students. Relatively fewer science students (22.5 percent) rated their prior knowledge as good or very good compared with business students (30.5 percent). These findings may reflect differences in the age distribution for the two groups.

Thirty-nine percent of business students in Week 9 rated their knowledge as good or better. The same proportion of business students claimed only average knowledge. Among the science students 40 percent rated their knowledge as good or better, while a higher proportion (compared with business students) gave average as a response. Just under half of the business students thought that programming was OK or did not like it at all. On the other hand about a third of the science students were neutral or disliked programming. Smaller proportions of both samples expected only to pass.

Between 55 and 60 percent of students in the samples were contemplating a computing or related career. Interestingly 39 percent of the business students and 35 percent of the science students expected a distinction or a high distinction for the programming units. Usually in programming units the proportions of high distinctions are much less that those expected among the students.
| Q1. Prior knowledge | Business Computing | | Computer Science | |
|---------------------|--------------------|----------------|------------------|
|                     | Number | Percent | Number | Percent |
| Very good           | 7      | 11.9    | 3      | 7.5     |
| Quite good          | 11     | 18.6    | 6      | 15.0    |
| Average             | 8      | 13.6    | 13     | 32.5    |
| Minimal             | 11     | 18.6    | 5      | 12.5    |
| None                | 22     | 37.3    | 13     | 32.5    |
| Q6. Knowledge in Week 9 | | | | |
| Very Good           | 8      | 13.6    | 4      | 10.0    |
| Quite Good          | 15     | 25.4    | 12     | 30.0    |
| Average             | 23     | 39.0    | 19     | 47.5    |
| Minimal             | 13     | 22.0    | 3      | 7.5     |
| Q7. Enjoyment       |        |         |        |         |
|                     | Number | Percent | Number | Percent |
| Love it             | 10     | 5.1     | 3      | 7.5     |
| Like a lot          | 18     | 23.7    | 14     | 35.0    |
| Like a bit          | 14     | 23.7    | 9      | 22.5    |
| It's OK             | 14     | 30.5    | 8      | 20.0    |
| Don't like it at all| 3      | 16.9    | 5      | 12.5    |
| Q8. Expected Grade  |        |         |        |         |
|                     | Number | Percent | Number | Percent |
| High Distinction    | 8      | 13.6    | 6      | 15.0    |
| Distinction         | 15     | 25.4    | 8      | 20.0    |
| Credit              | 24     | 40.7    | 15     | 37.5    |
| Pass                | 12     | 20.3    | 8      | 20.0    |
| Fail                | 0      | 0.0     | 2      | 5.0     |
| Q9. Career          |        |         |        |         |
|                     | Number | Percent | Number | Percent |
| Computing or related| 34     | 57.7    | 22     | 55.0    |
| Non-computing/ don’t know | 21 | 35.6  | 14 | 35.0 |

1 Two computer science students did not answer this question.
2 One computer science student did not answer this question.
3 Four students from each of the groups did not answer this question.

Table 7.4: Responses to Programming Experience
7.2.4 Problem Solving

The first four questions in this section required choosing the correct answer from four possibilities. Each correct answer was given a value of 1. The answers to the remaining questions were scored as 2 if completely correct, 1 where some minor error had been made and 0 if incorrect. The latter questions were scored higher than the first four because they were more difficult and required written answers.

Many respondents found Question 7 difficult to follow. Some sought clarification while attempting this question. Others indicated their confusion on the survey form. Only two students answered correctly. Consequently this question was eliminated from the analysis.

A variable PROB was calculated as the sum of all scored answers. The maximum score for PROB was 12. The correlation coefficients between this and the examination mark are similar. For the business students the coefficient of 0.31 was significantly different to zero. In the smaller sample of science students the coefficient of 0.30 was not significant. Business students who did well on one of the algebra questions uniformly did well in the exam. Correctly answering the other algebra question tended to be correlated with examination mark for the science students.

7.2.5 Educational Background

Table 7.5 contains summaries of secondary school performance. In each subject the maximum score was 50. For the two samples there are the means for English, three mathematics subjects and two information technology subjects. The mathematics subjects vary in difficulty. The relative difficulty of each is indicated. Level 1 is the easiest. Tertiary Entrance Ranking (TER) is the criterion used to decide university entrance. It is a percentile ranking based on the final two years of secondary school. Another indicator of secondary school performance is ‘raw secondary score’. It consists of the score for English plus the scores of the next best three subjects plus 10 percent of the score of any other subject.

On average TERs were about 13 percentage points higher for business than for science. The spread of percentile ranks was half that of science. In general the business students did better in the final years of secondary school than the science students.
<table>
<thead>
<tr>
<th></th>
<th>Business Computing</th>
<th></th>
<th>Computer Science</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>n = 59</td>
<td></td>
<td>n = 40</td>
<td></td>
</tr>
<tr>
<td>Tertiary Entrance Ranking- TER</td>
<td>36</td>
<td>68.0</td>
<td>5.0</td>
<td>27</td>
</tr>
<tr>
<td>Raw Secondary Score</td>
<td>38</td>
<td>130.5</td>
<td>19.5</td>
<td>33</td>
</tr>
<tr>
<td>English</td>
<td>32</td>
<td>29.2</td>
<td>7.3</td>
<td>33</td>
</tr>
<tr>
<td>Further Mathematics: Level 1</td>
<td>8</td>
<td>37.9</td>
<td>2.4</td>
<td>11</td>
</tr>
<tr>
<td>Mathematics Methods: Level 2</td>
<td>11</td>
<td>29.6</td>
<td>2.8</td>
<td>12</td>
</tr>
<tr>
<td>Specialist Mathematics: Level 3</td>
<td>1</td>
<td>22.0</td>
<td>—</td>
<td>4</td>
</tr>
<tr>
<td>Information Processing and Management</td>
<td>16</td>
<td>34.7</td>
<td>4.4</td>
<td>18</td>
</tr>
<tr>
<td>Information Systems</td>
<td>5</td>
<td>31.6</td>
<td>7.7</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 7.5: Educational Backgrounds

In terms of raw secondary score these differences are considerably diminished. There is little difference between the samples on average in all but two subjects. These are the lowest level mathematics unit and Information Systems. The numbers providing results in individual subjects were however small.

For the two samples there were no significant correlations between mathematics scores at school and examination marks in programming. The Further Mathematics result tended to be negatively correlated with examination mark among the business students. There was a significant, negative correlation between English achievement at school and examination result for programming for the business group. Raw secondary scores were negatively correlated with examination result in programming (significant at the one percent level).

TER is positively correlated with examination result for programming for business students (significant at the five percent level). TER gives the percentage of all Year 12 students who did no better than a particular candidate over all subjects did. This measure obscures actual scores in Year 12.
Raw secondary score reflects actual scores in Year 12 subjects. This may account for the change of sign over the two correlations.

For the science students, correlations for English, TER and raw secondary score were positive. None was significant at the five percent level. For the four science students taking Information Systems, there is a close positive association between school result and university examination mark. Among the five business students who undertook the subject, this correlation is not evident. Table 5.3 showed previous experience of programming at school assured higher overall university grades. This observation, based on a sample of 13 student, appears contradictory. Perhaps an explanation lies in the small sample size in each case. The sample of 13 was enrolled in a business computing degree in the year before the group reported in this chapter.

Greater numbers took Information Processing and Management at school. The business students’ university examination mark is negatively correlated with information systems mark (significance for the two-tailed t-test is 5.9 percent). Among science students the corresponding correlation is not significantly different to zero.

7.3 Regression Analysis

In this section factors affecting success in programming are explored using multiple regression analysis. Dependent variables are either examination mark or a combined mark for programming performance consisting of weighted assignment and examination marks.

Examination marks were available for the programming units (Systems Implementation for the business students and Data Structures and Algorithms for the science students). Overall scores were available for these programming units and for Basic Programming Concepts undertaken by the science students in their first semester. In their first programming unit the business students studied Visual Basic while the science students studied C.\textsuperscript{12} A large number of independent variables were tried. These were selected from findings of preliminary surveys (Sections 3.4, 3.5 and 5.4), interviews (Section 5.4), a focus group session (Chapter 4) and aptitude testing (Section 5.4).

\textsuperscript{12} These units are described in Section 6.2.
Data for the regressions were gathered using the survey described in Chapter 6. Results are shown in Tables 7.7 to 7.9. Coefficients for only a small number of independent variables are shown. Others were tried but did not add significantly to the explanatory power of the models in F tests at the 5 percent level of significance. Exceptions are problem solving and gender which in some regressions added significantly to the models and in others did not. Generally these variables are not included in the models reported below. Significant associations occurred among a number of variables.

Students' knowledge of programming was significantly associated with job expectations on a Chi-squared test at better than the one percent level. The variables tried in regressions and rejected are shown in Table 7.6.

<table>
<thead>
<tr>
<th>Learning Style Scores</th>
<th>Programming Experience</th>
<th>Background and Education</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract - Concrete</td>
<td>Current Knowledge</td>
<td>Age</td>
</tr>
<tr>
<td>Active - Reflective</td>
<td>Computing in current job</td>
<td>Preference for Course</td>
</tr>
<tr>
<td>Concrete Experience</td>
<td>Expected Grade</td>
<td>Tertiary Entrance Ranking</td>
</tr>
<tr>
<td>Reflective Observation</td>
<td>Likely Job</td>
<td>Year 12 Score</td>
</tr>
<tr>
<td>Abstract Conceptualization</td>
<td></td>
<td>• English</td>
</tr>
<tr>
<td>Active Experimentation</td>
<td></td>
<td>• Mathematics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Information Technology</td>
</tr>
</tbody>
</table>

Table 7.6: Unhelpful Variables

7.3.1 Success in the First Programming Unit

Table 7.7 shows regression results for business and science students. These were obtained from MINITAB using the best sub-sets approach to select among lists of independent variables. The MINITAB linear regression procedure was then used to estimate coefficients and the diagnostics shown in the table. The samples used are smaller than the 59 business and 40 science students reported earlier. The reason is that values for the 'raw secondary score' variable could not be calculated for 11 business and 8 science students.
For business students both examination score and overall score were available and these are the dependent variables for the first two columns. For science students only overall score was available. Overall scores are weighted sums of examination and assignment marks.

Note first in Table 7.7 that gender and raw secondary score are shown in the three regressions. Among the business students gender is not significantly different to zero at the five percent level. However for the science students gender is significant at the five percent level and is an element of the best subset determined with Minitab.\(^{13}\)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Business Computing</th>
<th></th>
<th>Computer Science</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Exam Mark</td>
<td>Overall</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(\beta)</td>
<td>(t)</td>
<td>(\beta)</td>
<td>(t)</td>
</tr>
<tr>
<td>Average mark in other units</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Good Previous Knowledge</td>
<td>12.85</td>
<td>2.83 (*)</td>
<td>10.09</td>
<td>2.42 (*)</td>
</tr>
<tr>
<td>Gender</td>
<td>-4.83</td>
<td>-1.21</td>
<td>-1.82</td>
<td>0.50</td>
</tr>
<tr>
<td>Problem-solving Score</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Like Programming</td>
<td>15.98</td>
<td>2.08 (*)</td>
<td>8.57</td>
<td>1.22</td>
</tr>
<tr>
<td>Dislike of programming</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-12.50</td>
</tr>
<tr>
<td>Raw Secondary Score</td>
<td>-0.34</td>
<td>-2.93 (*)</td>
<td>-0.35</td>
<td>-3.37 (**)</td>
</tr>
<tr>
<td>Constant</td>
<td>91.08</td>
<td>5.56 (**)</td>
<td>102.62</td>
<td>6.82 (**)</td>
</tr>
<tr>
<td>(\bar{R}^2)</td>
<td>0.53</td>
<td>0.50</td>
<td>0.43</td>
<td></td>
</tr>
<tr>
<td>(F)</td>
<td>11.54 (**)</td>
<td>10.09 (**)</td>
<td>5.74 (**)</td>
<td></td>
</tr>
<tr>
<td>Durbin-Watson</td>
<td>1.73</td>
<td>1.69</td>
<td>2.35</td>
<td></td>
</tr>
<tr>
<td>Sample Size</td>
<td>38</td>
<td>38</td>
<td>32</td>
<td></td>
</tr>
</tbody>
</table>

1. \(\beta\) is the coefficient of the variable
2. \(t\) is the significance level (* is 5 percent level or better and \(**\) is 1 percent or better)

Table 7.7: Performance in Introductory Programming

\(^{13}\) An excellent description of the best subsets approach is in Berenson et al. (1983)
Gender is coded as 1 for male and 0 for female. Consequently, the negative coefficient of \(-4.83\) means that, all else being equal, males are estimated to have approximately a penalty of five examination marks and two overall marks. Among science students the coefficient is positive. Males on average received a bonus of 7.5 marks compared with females. The negative coefficients for the business group are significantly different to zero at no better than the 23 percent level. There is therefore a considerable likelihood that males and females, having other characteristics the same, do not perform differently overall or in the examination. There may be no difference on the basis of gender in the business group, but gender does matter in the environment in which science students operate.

Raw secondary score is significantly different to zero in all regressions. The magnitudes of the coefficients are of the same order but the signs are different for business and science. Among the business students an increase of one in the indicator of secondary school performance reduces the mark in university programming by about one-third of a mark. Thus if two students' raw secondary scores differ by ten then, all else being equal, their overall programming marks would differ by 3.5. The unexpected reduction may be due to the subjects the business students studied at secondary school. Many studied social science and humanities-based subjects. On the other hand, the science group largely studied mathematics, information technology and sciences such as chemistry and physics. For them performance is projected to increase with raw secondary score.

If the business students rated themselves as possessing a good knowledge of programming before coming to university then they received a premium of almost 13 marks in the examination and 10 overall. Among the 38 students in the regression for business, 29 percent rated their knowledge as good or better (compared with twenty percent of the science students). It appears that good previous knowledge matters more in the business environment. Previous knowledge was not part of the best subset of regressors for the science students. The business students expressing a liking for programming tended to receive a premium of approximately 16 examination marks.
Overall the premium is much smaller and not significantly different to zero at the five percent level. Among the science students *Like Programming* is not in the best subset of regressors but *Rather Dislike* is \(^{14}\). As would be expected, at least in this group, a dislike of programming brings a penalty.

In the regressions for the science group the average mark for other units studied concurrently with programming is positive and significantly different to zero at the five percent level. The coefficient is less than one. This suggests that every extra mark in other units, on average, converts to an additional third of a mark in programming. This variable was not included in the regressions for business because scores in other units were not available; only letter grades were available.

Among the science students it appears that an extra mark for problem solving is associated with rather more than one extra mark in the programming examination. Its coefficient is significantly different to zero at somewhat better than the ten percent level. Problem solving is not in the list of regressors for business computing. However there is a significant association (based on a Chi-square test) between gender and problem-solving score for the business group. Another regression for examination scores among the business students is shown in Table 7.8 where problem solving replaces gender. The diagnostics shown in the foot of the two columns are similar. In this regression the coefficient *Problem Solving* is 1.06. It is only significant at the 23 percent level.

On the basis of the results in these two tables it appears that different variables influence outcomes in business and science educational environments. Notably, gender and attitude to the study of programming (like or dislike) differently affect outcomes among the student groups. Another finding is that secondary school performance can erode programming results for business students. The explanation for this seems to rely on the subjects studied at secondary school.

\(^{14}\) Question 7 of Section 2 of the survey was ‘How much do you enjoy computer programming?’ Three dummy variables were constructed: *Like Programming* has the value 1 if the response was either *I love it* or *I like it quite a lot. I don’t like it at all* has the value 1 for the dummy *Dislike of Programming*. A third dummy indicating *I like it a bit or It’s OK* was constructed. This last variable was not included in the regressions. It specified the reference category.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Business Computing (with problem solving)</th>
<th>Business Computing (with gender)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\beta$</td>
<td>$t$</td>
</tr>
<tr>
<td>Gender</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Problem-solving Score</td>
<td>1.06</td>
<td>1.22</td>
</tr>
<tr>
<td>Good Previous Knowledge</td>
<td>13.95</td>
<td>3.08 **</td>
</tr>
<tr>
<td>Raw Secondary Score</td>
<td>-0.27</td>
<td>-2.28 *</td>
</tr>
<tr>
<td>Like Programming</td>
<td>13.91</td>
<td>1.83</td>
</tr>
<tr>
<td>Constant</td>
<td>72.72</td>
<td>3.91 **</td>
</tr>
<tr>
<td>$R^2$</td>
<td>0.53</td>
<td></td>
</tr>
<tr>
<td>$F$</td>
<td>11.55 **</td>
<td></td>
</tr>
<tr>
<td>Durbin-Watson</td>
<td>1.58</td>
<td></td>
</tr>
<tr>
<td>Sample</td>
<td>38</td>
<td></td>
</tr>
</tbody>
</table>

* $\beta$ is the coefficient of the variable

* $t$ is the significance level (*) is 5 percent level or better and (**) is 1 percent or better

**Table 7.8: Problem Solving and Gender: Business Students**

7.3.2 Programming Over Two Semesters

Recall that the survey was conducted when the science students were undertaking their second unit of programming but the third computing unit overall. The information technology unit and the first unit of programming were taken in the previous semester. Overall marks were available for all three units as was examination mark for the second programming unit.

This provided an opportunity to investigate whether attributes such as learning style mattered at different stages of the study of computing (Table 7.9). It might be that when the survey was completed (during the third unit) many of the characteristics of successful programmers were already developed. It might be further argued that because the students in the survey had done all three units and were specialized in computing, little can be learned from earlier units.
These criticisms are partly valid. However careful regression analysis reveals that some attributes are important in early units and some become important later. While revealing variations over the units, it will not show any uniform improvements across the sample in programming skills or characteristics.

The second column of results in Table 7.9 is reproduced from Table 7.6. Only the variable average score in other units appears in all of the models shown in the table. The coefficient of this variable is roughly the same for information technology and the first programming unit. However the coefficient for the second programming unit is about double for the examination and overall marks.

Explanations for this difference might be that:

(i) students are less interested in semester 1 assessment but are more interested in assessment modes by semester 2
(ii) the skills required in the third unit are more closely proxied by average mark than those employed in the previous units
(iii) there is easier assessment in the third compared with the previous units

Relative abstractness appears in the regression for the Information Technology unit. Increasing abstractness has a positive effect on overall performance. It does not feature in the other regressions. This is puzzling as the first and second units are studied concurrently. It may be that despite the finding of the focus group, thinking abstractly does not matter in learning the basics of programming in computer science. Further investigation is required to clarify this. When students study the third unit, programming knowledge prior to commencing university is important in examination outcomes but not in overall performance. An implication for education policy is that students might begin the study of programming before entering university. This finding is consistent with the calls of many information technology professionals who advocate the extensive teaching of programming skills in schools. Whether this training should be extended into primary school is an issue that requires further investigation.

The next is dislike of programming. It does not appear in the first column as this unit does not involve programming. As found in Table 7.7 for the unit Basic Programming Concepts, an aversion to programming detracts from performance. In the final regression it can be seen that a person who dislikes programming would have obtained a score which is 22 points lower than otherwise similar students.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Information Technology</th>
<th>Basic Programming Concepts</th>
<th>Data Structures and Algorithms</th>
<th>Exam Score</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Overall</td>
<td>Overall</td>
<td>Overall</td>
<td>Exam Score</td>
</tr>
<tr>
<td></td>
<td>β¹</td>
<td>t²</td>
<td>β¹</td>
<td>t²</td>
</tr>
<tr>
<td>Average in other units</td>
<td>0.39</td>
<td>4.36 **</td>
<td>0.32</td>
<td>2.07 *</td>
</tr>
<tr>
<td>Abstractness (AC-CE)</td>
<td>0.18</td>
<td>2.60 *</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Good previous knowledge</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Dislike of programming</td>
<td>-</td>
<td>-</td>
<td>-12.50</td>
<td>-2.52 *</td>
</tr>
<tr>
<td>Problem solving</td>
<td>1.04</td>
<td>2.13 *</td>
<td>1.36</td>
<td>1.84</td>
</tr>
<tr>
<td>Secondary Score (raw)</td>
<td>-</td>
<td>-</td>
<td>0.26</td>
<td>2.87 **</td>
</tr>
<tr>
<td>Gender</td>
<td>-</td>
<td>-</td>
<td>7.52</td>
<td>2.11 *</td>
</tr>
<tr>
<td>Constant</td>
<td>37.9</td>
<td>7.35 **</td>
<td>7.74</td>
<td>0.59</td>
</tr>
<tr>
<td>$\bar{R}^2$</td>
<td>0.65</td>
<td>0.43</td>
<td>0.50</td>
<td>0.50</td>
</tr>
<tr>
<td>$F$</td>
<td>16.03 **</td>
<td>5.74 **</td>
<td>20.03 **</td>
<td></td>
</tr>
<tr>
<td>Durbin-Watson</td>
<td>2.02</td>
<td>2.35</td>
<td>1.54</td>
<td>1.90</td>
</tr>
<tr>
<td>Sample Size</td>
<td>25</td>
<td>32</td>
<td>39</td>
<td>39</td>
</tr>
</tbody>
</table>

1 β is the coefficient of the variable
2 t is the significance level (* is 5 percent level or better and ** is 1 percent or better)

Table 7.9: Performance over Two Semesters: Science Students

Problem solving ability improves the information technology score, mark for mark. As noted above for Basic Programming Concepts the estimated coefficient is greater than 1, but the effect is not significant at the five percent level. In the third unit problem solving does not matter at all. That is problem solving ability appears to be of diminishing importance over the three units. To an extent the expert view that problem solving influences success in elementary programming is borne out.
Why problem solving diminishes in importance during the first year is not clear. Perhaps there is a convergence of problem-solving abilities by the conclusion of the third unit.

The variable raw secondary score is a continuous variable measuring performance in secondary school. It matters only for Basic Programming Concepts. The presence of this variable as well as the average score in other units, is suggestive that measures of ability proxied by the latter do not exhaust all of the factors important in the initial stages of programming. The correlation between the two measures is low in our data. It is possible that one or other of these variables is capturing elements of problem solving ability. On its own our measure of problem solving was not significant.

For the first programming unit the individual characteristic of gender had a direct effect on performance, with males obtaining an eight-point premium approximately. The influence of gender was insignificant in the regressions for performance in the third unit. Recall that gender was not a significant explanator in the comparable regression for business students (Table 7.7). In our results there is no evidence of continuing effect due to gender. It would appear that by the time Data Structures and Algorithms is studied the science environment is also gender neutral. It may be that women become acclimatized to the science environment after one unit of programming. In terms of human resource strategy, policies are required to allow females to acclimatize. This might be in addition to strategies designed to assist females to deal with the problems they confront when first they study programming.

7.4 Conclusion

Multiple regressions were undertaken to determine whether variables relating to demographic composition of the sample, academic ability, learning style, problem solving and indicators of personal motivation influenced performance in computing units. For two samples the incidence of these characteristics are discussed in the early sections.

Broadly, indicators of basic academic ability mattered across four units, three of which were taken by science students and one by business students. This was one of the four goals identified by the focus group (Chapter 4).
Two other goals, problem solving and indicators of personal motivation appear in some regressions. These indicators appear not to be universally indicative of examination or overall score.

Previous experience in programming (for example at school) was of importance in the highest-level computing unit taken by either the business or science samples. Enjoyment of programming had divergent effects in the samples: for business students liking programming mattered positively. Among science students enjoyment was irrelevant but dislike was a strong negative influence.

The finding of no correlation between performances in mathematics and computer programming is surprising. Possibly, this result and the tendency to negative correlation of mathematics and computer programming among business students are associated with the small numbers of students studying mathematics beyond the elementary level.

Among the demographic variables, gender played different roles in the basic programming regressions. Despite a strong positive influence for science students, by the time the next unit was completed gender no longer mattered. This is an interesting finding as all students in the regression for basic programming were included in the subsequent estimation. The regressions provide little support for the idea that learning style affects outcomes in programming.
CHAPTER 8

CONCLUSIONS

The motivation for the research was to understand what factors affect performance in learning computer programming. The setting for the research was a university in Melbourne where programming is taught to undergraduates in two faculties: Business and Law and Science and Technology. There is debate about effective strategies, languages and teaching methods. The quantitative research that is reported in Chapters 3, 5 and 7 provides some evidence of outcomes for business and science students.

The research was conducted in two parts. The preliminary stage consisted of a survey of students doing an introductory unit in information technology, a focus group discussion and interviews with students doing their first programming unit. These investigations allowed the researcher to:

1. extend insights from the literature review to understand changes in the importance and emergence of factors
2. allow for local influences.

An example of changing importance is the decline in anxiety in using computers noted since earlier research in the 1980s. Preparation of students prior to university is one local influence. In the Victorian system Year 12 students do not necessarily take any information technology or programming studies. Another example concerns the intrinsic cultures of the faculties.

The variables computer anxiety, computer playfulness and locus of control were much researched in earlier decades. By the mid-1990s anxiety is reduced and playfulness is increased. There were correlations among these and other measures of experience in the preliminary survey. The correlations of anxiety and playfulness with knowledge and experience suggested that they could be omitted from the main study. Locus of control was largely independent of the other two variables and of measures of experience. The focus group and the interviewed students placed little emphasis on this indicator. From the preliminary study it was clear that indicators of experience in computing would influence assessment in programming.
There was agreement on a range of indicators between the literature and outcomes from the focus group. The focus group consisted of experts—university staff involved in teaching programming in first year to either business or science students. Of the agreed factors four were emphasized: problem solving, academic ability, goal orientation and motivation.

The experts related goal orientation to determination, independence, competitiveness and pride. These were not investigated. There was considerable discussion of learning style by the experts. Instruments were available for assessing learning style and problem-solving ability and were included in the main study. Neither learning style nor problem solving was a robust estimator of results in programming units. However, both the business and the science students were on average the assimilator type. They tended to be more reflective than data processors and practising computer scientists in processing information. They were a little less abstract in their approach to perceiving new information.

The problem-solving instrument used in the main study was a series of test questions drawn from sources in the literature. The scoring of questions is open to interpretation. This discretion and the diversity of problem sources might reduce the importance of problem solving in regressions. The issue of scoring may not be so important because the results in Chapter 7 were unchanged by applying a different scheme. However problem solving verged on significance as an explanator of performance in some situations. In particular it mattered among science students. In relation to business its influence may be confounded with gender (which was not the case for science students). These issues warrant further investigation. Perhaps different problem-solving skills matter in business than matter in science. Shute (1991) notes that some problem-solving skills matter more than others do in learning Pascal.

Academic ability was modeled in the main study using indicators of performance at secondary school and at tertiary level. Some of these emerged as robust indicators of examination results and overall programming performance. In the case of business students programming performance was inversely related to secondary school performance. Finding a negative association for business students is tantalizing and warrants further exploration. As noted in Chapter 7 this may be to do with subjects underlying the aggregate secondary score.
Science students' performance in the first programming unit increased with both aggregate secondary score and the average score in other tertiary units. This is consistent with other literature (Nowaczyk 1984). In the first programming unit an increase of three in indicators of ability was required to improve programming result by one mark. For the second programming unit there was almost a one-for-one increase. By the second unit secondary school performance no longer mattered.

Consensus in the focus group was that personal motivation affected performance. This was contrasted with motivation induced by the teacher. The experts considered that things such as reason for studying, knowledge of computers, enthusiasm, interest in computing and relevance to larger goals (like jobs) were important for motivation.

In the main study a number of explanators were tried. For example, the type of job that students expected to get did not influence grade in programming. Among business students, liking programming improved grades. For science students no such effect was discernible. Rather a significant impact arose if students expressed dislike of programming. This influence persisted into the second unit of programming. To the extent that previous knowledge of computing is a measure of personal motivation, the analysis in Chapter 7 reveals a positive influence on performance.

Two demographic variables, age and gender were explored extensively. Age was not linked to results for either business or science students. This however is unsurprising as 60 percent (or more) of each sample were 21 or younger. For the business students, scores on problem-solving tests were in some way linked to gender, albeit weakly. There are considerable differences between the samples. For business students performance and gender are not strongly related. To the extent that there is a linkage it acts to females' advantage. Among science students in their first programming unit there is clear evidence of discrimination in outcomes.

Perhaps this result is attributable to different cultures. Unexplained differences between subject characteristics may influence the relative importance of learning outcomes for males and females in the samples. Two cultural differences are the programming language and the hardware platform used in teaching the two groups. A third difference implied by the first two is that management decisions made about language and platform are indicative of other cultural differences that affect outcomes for females.
The business students studied Visual Basic in the Windows environment on personal computers. The science students learned C using Sun workstations. Business students certainly had the benefit of a computing environment familiar for many of them. Workstation environments by comparison were relatively foreign to the science students and required greater technical competence to ensure effective interaction.

While gender was a significant explanator in the outcomes for science students in the first programming unit it was not so for the second unit. The sample analyzed in this second unit included all of those considered in the analysis of the first programming unit. An implication and perhaps a pointer to policy is that the influence of gender diminishes in time. The period of time is relatively short and is less than one year.

The main study has relied on surveying students during the second semester unit in programming. To study performances in first semester, grades from central databases were appended to the coded survey responses. This approach produces results that suggest some factors change in importance as learning continues. This conclusion is tentative, as data on characteristics such as learning style are available only as a snapshot. It is clear that a longitudinal approach involving surveys before tertiary studies begin and then during each semester may add to knowledge of how novices acquire expertise. Further, as more and more students choose to take up computer programming studies in secondary schools, these surveys may need to be done at the school level. A longitudinal approach is certainly an area for future activity.

The students studied in Chapter 7 were committed to the study of programming. All had done at least one prerequisite unit. Thus the regression analysis illuminates success in programming for committed students. What can be done to understand lack of success? That is, what can be discovered about students who start, but do not complete a sequence in programming? These questions too can be investigated with the longitudinal approach suggested in the preceding paragraph. Understanding what leads to lack of success is of considerable economic as well as educational interest. The economic significance lies in educating Australians efficiently and effectively to meet growing demands for professionals in programming-related areas of information technology.
APPENDICES

APPENDIX A: Preliminary Survey
Letter of invitation to participate in the study in Semester 1, 1994
Questionnaire- Preliminary

APPENDIX B: Focus Group
Letter of invitation to participate in the Focus Group
Follow-up letter of thanks requesting necessary alterations

APPENDIX C: Student Interview Documents
Letter of invitation to participate in interviews in Semester 2, 1994
Consent form
Interview schedule
Questionnaire- Interview

APPENDIX D: Main Study
Letter of invitation to participate in the study in Semester 2, 1995
Questionnaire- Main Study
30th May, 1994

Dear Student,

My name is Annegret Goold. I am a student undertaking a Master of Commerce by Research at Deakin University, Burwood Campus.

This letter is an invitation to participate in my research project. The purpose of the research project is to investigate factors which affect performance in undergraduate computing. The benefits of the project will be a greater understanding of these factors including computer anxiety and playfulness with computers. There are two phases to the research and I am asking you to help me with the first phase which involves a survey. If you decide to participate then it will take you at the most twenty minutes to fill out the attached survey form. If you have withdrawn from Introduction to Information Technology or if you are under the age of eighteen years, then please ignore this letter and the survey.

The survey asks you to describe your attitude to and past experience with computers. Some biographical information is also requested. By returning the completed survey form you will be indicating your consent to participate in the survey.

You can be assured that any information that you provide in this survey will remain absolutely confidential. I promise that I will lock all of the completed surveys in a cabinet to preserve confidentiality. I will not discuss any of the information with any other person, except my supervisor, Mr. Alastair Anderson, Senior Lecturer, School of MIS. Neither I nor my supervisor will have any means of identifying you. Only aggregated results will be used for research purposes and may be reported in scientific and academic journals.

If you questions about the survey then you can contact me on 244-6924 or on 807-7523.

Thanks for helping me with my project.

Annegret Goold
Completion of this survey indicates your consent to participate in the research. Any aggregated results from the survey may be reported in scientific and academic journals. The survey is strictly confidential. Please do not write your name or any other form of identification on it.

SECTION A:
This section asks you to circle the response that best describes your feelings about or reaction to each statement as follows:

Strongly Agree = SA  Agree = A  Unsure = U  Strongly Disagree = SD

1. I am confident that I could learn microcomputer skills
   SA  A  U  D  SD

2. I am unsure of my ability to learn a computer programming language
   SA  A  U  D  SD

3. I feel apprehensive about using the microcomputer
   SA  A  U  D  SD

4. If given the opportunity to use a microcomputer, I feel as though I may damage it
   SA  A  U  D  SD

5. I have avoided microcomputers because they are unfamiliar to me
   SA  A  U  D  SD

6. I hesitate to use the computer for fear of making mistakes that I cannot correct
   SA  A  U  D  SD

7. I am unsure of my ability to interpret a computer printout
   SA  A  U  D  SD

8. I have difficulty understanding most technological matters
   SA  A  U  D  SD

9. Computer terminology sounds like confusing jargon to me
   SA  A  U  D  SD

10. Human beings will misuse the power of a computer
    SA  A  U  D  SD

11. Computers are changing the world too rapidly
    SA  A  U  D  SD

12. Our country relies too much on computers
    SA  A  U  D  SD

13. Computers dehumanize society by treating everyone as a number
    SA  A  U  D  SD
SECTION B:

This section asks you to circle the response that best matches a description of yourself when you react with computers.

Strongly Agree = SA  Agree = A  Unsure = U  Disagree = D  Strongly Disagree = SD

1. spontaneous  SA  A  U  D  SD
2. conscientious  SA  A  U  D  SD
3. unimaginative  SA  A  U  D  SD
4. experimenting  SA  A  U  D  SD
5. serious  SA  A  U  D  SD
6. bored  SA  A  U  D  SD
7. flexible  SA  A  U  D  SD
8. mechanical  SA  A  U  D  SD
9. creative  SA  A  U  D  SD
10. erratic  SA  A  U  D  SD
11. curious  SA  A  U  D  SD
12. intellectually stagnant  SA  A  U  D  SD
13. inquiring  SA  A  U  D  SD
14. routine  SA  A  U  D  SD
15. playful  SA  A  U  D  SD
16. investigative  SA  A  U  D  SD
17. constrained  SA  A  U  D  SD
18. unoriginal  SA  A  U  D  SD
19. scrutinizing  SA  A  U  D  SD
20. uninventive  SA  A  U  D  SD
21. inquisitive  SA  A  U  D  SD
22. questioning  SA  A  U  D  SD
SECTION C:

PLEASE READ THE FOLLOWING INFORMATION AND INSTRUCTIONS CAREFULLY TO ENSURE THE ACCURATE COMPLETION OF THE SURVEY

This section contains questions to find out how certain important events in our society affect different people. Each item consists of a pair of alternatives lettered “a” or “b”. Please select the statement in each pair (and only one) which you believe more strongly to be the case, as far you’re concerned. Be sure to select the one you actually believe to be true rather than the one you think you should choose or the one you would like to be true. This is a measure of personal belief. Obviously there are no right or wrong answers.

Please answer these items carefully, but do not spend too much time on any one item. Respond to each item independently when making your choice, do not be influenced by your previous choices. Be sure to find an answer for every question. In some instances you may believe each statement, or neither one. In such cases be sure to circle the one you believe more strongly to be the case.

CIRCLE ONLY ONE LETTER FOR EACH QUESTION.

1. a. Children get into trouble because their parents punish them too much.
   b. The trouble with most children nowadays is that their parents are too easy with them.

2. a. Many of the unhappy things in people's lives are partly due to bad luck.
   b. People's misfortunes result from the mistakes they make.

3. a. One of the major reasons we have wars is because people don't take enough interest in politics.
   b. There will always be wars, no matter how hard people try to prevent them.

4. a. In the long run people get the respect they deserve in this world.
   b. Unfortunately, an individual's worth often passes unrecognized no matter how hard he tries.

5. a. The idea that teachers are unfair to students is nonsense.
   b. Most students don't realise the extent to which their grades are influenced by accidental happenings.
6.  a. Without the right breaks one cannot be an effective leader.
b. Capable people who fail to become leaders have not taken advantage of
their opportunities.

7.  a. No matter how hard you try some people just don't like you.
b. People who can't get others to like them don't understand how to get
along with others.

8.  a. Heredity plays the major role in determining one's personality.
b. It is one's experiences in life which determine what they're like.

9.  a. I have often found that what is going to happen will happen.
b. Trusting to fate has never turned out as well for me as making a decision
to take a definite course of action.

10. a. In the case of a well prepared student there is rarely if ever such a thing
as an unfair test.
b. Many times exam questions tend to be so unrelated to course work that
studying is really useless.

11. a. Becoming a success is a matter of hard work, luck has little or nothing to
do with it.
b. Getting a good job depends mainly on being the right place at the right
time.

12. a. The average citizen can have an influence in government decisions.
b. The few people in power run this world, and there is not much the little
guy can do about it.

13. a. When I make plans, I am almost certain that I can make them work.
b. It is not always wise to plan too far ahead because things turn out to be a
matter of good or bad fortune anyhow.

14. a. There are certain people who are just no good.
b. There is some good in everybody.

15. a. In my case getting what I do or want has little or nothing to do with luck.
b. Many times we might just as well decide what to do by flipping a coin.

16. a. Who gets to be the boss often depends on who was lucky enough to be
in the right place first.
b. Getting people to do the right thing depends on ability, luck has little or
nothing to do with it.
17. a. As far as world affairs are concerned, most of us are the victims of forces we can neither understand, nor control.
   b. By taking an active part in political and social affairs the people can control world events.
18. a. Most people don't realise the extent to which their lives are controlled by accidental happenings.
   b. There is really no such thing as "luck".
19. a. One should always be willing to admit mistakes.
   b. It is usually best to cover up one's mistakes.
20. a. It is hard to know whether or not a person really likes you.
   b. How many friends you have depends on how nice a person you are.
21. a. In the long run the bad things that happen to us are balanced by the good ones.
   b. Most misfortunes are the result of lack of ability, ignorance, laziness, or all three.
22. a. With enough effort we can wipe out political corruption.
   b. It is difficult for people to have much control over the things politicians do in office.
23. a. Sometimes I can't understand how teachers arrive at the grades they give.
   b. There is a direct connection between how hard I study and the grades I get.
24. a. A good leader expects people to decide for themselves what they should do.
   b. A good leader makes it clear to everybody what his or her jobs are.
25. a. Many times I feel that I have little influence over the things that happen to me.
   b. It is impossible for me to believe that change or luck plays an important role in my life.
26. a. People are lonely because they don't try to be friendly.
   b. There's not much use in trying too hard to please people, if they like you, they like you.
27. a. There is too much emphasis on athletics in high school.
   b. Team sports are an excellent way to build character.
28. a. What happens to me is my own doing.
   b. Sometimes I feel that I don't have enough control over the direction my life is taking.

29. a. Most of the time I can't understand why politicians behave the way they do.
   b. In the long run the people are responsible for bad government on a national as well as on a local level.
SECTION D:

This section asks you about your computing experience. Please answer each question by placing a tick in the appropriate box or written responses when requested.

1. Do you or the persons with whom you live own a microcomputer?
   Yes ☐
   No ☐

   If you answered No to this question, go to question 5

2. Was the microcomputer bought because of your tertiary studies?
   Yes ☐
   No ☐

3. Please indicate the type of microcomputer that you (or the people with whom you live) own
   IBM (or IBM compatible) ☐
   Macintosh ☐
   Amiga ☐
   Other ☐

4. Is the microcomputer equipped with a modem?
   Yes ☐
   No ☐

5. Before enrolling in Business Information Systems, what was your knowledge of the following:

   Windows
   None ☐ A Little ☐ Passable ☐ Quite Good ☐ Superior ☐

   Spreadsheets
   Lotus 1-2-3 ☐ A Little ☐ Passable ☐ Quite Good ☐ Superior ☐
   Excel ☐ A Little ☐ Passable ☐ Quite Good ☐ Superior ☐
   Microsoft Works ☐ A Little ☐ Passable ☐ Quite Good ☐ Superior ☐
   Other (please specify) ☐ A Little ☐ Passable ☐ Quite Good ☐ Superior ☐

   Word-processing
   WordPerfect ☐ A Little ☐ Passable ☐ Quite Good ☐ Superior ☐
   Microsoft Word ☐ A Little ☐ Passable ☐ Quite Good ☐ Superior ☐
   Microsoft Works ☐ A Little ☐ Passable ☐ Quite Good ☐ Superior ☐
   Word Windows ☐ A Little ☐ Passable ☐ Quite Good ☐ Superior ☐
   Other (please specify) ☐ A Little ☐ Passable ☐ Quite Good ☐ Superior ☐
Database Programs

<table>
<thead>
<tr>
<th>Database</th>
<th>☐</th>
<th>☐</th>
<th>☐</th>
<th>☐</th>
<th>☐</th>
<th>☐</th>
<th>☐</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBase3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBase4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Access</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Microsoft Works</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other (please specify)</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

6. Before enrolling in Business Information Systems how would you have described your overall computing knowledge?

   None ☐
   A little ☐
   Passable ☐
   Quite Good ☐
   Superior ☐

If you answered **NONE** to this question, go to question 9

7. How would you describe your computer programming knowledge?

   None ☐
   A little ☐
   Passable ☐
   Quite Good ☐
   Superior ☐

If you answered **NONE** to this question, go to question 9

8. What computer programming language(s) have you used?

   BASIC ☐
   Pascal ☐
   C ☐
   Logo ☐
   (please specify any other)

9. Do you use computers in any paid employment?

   Yes ☐
   No ☐

If you answered **NO** to this question, go to question 11
10. Please specify how computers are used by you in your job.
(Please tick one box only)
   User ☐
   Programmer ☐
   Systems Analyst ☐

11. What do you like most about computers and computing?

   ____________________________________________________________

   ____________________________________________________________

   ____________________________________________________________

12. What do you like least about computers and computing?

   ____________________________________________________________

   ____________________________________________________________

   ____________________________________________________________
SECTION E:

This section asks you questions about yourself and your educational background. Please answer each question by placing a tick in the appropriate box and providing written responses where requested.

1. Gender
   Male
   Female

2. Age
   < 20
   20 < 25
   25 < 30
   30 < 35
   35 < 40
   40 < 45
   45 < 50
   50+

3. Status
   Full-time
   Part-time

4. What is your course major?
   Accounting and Finance
   Administrative Management
   Management Information Systems
   Economics
   Insurance
   Law
   Management
   Marketing
   Human Resource Management
   Sports Management
   Arts/Commerce
   Teaching/Commerce
   Single Subject
   Other ____________________________  


5. Why did you choose the course major as your area of study?


6. Have you previously been enrolled in Information Systems 1?
Yes
No

7. Why did you choose Deakin University for your studies?


8. If you completed VCE studies last year, what were the subjects you attempted and what scores (standardized) did you obtain?

<table>
<thead>
<tr>
<th>a. Subject</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

b. Secondary School attended


9. If you applied for your course through VTAC, what preference did you give the course you are now enrolled in?

First  □
Second □
Third  □
Fourth □
Fifth  □
Sixth  □
Seventh □
Eighth □

THANK YOU FOR YOUR ASSISTANCE IN COMPLETING THIS SURVEY.

Please return the survey form in the Reply Paid Envelope. Either mail it or place it in the box provided at the:

Faculty of Management Office
Level 2, Building B
Monday May 2, 1994

Dear

One of our Masters by Research students, Annegret Goold, is exploring the topic ‘Factors Affecting Success in Elementary Computing’. As part of the research methodology she needs to ascertain the viewpoint of experts in the field as to those factors which affect success in undergraduate computer programming.

I would like to invite you to take part in a session which will attempt to answer the following –

‘What are the factors which determine success in elementary computer programming?’

The session will be held on Wednesday 18th May at 5 PM in the Blue Room, Building B, Faculty of Management, Burwood Campus (refer to map enclosed). The session will take about two hours and it will be an opportunity for you to meet colleagues from other tertiary institutions who are teaching computer programming. Light refreshments will be served.

Please direct any enquiries and your intention to attend to Mr. Alastair Anderson on 244 6532.

I hope to welcome you on the 18th.

A Anderson
Once again many thanks for giving up your time to attend the brainstorming session on the question of success in elementary computer programming.

We think that it was resolved that success means achieving more than a pass in an elementary programming subject, say a programming subject taught at the first year level of an undergraduate program. Keeping this in mind, we would like you to give us just a little more assistance if you can.

Enclosed with this letter are three cognitive maps which we have constructed on the basis of the ideas generated at the meeting which you attended. The three maps summarise three factors which we have extracted as affecting success in elementary computer programming. These factors are:

Map 1  (69)  Personal Motivation...Teacher Motivation
Map 2  (2)  Problem Solving Skill
         (32)  Goal Orientation
Map 3  (23)  Basic Academic Ability

The first map has as the superordinate construct (dominant influence) personal motivation...teacher motivation. The...between personal motivation and teacher motivation should be read as "rather than". In our lingo we call personal motivation the emergent pole of a construct and teacher motivation the submerged pole. So we see success in elementary computer programming as a function of personal motivation rather than teacher motivation. In our view, personal motivation is dominant.

In Map 1 you will notice arrows all over the place. The arrows link issues/personal attributes and goals with the superordinate construct for this map. Where an arrow has a negative sign attached to it we infer that a specific idea is linked to the submerged pole of a construct. You will see for example that in Map 1 ‘(61) Statistically more students means more success’ is linked to teacher motivation.
We would like you to peruse the three maps and make any alterations or adjustments you like. Please feel free to write notes and comments on the maps. If you would like to talk about the maps over the phone then please do so (contact phone numbers shown below). In fact we would encourage you to so as this may make your task easier.

Also included with this letter is a list of 17 words/statements. On the right side of this list is a column with the heading Opposite/Contrast. If you can, please supply an opposite or a contrast for each of these statements in the column provided. There are no right or wrong answers, just your individual ideas.

Once again thanks for your time and please excuse this letter if it is a little vague but these things are somewhat difficult to do at a distance. Please send your maps, completed responses for the constructs and any other feedback within the next seven days (by mail or by fax) to:

Alastair Anderson
School of Management Information Systems
Faculty of Management
Deakin University
Burwood Campus
FAX 244-6520

Feel free to phone Alastair at any time. The Deakin phone number is 244-6532 and the home phone number is 499-4916. A notice of a forthcoming seminar is included. You are most welcome to attend.

Regards,

Alastair Anderson, Bernard Petter & Annegret Goold
Please supply the bipolar constructs or contrasts to the words/statements shown below.

<table>
<thead>
<tr>
<th>Word/Statement</th>
<th>Opposite/Contrast</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Parental Pressure</td>
<td></td>
</tr>
<tr>
<td>2. Practical hands-on work</td>
<td></td>
</tr>
<tr>
<td>3. Requires enthusiasm about computers</td>
<td></td>
</tr>
<tr>
<td>4. Clear thinking</td>
<td></td>
</tr>
<tr>
<td>5. Step by step approach</td>
<td></td>
</tr>
<tr>
<td>6. Attention to detail</td>
<td></td>
</tr>
<tr>
<td>7. Thoroughness</td>
<td></td>
</tr>
<tr>
<td>8. Capacity for precision</td>
<td></td>
</tr>
<tr>
<td>9. Work independently</td>
<td></td>
</tr>
<tr>
<td>10. Perserverance with difficulties</td>
<td></td>
</tr>
<tr>
<td>11. Capable of selecting models from various disciplines</td>
<td></td>
</tr>
<tr>
<td>12. Ability to see the whole from the parts</td>
<td></td>
</tr>
<tr>
<td>13. Ability to see the parts that make the whole</td>
<td></td>
</tr>
<tr>
<td>14. Competent at building mental models</td>
<td></td>
</tr>
<tr>
<td>15. Ability to decompose a problem into sub problems</td>
<td></td>
</tr>
<tr>
<td>16. Ability to re-use existing problems</td>
<td></td>
</tr>
<tr>
<td>17. Ability to handle and understand repetition</td>
<td></td>
</tr>
</tbody>
</table>
Dear Student,

My name is Annegret Goold. I am a student undertaking a Master of Commerce by Research at Deakin University, Burwood Campus.

This letter is an invitation to participate in my research project. The purpose of the research project is to investigate factors which affect performance in Business Computing. The benefits of the project will be a greater understanding of how to improve teaching methods for business computing. There are two phases to the research. I am asking you to help me with the second phase which will involve an interview. The interview will take at most one hour. The interviews will be semi-structured and my purpose is to hear your views about computers and computer programming. I may prompt you from time to time or ask you for clarification. If you have any questions during the course of the interview then please ask me. If there is any question which I ask and you would prefer not to answer it, then just say so and we will move on. You may stop the interview at any time if you want to.

I want you to be sure that any information that you provide to me in this interview will remain absolutely confidential. With your consent I would like to tape these interviews and I promise that I will lock the tapes and any other materials in a cabinet to preserve confidentiality.

I will not discuss any of the information with any other person, except my supervisor (Mr. Alastair Anderson, Senior Lecturer, School of MIS) or a person who may retype the interviews for me. My supervisor and the typist will have no means of identifying you. If you like you can use a nickname during the interview.

If you want to contact me my telephone contact number at work is 244-6924 or at home 807-7523.

Thanks for helping me with my project.
DEAKIN UNIVERSITY
ETHICS COMMITTEE
CONSENT FORM (For Interview Participants)

I, ................................................ of ..........................................................

Herby consent to be a subject of a human research study to be undertaken
by Ms. Annegret Goold

I acknowledge
1. That the aims, methods, and anticipated benefits, and possible hazards of the research study, have been explained to me.

2. That I voluntarily and freely give my consent to my participation in such research study.

3. That I have agreed/not agreed to give my consent to the taping of this interview.

4. I understand that aggregated results will be used for research purposes and may be reported in scientific journals and academic journals.

5. Individual results will not be released to any person except at my request and on my authorisation.

6. That I am free to withdraw my consent at any time during the study, in which event my participation in the research study will immediately cease and any information obtained from me will not be used.

Signature: ....................
Date: ..........................
INTERVIEW SCHEDULE

1. Introduction and Description of Study

2. Administration
   - Letter and consent form

3. Background of student
   - B1S and SD1 results
   - Subjects done at VCE level?
   - Subjects you liked most then? now?
   - Resources Computer/modem/software at home?
   - Friends? / Family members / Work?

4. Why computing?

5. Why commerce computing as opposed to computer science?

6. Why Deakin?

7. What were your expectations of computer programming prior to this semester?
   - Has this changed? How?

8. Describe how you learn computer programming.
   - value of lectures?
   - value of tutes?
   - value of assignment work?
   - time spent on each of the above?
   - value of textbooks?
   - other resources used?

8. Describe the steps you undertake in doing an assignment.
   - where are the problems and how do you solve them?
   - how do you debug your programs?
   - has your technique changed since assignment 1?

9. What are your strengths in computer programming?

10. Where are your weaknesses?

11. Do you plan to continue with computing?
    - where do you see yourself in five years time?
    - describe your job

12. Survey Form
Completion of this survey indicates your consent to participate in the research. Any aggregated results from the survey may be reported in scientific and academic journals. The survey is strictly confidential. Please do not write your name or any other form of identification on it.

SECTION A:
This section asks you to circle the response that best describes your feelings about or reaction to each statement as follows:

Strongly Agree = SA    Agree = A    Unsure = U    Strongly Disagree = SD
Disagree = D

1. I am confident that I could learn microcomputer skills
   SA  A  U  D  SD

2. I am unsure of my ability to learn a computer programming language
   SA  A  U  D  SD

3. I feel apprehensive about using the microcomputer
   SA  A  U  D  SD

4. If given the opportunity to use a microcomputer, I feel as though I may damage it
   SA  A  U  D  SD

5. I have avoided microcomputers because they are unfamiliar to me
   SA  A  U  D  SD

6. I hesitate to use the computer for fear of making mistakes that I cannot correct
   SA  A  U  D  SD

7. I am unsure of my ability to interpret a computer printout
   SA  A  U  D  SD

8. I have difficulty understanding most technological matters
   SA  A  U  D  SD

9. Computer terminology sounds like confusing jargon to me
   SA  A  U  D  SD

10. Human beings will misuse the power of a computer
    SA  A  U  D  SD

11. Computers are changing the world too rapidly
    SA  A  U  D  SD

12. Our country relies too much on computers
    SA  A  U  D  SD

13. Computers dehumanize society by treating everyone as a number
    SA  A  U  D  SD
SECTION B:

This section asks you to circle the response that best matches a description of yourself when you react with computers.

Strongly Agree = SA Agree = A Unsure = U Disagree = D Strongly Disagree = SD

1. spontaneous SA A U D SD
2. conscientious SA A U D SD
3. unimaginative SA A U D SD
4. experimenting SA A U D SD
5. serious SA A U D SD
6. bored SA A U D SD
7. flexible SA A U D SD
8. mechanical SA A U D SD
9. creative SA A U D SD
10. erratic SA A U D SD
11. curious SA A U D SD
12. intellectually stagnant SA A U D SD
13. inquiring SA A U D SD
14. routine SA A U D SD
15. playful SA A U D SD
16. investigative SA A U D SD
17. constrained SA A U D SD
18. unoriginal SA A U D SD
19. scrutinizing SA A U D SD
20. un inventive SA A U D SD
21. inquisitive SA A U D SD
22. questioning SA A U D SD
SECTION C:

PLEASE READ THE FOLLOWING INFORMATION AND INSTRUCTIONS CAREFULLY TO ENSURE THE ACCURATE COMPLETION OF THE SURVEY

This section contains questions to find out how certain important events in our society affect different people. Each item consists of a pair of alternatives lettered "a" or "b". Please select the statement in each pair (and only one) which you believe more strongly to be the case, as far you're concerned. Be sure to select the one you actually believe to be true rather than the one you think you should choose or the one you would like to be true. This is a measure of personal belief. Obviously there are no right or wrong answers.

Please answer these items carefully, but do not spend too much time on any one item. Respond to each item independently when making your choice; do not be influenced by your previous choices. Be sure to find an answer for every question. In some instances you may believe each statement, or neither one. In such cases be sure to circle the one you believe more strongly to be the case.

CIRCLE ONLY ONE LETTER FOR EACH QUESTION.

1. a. Children get into trouble because their parents punish them too much.
    b. The trouble with most children nowadays is that their parents are too easy with them.

2. a. Many of the unhappy things in people's lives are partly due to bad luck.
    b. People's misfortunes result from the mistakes they make.

3. a. One of the major reasons we have wars is because people don't take enough interest in politics.
    b. There will always be wars, no matter how hard people try to prevent them.

4. a. In the long run people get the respect they deserve in this world.
    b. Unfortunately, an individual's worth often passes unrecognized no matter how hard he tries.

5. a. The idea that teachers are unfair to students is nonsense.
    b. Most students don't realise the extent to which their grades are influenced by accidental happenings.
6. a. Without the right breaks one cannot be an effective leader.
   b. Capable people who fail to become leaders have not taken advantage of their opportunities.

7. a. No matter how hard you try some people just don't like you.
   b. People who can't get others to like them don't understand how to get along with others.

8. a. Heredity plays the major role in determining one's personality.
   b. It is one's experiences in life which determine what they're like.

9. a. I have often found that what is going to happen will happen.
   b. Trusting to fate has never turned out as well for me as making a decision to take a definite course of action.

10. a. In the case of a well prepared student there is rarely if ever such a thing as an unfair test.
   b. Many times exam questions tend to be so unrelated to course work that studying is really useless.

11. a. Becoming a success is a matter of hard work, luck has little or nothing to do with it.
    b. Getting a good job depends mainly on being the right place at the right time.

12. a. The average citizen can have an influence in government decisions.
    b. The few people in power run this world, and there is not much the little guy can do about it.

13. a. When I make plans, I am almost certain that I can make them work.
    b. It is not always wise to plan too far ahead because things turn out to be a matter of good or bad fortune anyhow.

14. a. There are certain people who are just no good.
    b. There is some good in everybody.

15. a. In my case getting what I do or want has little or nothing to do with luck.
    b. Many times we might just as well decide what to do by flipping a coin.
16. a. Who gets to be the boss often depends on who was lucky enough to be in the right place first.
   b. Getting people to do the right thing depends on ability, luck has little or nothing to do with it.

17. a. As far as world affairs are concerned, most of us are the victims of forces we can neither understand, nor control.
   b. By taking an active part in political and social affairs the people can control world events.

18. a. Most people don't realise the extent to which their lives are controlled by accidental happenings.
   b. There is really no such thing as "luck".

19. a. One should always be willing to admit mistakes.
   b. It is usually best to cover up one's mistakes.

20. a. It is hard to know whether or not a person really likes you.
   b. How many friends you have depends on how nice a person you are.

21. a. In the long run the bad things that happen to us are balanced by the good ones.
   b. Most misfortunes are the result of lack of ability, ignorance, laziness, or all three.

22. a. With enough effort we can wipe out political corruption.
   b. It is difficult for people to have much control over the things politicians do in office.

23. a. Sometimes I can't understand how teachers arrive at the grades they give.
   b. There is a direct connection between how hard I study and the grades I get.

24. a. A good leader expects people to decide for themselves what they should do.
   b. A good leader makes it clear to everybody what his or her jobs are.
25. a. Many times I feel that I have little influence over the things that happen to me.
b. It is impossible for me to believe that change or luck plays an important role in my life.

26. a. People are lonely because they don't try to be friendly.
b. There's not much use in trying too hard to please people, if they like you, they like you.

27. a. There is too much emphasis on athletics in high school.
b. Team sports are an excellent way to build character.

28. a. What happens to me is my own doing.
b. Sometimes I feel that I don't have enough control over the direction my life is taking.

29. a. Most of the time I can't understand why politicians behave the way they do.
b. In the long run the people are responsible for bad government on a national as well as on a local level.
**SECTION D: COMPUTER PROGRAMMING FACTORS**

This section asks you to rate the importance of the following factors in mastering computer programming:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Strongly Agree</th>
<th>Agree</th>
<th>Unsure</th>
<th>Disagree</th>
<th>Strongly Disagree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Prior experience with computers</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td>2.</td>
<td>Enthusiasm about computers</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td>3.</td>
<td>Ability to play computer games</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td>4.</td>
<td>Previous experience with application programs</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td>5.</td>
<td>Maths ability</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td>6.</td>
<td>Motivation to succeed</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td>7.</td>
<td>Lectures</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td>8.</td>
<td>Tutorials</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td>9.</td>
<td>Discussion with peers</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td>10.</td>
<td>Textbooks</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td>11.</td>
<td>Ability to break down a problem into smaller tasks</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td>12.</td>
<td>Intelligence</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td>13.</td>
<td>Subjects chosen at VCE level</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td>14.</td>
<td>Ability to construct algorithms</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td>15.</td>
<td>Computer anxiety</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td>16.</td>
<td>Method of teaching computer programming</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td>17.</td>
<td>Thoroughness</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td>18.</td>
<td>Clear-thinking</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td>19.</td>
<td>Problem-solving ability</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td>20.</td>
<td>Ability to follow directions</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td>21.</td>
<td>Precision</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td>22.</td>
<td>Ability to analyse problems</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td></td>
<td>Strongly Agree</td>
<td>Agree</td>
<td>Unsure</td>
<td>Disagree</td>
<td>Strongly Disagree</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>----------------</td>
<td>-------</td>
<td>--------</td>
<td>----------</td>
<td>------------------</td>
<td></td>
</tr>
<tr>
<td>23.</td>
<td>Methodical approach</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td>24.</td>
<td>Pride in finished product</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td>25.</td>
<td>Perseverance</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td>26.</td>
<td>Ability to modify or re-use existing programs</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td>27.</td>
<td>Reasons for undertaking the subject</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td>28.</td>
<td>Ability to explore and experiment</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td>29.</td>
<td>Ability to read and understand programs</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td>30.</td>
<td>Understanding the use of &quot;arrays&quot;</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td>31.</td>
<td>Ability to understand &quot;repetition&quot;</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
<tr>
<td>32.</td>
<td>Creativity</td>
<td>SA</td>
<td>A</td>
<td>U</td>
<td>D</td>
<td>SD</td>
</tr>
</tbody>
</table>
Dear Student,

My name is Annegret Goold and I am a student undertaking a Master of Commerce by Research at Deakin University, Bunwood Campus.

This letter is an invitation to participate in a survey which is part of my research project. The purpose of the research project is to investigate factors which affect performance in computer programming. The benefits of the project will be a greater understanding of these factors and the implications they may have on teaching practice.

If you are under the age of eighteen years, please do not participate in the survey.

The survey consists of four sections. Section 1 asks you about your learning style preferences, Section 2 asks you about your previous experience with computers and computer programming, Section 3 asks you to complete some short problem-solving exercises and Section 4 asks you for some biographical information. Sections 1, 2 and 4 will only take a few minutes to complete. Section 3 should only take about 15 minutes to complete.

You can be assured that whilst I will know your identity, any information that you provide in this survey will remain absolutely confidential. I will not provide any information whatsoever to any other person. I am not your tutor nor your lecturer and I have no authority in respect of your assessment for this unit. The answers you provide for Section 3 which involves problem-solving will have no bearing whatsoever on your assessment for this unit.

If you have any questions about this project then you can contact me on 9 244 7159.

Thank you very much for participating

Annegret Goold
Thank you for agreeing to be part of this survey.

Please fill in the consent form below before answering survey questions. This is a requirement of the Deakin University Ethics Committee to ensure that you have a degree of protection of privacy for personal information. It is the standard practice for surveys of this nature.

I, .................................................. Student Number ......................

of .................................................................................................

Hereby consent to be a subject of a human research study to be undertaken by Annegret Goold
and I understand that the purpose of the research is to investigate factors which affect performance in computer programming. The factors being investigated by this questionnaire are computer experience, computer knowledge, educational background, learning style and problem-solving ability. These factors will be correlated to computer programming performance in the end-of-semester examination.

I acknowledge that

1. Upon receipt, my questionnaire will be coded and my name, student number and address kept separate from it.
2. Any information that I provide will not be made public in any form that could reveal my identity to an outside party ie. That I will remain fully anonymous.
3. Aggregated results will be used for research purposes and may be reported in scientific and academic journals.
4. Individual results will not be released to any person except at my request and on my authorisation.
5. That I am free to withdraw my consent at any time during the study in which event my participation in the research study will immediately cease and any information obtained from me will not be used.

Signature ................................................................. Date ......./ ....../ .......
Learning-Style Inventory: Instructions

The Learning-Style Inventory describes the way you learn and how you deal with ideas and day-to-day situations in your life. Below are 12 sentences with a choice of four endings. Rank the endings for each sentence according to how well you think each one fits how you would go about learning something. Try to recall some recent situations where you had to learn something new, perhaps in your job. Then, using the spaces provided, rank a ‘4’ for the sentence ending that describes how you learn best, down to a ‘1’ for the sentence ending that seems least like the way you would learn. Be sure to rank all the endings for each sentence unit. Please do not make ties.

Example of completed sentence set:

<table>
<thead>
<tr>
<th>e.g.</th>
<th>When I learn:</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>When I learn:</td>
<td>— I like to deal with my feelings</td>
<td>— I like to watch and listen</td>
<td>— I like to think about ideas</td>
</tr>
<tr>
<td>2.</td>
<td>I learn best when:</td>
<td>— I trust my hunches and feelings</td>
<td>— I listen and watch carefully</td>
<td>— I rely on logical thinking</td>
</tr>
<tr>
<td>3.</td>
<td>When I am learning:</td>
<td>— I have strong feelings and reactions</td>
<td>— I am quiet and reserved</td>
<td>— I tend to reason things out</td>
</tr>
<tr>
<td>4.</td>
<td>I learn by:</td>
<td>— feeling</td>
<td>— watching</td>
<td>— thinking</td>
</tr>
<tr>
<td>5.</td>
<td>When I learn:</td>
<td>— I am open to new experiences</td>
<td>— I look at all sides of issues</td>
<td>— I like to analyse things, break them down into their parts</td>
</tr>
<tr>
<td>6.</td>
<td>When I am learning:</td>
<td>— I am an intuitive person</td>
<td>— I am an observing person</td>
<td>— I am a logical person</td>
</tr>
<tr>
<td>7.</td>
<td>I learn best from:</td>
<td>— personal relationships</td>
<td>— observation</td>
<td>— rational theories</td>
</tr>
<tr>
<td>8.</td>
<td>When I learn:</td>
<td>— I feel personally involved in things</td>
<td>— I take my time before acting</td>
<td>— I like ideas and theories</td>
</tr>
<tr>
<td>9.</td>
<td>I learn best when:</td>
<td>— I rely on my feelings</td>
<td>— I rely on my observations</td>
<td>— I rely on my ideas</td>
</tr>
<tr>
<td>10.</td>
<td>When I am learning:</td>
<td>— I am an accepting person</td>
<td>— I am a reserved person</td>
<td>— I am a rational person</td>
</tr>
<tr>
<td>11.</td>
<td>When I learn:</td>
<td>— I get involved</td>
<td>— I like to observe</td>
<td>— I evaluate things</td>
</tr>
<tr>
<td>12.</td>
<td>I learn best when:</td>
<td>— I am receptive and open minded</td>
<td>— I am careful</td>
<td>— I analyse ideas</td>
</tr>
</tbody>
</table>
Section 2: This section asks you about your computing experience and how you feel about computer programming in particular. Please answer each question by placing a tick in the appropriate box or written answers where required.

1. Prior to commencing your studies at Deakin how would you have described your computer programming knowledge?
   - Very good
   - Quite good
   - Average
   - Minimal
   - None
   *(If NONE go to Question 6)*

2. Where did you obtain this knowledge?
   ........................................................................................................................................
   ........................................................................................................................................

3. What programming language(s) did/do you use?
   ........................................................................................................................................
   ........................................................................................................................................

4. Do you use computers in any paid employment?
   - Yes
   - No *(If no, go to Question 6)*

5. Please specify how computers are used by you in your job. *(Tick one)*
   - User
   - Systems analyst
   - Other *(specify)*

6. How would you describe your programming knowledge now?
   - Very good
   - Quite good
   - Average
   - Minimal

7. How much do you enjoy computer programming?
   - I love it
   - It's OK
   - I like it quite a lot
   - I don't like it at all
   - I like it a bit

8. What grade do you expect to get for the end-of-semester examination in SCC181?
   - HD
   - D
   - C
   - P
   - N

9. What would you like to do (jobwise) in the future?
   ........................................................................................................................................
   ........................................................................................................................................
Section 3: This section asks you to demonstrate your problem-solving skills and do not be concerned if you cannot answer the questions. Your answers will not be used for assessment purposes in this unit nor will any lecturer or tutor involved in your assessment have access to the answers you give.

1. Complete the following: 5:125 as 4:________
   (a) 100 (b) 120 (c) 124 (d) 160

   *Answer:* ..................

2. Complete the following: BED: 254 as ACE: _________
   (a) 146 (b) 135 (c) 157 (d) 160

   *Answer:* ..................

3. If \( x = y, y : : z \) which of the following is NOT true?
   (a) \( x + y = z \) (b) \( x - 1 = z - 1 \) (c) \( y + 1 = z \) (d) \( z - 1 = x \)

   *Answer:* ..................

4. Complete the following sequence: 1, 8, 22, 50, ________
   (a) 64 (b) 106 (c) 134 (d) 100

   *Answer:* ..................

5. If the total length of the Loch Ness monster is 20 metres and half its own length, how long is it?

   *Answer:* ..................

6. A lake has a top view in the shape of a square. At each corner of the lake there is a tree. You wish to enlarge the lake so that the top view is now that of a square exactly twice the area of the original square. You don't want any of the trees to be surrounded by water nor do you want to move them. How is this possible? (Draw your answer.)

   *Answer:*
Section 3 (Continued):

7. Follow the instructions below, using the boxes and their values as indicated. When you put a number in a box it is understood that the previous value has been erased.

<table>
<thead>
<tr>
<th>Box</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>3</td>
<td>7</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>12</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

Start

1. Put (number in box 8) into box 1.

2. Add: (number in box 1) + (number in box 2), put result in box 3.

3. Change Instruction 2: increase the second box number mentioned in it, by 1.

4. Is the second box-number mentioned in Instruction 2, greater than (number in box 7)?

No

Yes

End

What number is now in box 1?

Answer: .................

8. A father and two sons have to cross a crocodile infested river. They find a boat which can be rowed across but which sinks if the load is more than 100 kg. Each son weighs 50 kg and the father weighs twice as much as a son. How do father and sons cross the river, assuming that they don’t want to swim or hang onto the side of the boat and that the boat needs to be occupied by at least one person on a river crossing?

Answer: .................................................................

.................................................................

.................................................................

9. A car rental service charges $50/day and 30 cents a kilometre to hire a car. Write down an expression which will give the cost in dollars of renting a car D days to travel K kilometres.

Answer: .................................................................
Section 4: This section asks you questions about yourself and your educational background. Please answer each question by placing a tick in the appropriate box or written answers where required.

1. Are you studying full time?
   - [ ] Yes
   - [ ] No

2. What year were you born?
   19....

3. If you applied for your course through VTAC, what preference did you give the course you are now enrolled in?
   - [ ] One
   - [ ] Two
   - [ ] Three
   - [ ] Four
   - [ ] Five
   - [ ] Six
   - [ ] Seven
   - [ ] Eight

4. Supposing you had the necessary requirements (sufficient TER score etc) for entry into any other course, what course would you have chosen to do?
   - [ ] Chosen the same course at Deakin University
   - [ ] Chosen a similar course at another institution
     - Which institution? ..................
   - [ ] Chosen another course entirely
     - Which course? ..................
     - Which institution? ..................

5. What changes (if any) are you planning to make regarding your course enrolments?
   ........................................................................................................

The remaining questions are only for students who completed VCE last year.

6. What was your TER score (adjusted)? .................................

7. What mark did you obtain in English 3 & 4? .............................

8. If you attempted any of the following subjects in the VCE please indicate the mark you obtained.

<table>
<thead>
<tr>
<th>Subject</th>
<th>Mark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Further Maths 3 and 4</td>
<td>......</td>
</tr>
<tr>
<td>Maths Methods</td>
<td>......</td>
</tr>
<tr>
<td>Specialist Maths</td>
<td>......</td>
</tr>
<tr>
<td>Information Processing &amp; Management</td>
<td>......</td>
</tr>
<tr>
<td>Information Systems 3 and 4</td>
<td>......</td>
</tr>
</tbody>
</table>
REFERENCES


Crebbin, W and CAUT Project Team (1994) Teaching in Higher Education for Active and Lifelong Learning, University of Ballarat, Ballarat.


Deakin University Undergraduate Studies Handbook (1994) Volume 1, Deakin University, Geelong.


Miller, G A (1956) The Magical Number Seven Plus or Minus Two: Some Limits on our Capacity for Processing Information, Psychological Review, 63 2, pp. 81-97.


Definitive and Parsimonious Measure, Human Factors, 32 4, pp. 477-491.

(eds) Educational Computing and Problem Solving, pp. 91-104 The Haworth Press,
New York.

Reed, W M and J.K Burton (1988 (eds) Educational Computing and Problem Solving,
The Haworth Press, New York.

Soloway and S Iyengar (eds) Empirical Studies of Programmers, Norwood, New
Jersey.

Rogalski, J and R Samurcay (1990) Acquisition of Programming Knowledge and Skills,
in J -M Hoc et al. (eds), Psychology of Programming, Academic Press, San Diego,
California, pp. 157-174.

Rotter, J B (1966) Generalized Expectancies for Internal versus External Control of
Reinforcement, Psychological Monographs, 80 1, pp. 1-28.

Rowe, H A (1985) Problem Solving and Intelligence, Lawrence Erlbaum Associates
Inc., New Jersey.

Programming Language, International Journal of Man-Machine Studies, 37, pp. 191-
214.

Methods, Perceptual and Motor Skills, 72, pp. 243-248.

Severiens, S and G T M Ten Dam (1994) Gender Differences in Learning Style: A

Shackleford, R and A Badre (1993) Why Can't Smart Students Solve Simple
Programming Problems? International Journal of Man-Machine Studies, 38, pp. 985-
987.

pp. 689-708.

Shneiderman, B (1980) Software Psychology: Human Factors in Computer and

Shneiderman, B (1986) Empirical Studies of Programmers: The Territory, Paths and
Destinations, in E Soloway and S Iyengar (eds) Empirical Studies of Programmers,
Ablex Publishing Corporation, Norwood, New Jersey.

Shneiderman, B and R Mayer (1979) Syntactic/Semantic Interactions in Programmer
Behavior: A Model and Experimental Results, International Journal of Computer and
Information Sciences, 8 3.


